Stig F. Mjølsnes Sjouke Mauw
Sokratis K. Katsikas (Eds.)

Public Key Infrastructure

5th European PKI Workshop:
Theory and Practice, EuroPKI 2008
Trondheim, Norway, June 16-17, 2008
Proceedings

Springer

Volume Editors

Stig F. Mjølsnes
Norwegian University of Science & Technology (NTNU)
Department of Telematics
7491 Trondheim, Norway
E-mail: sfm@item.ntnu.no

Sjouke Mauw
Université du Luxembourg
Faculté des Sciences, de la Technologie et de la Communication (FSTC)
6 rue Richard Coudenhove-Kalergi, 1359 Luxembourg-Kirchberg, Luxembourg
E-mail: sjouke.mauw@uni.lu

Sokratis K. Katsikas
University of Piraeus
Department of Technology Education & Digital Systems
150 Androutsou St., 18532 Piraeus, Greece
E-mail: ska@unipi.gr

Library of Congress Control Number: 2008928842

CR Subject Classification (1998): E.3, D.4.6, C.2.0, F.2.1, H.3, H.4, K.4.4, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-540-69484-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-69484-7 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12320028 06/3180 5 4 3 2 1 0

Preface

This book contains the proceedings of the 5th European Public Key Infrastructure Workshop: Theory and Practice, EuroPKI 2008, which was held on the NTNU campus Gløshaugen in Trondheim, Norway, in June 2008.

The EuroPKI workshop series focuses on all research and practice aspects of public key infrastructures, services and applications, and welcomes original research papers and excellent survey contributions from academia, government, and industry.

Simply put, public keys are easier to distribute than secret keys. Nevertheless, constructing effective, practical, secure and low cost means for assuring authenticity and validity of public keys used in large-scale networked services remains both a technological and organizational challenge. In a nutshell, this is the PKI problem, and the papers presented herein propose new solutions and insight for these questions.

This volume holds 16 refereed papers including the presentation paper by the invited speaker P. Landrock. In response to the EuroPKI 2008 call for papers, a total of 37 paper proposals were received. All submissions underwent a thorough blind review by at least three PC members, resulting in a careful selection and revision of the accepted papers. The authors came from 10 countries: Belgium, Brazil, Canada, Finland, Germany, Japan, Malaysia, Norway, Spain, and the USA. The accepted papers were organized into the topical sessions: Invited Talk, Certificates, Authentication, Practice, Signatures, Analysis, and Networks.

The use and exploitation of large-scale public key infrastructures have arrived at a slower tempo and perhaps in other directions than originally envisioned a decade ago. A case in point, only 2 out of 16 authors in this workshop found it convenient to provide a digital signature on the copyright transfer form for the submitted paper. So we are not there yet!

We thank all the people who contributed to this workshop: the authors, the invited speaker, the members of the Program Committee, the members of the Organization Committee, the staff at Springer, the sponsors for their support, and finally all the workshop participants. They all made this workshop successful.

June 2008

Stig F. Mjølsnes
Sjouke Mauw
Sokratis Katsikas

Organization

EuroPKI 2008 was hosted by NTNU.

Chairs

General Chair	Stig F. Mjølsnes, Norwegian University of Science and Technology (Norway)
Program Chairs	Sjouke Mauw, University of Luxembourg (Luxembourg)
	Sokratis Katsikas, University of Piraeus (Greece)
Organizing Chair	Martin Eian, Norwegian University of Science and Technology (Norway)

Program Committee

S. Mjølsnes, Norwegian University of Science and Technology, Norway
Y. Mu, University of Wollongong, Australia
E. Okamoto, Tsukuba University, Japan
R. Oppliger, eSECURITY Technologies, Switzerland
T. Pedersen, Cryptomathic, Denmark
G. Pernul, University of Regensburg, Germany
D. Polemi, University of Piraeus, Greece
B. Preneel, Katholieke University Leuven, Belgium
S. Radomirović, University of Luxembourg, Luxembourg
M. Roe, Microsoft, UK
C. Rong, University of Stavanger, Norway
K. Sakurai, Kyushu University, Japan
R. Sandhu, University of Texas San Antonio, USA
B. Schoenmakers, Eindhoven University of Technology, The Netherlands
S. Smith, Dartmouth College, USA
Y. Stamatiou, University of Ioannina, Greece
G. Tsudik, University of California Irvine, USA
J. Zhou, Institute for Infocomm Research, Singapore
S. Zhu, Penn State University, USA
C. Xenakis, University of Piraeus, Greece

Organization Committee

Jorunn Sommervold, Peter Herrmann, NTNU
Ingrid Melve, Torgrim Lauritsen, UNINETT

Sponsoring Institutions

- Department of Telematics, NTNU
- Faculty of Information Technology, Mathematics and Electrical Engineering, NTNU
- UNINETT
- NISNET (Information Security Research Network Project supported by The Research Council of Norway)

Table of Contents

New PKI Protocols Using Tamper Resistant Hardware

Peter Landrock

Cryptomathic
Cambridge Science Park
327 Milton Road
Cambridge CB4 0WG, UK
and
Mathematics Institute, Aarhus University, DK

Abstract. Keys are indispensable in secure communication, and can only be adequately protected by tamper resistant hardware. Conversely, once you have tamper resistant devices available, you may as well try to make the most of them. In this exposition, we present three novel ideas for the combination of PKI and appropriate use of tamper resistant devices for other purposes than traditional key management. The first is Electronic Negotiable Instruments, such as cash, but there are other important examples, such as Bills of Lading and endorsable checks. The challenge is to prevent double spending without the use of a central registry to keep track of ownership. The second is a new light way digital signature scheme which seems to work well with tamper resistant hardware, but not in software, where it can be broken. We briefly discuss how to make use of this in a transparent PKI solution to be employed by vehicles, which appears to be a hot research topic. Finally we introduce the concept of a digital signature server with central storage of user keys and a central signing facility only, which is operated at the full control of the user using a secure channel for proper authentication. The first and last scenarios have both been deployed in live systems and have been patented, in contrast to the light weight digital signature.

Keywords: Practical PKI, HSM, Light weight digital signature.

1 Introduction

Cryptographic techniques are not just of academic interest, although I have to admit it was because of the academic challenge I first got interested. But through my work with this field, I gradually developed at least partly from an academic researcher into a committed engineer, and this presentation aims to reflect some of the work I have been involved in with my engineering hat on. The main motivation has been practical challenges rather than existing theoretical research.

Of course, there is only one way to protect messages transmitted through insecure networks in almost any sense of the word "protect": By the use of cryptographic and/or error coding techniques. For this we need various algorithms, mechanisms and protocols, and common to all of them are that we need some secret parameters as well, typically called keys.

Fine, so we need to protect the secrets adequately. This can be done either by introducing procedures, e.g. by using key custodians to key in components of a key, or better, by using tamper resistant hardware, or a mixture of both. And once we have tamper resistant hardware for the protection of keys, it would be even better to build solutions where the key would actually never leave the protected environment of the tamper resistant hardware.

So, a few steps into the discussion of how to protect messages, it is already clear that without tamper resistant hardware, it would be a challenge except for the odd user with a trusted, isolated environment. But, if we need tamper resistant hardware anyway, why not try to make the most of it?

The purpose of this discussion is to focus on other aspects of the use of tamper resistant hardware than the obvious one, key management. What more can be achieved using these tamper resistant HSMs, Hardware Security Modules, as they are called? As the space is limited, we have decided to concentrate on three different scenarios, which are all PKI solutions, but where tamper resistant hardware allows for the introduction of novel approaches to address a practical challenge.

In security there is always a cost factor to take into account. We could just drizzle HSM's all over, building very secure, but at the same time prohibitively expensive solutions. Thus we want to include as few HSM's as we really need – but not fewer than that!

2 A Bit of History

As far as I am aware, IBM was the first company to realize and commercially exploit the importance of HSM's for other purposes than key management. When IBM introduced PKI into their Common Cryptographic Architecture in the early 90'ies, they operated with four different classes of public key pairs: Device keys, Signature keys, Encryption Keys and Certification Keys. The main reason it did not sell that well as a concept was that it was basically for closed groups, where everybody would have IBM hardware. The main point here is that it was designed to meet a particular business demand. This was also the motivation for Microsoft to design Palladium as well as the follow-up, NGSCB, and likewise with the TPM architecture adapted by a number of hardware vendors, one purpose of which is to introduce device key-pairs for proper identification of hardware. See [13] for more details.

As an illustration a digital signature cannot be forged by malicious software with access to data that can be signed unless this software also has access to a particular private key. This is contrary to classical protection mechanisms such as a log on a local machine which in principle may be forged, and this makes a fundamental difference.

The protection we need here is protection against adversaries with the ability to modify vital parts of the software. When we state that a device must be trusted to do or not to do something, we mean that we rely on that the software and hardware of the device ensures that the device provides the intended behavior and only that.

We begin with a short description of the background of the three solutions.

3 Three Scenarios

Scenario 1: Electronic Negotiable Instruments
"Negotiable Instruments" is the standard legal term for object that basically can be traded freely such as bank notes, i.e. cash, as the most obvious example. Other examples include endorable checks and Bills of Lading, which is a document being issued when - typically - a ship is loaded with a cargo. The Bill of Lading (B/L) represents the cargo, and whilst the ship travels from one continent to the other, the B/L may be traded many times over, and is needed to be presented finally to take hold of the cargo. (Strictly speaking a B/L is only a quasi-negotiable instrument, but we will spare the reader of this subtlety in the present discussion, as this is of legal implications only.

Challenge: prevent double spending (i.e. trading the same electronic negotiable instrument fraudulently several times without the use of a central registry to keep track of ownership.

Scenario 2: Light weight PKI in Vehicular Communication
It is envisaged that in the near future, automobiles will be able to communicate electronically for the sake of improved safety. It seems inevitable that a number of scenarios will demand some kind of transparent PKI, i.e. a PKI solution where the users are devices, in principle beyond the reach of the persons that actually drive the vehicles. This in particular implies that these devices must be tamper resistant, which gives a number of advantages and challenges. Due to the speed and communication band width requirements, traditional PKI based on e.g. RSA and EC seem too complex.

Challenge: Develop new light weight protocols and use tamper resistant modules to prevent these from being too weak.

Scenario 3: Mobility in Digital Signature Generation
Use case: A customer wants to log on from any work station and carry out his transactions in a secure manner.

Challenge: provide a central server which generates digital signatures on behalf of the owner of the private key at the entire control of the owner with no possibility for e.g. operators to log on as the rightful owner in such a way that e.g. legislation on non-repudiation can be satisfied.

4 1st Scenario: Electronic Negotiable Documents

For the sake of this discussion, en Electronic Negotiable Document (END) is an electronic version of some Negotiable Instrument, i.e. something that can be traded and has a (well-defined) value. As mentioned above, examples include cash of course as an obvious example, but another important example as mentioned is Bills of Lading, which is what we will focus on here, although we will stick to the term END for the electronic version of a B/L.

The average value of a Bill of lading is well over US $ 50,000 these days, so there is something a stake. When a ship is loaded with a cargo, the shipper issues a Bill of Lading (B/L). It could be a B/L for each container, or it could just be one on 5 million tons of oil. This B/L may then be traded (e.g. 20 times over) whilst the ship aims for its destination on another continent. The B/L must be presented by the owner to take possession over the cargo.

In 1995, we built a large pilot for electronic B/Ls in the BOLERO project, sponsored by the European Commission. This was later commercialized by SWIFT and others (see [2]). One of the challenges is obviously to provide a solution which doesn't allow an owner of a an END to sell the versy same END to different buyers. The solution for preventing double-spending in BOLERO was to have a central registry, which at all time keeps track of the current ownership of a particular B/L.

In particular this requires an online registry, which basically is aware of all trading of all B/Ls, and perhaps not always that attractive from a business point of view, as that register becomes very powerful. A more attractive solution would be to give each trader a tamper resistant B/L Token, with an appropriate functionality, which prevents double spending, and then allow trade to move on as it used to do in the past. Let us briefly sketch how this could be - and has been - achieved. This was first described in [5], which was never published in some proper form.

4.1 The Basic Idea

The basic property any electronic negotiable document (END) must have is that of *uniqueness*: By this we do not mean that the document as such cannot be copied. If this were so, it could not be exchanged through an unprotected network. Rather it means that *nobody, but the (rightful) owner*, is able to make any use of the document.

The only other way to provide uniqueness is to physically prohibit free copying. This would involve tamper resistance to realize a protected communication with restricted functionality, if possible.

We thought long and hard for alternative solutions, but in the end the only solution that seemed to work to achieve uniqueness here was quite simple: A message encrypted under a key known to only one entity is indeed unique, as long as it is encrypted, and establishes indisputable ownership by the mere fact that it will only be useful to (the tamper resistant device of) the owner of the key! Only the person in possession of the right key can make any use of the document, which in effect is the property of uniqueness. Only problem: You can't really make use of it as long as it remains encrypted!

The only way the rightful owner can verify that the right END has been encrypted by his key is by decrypting it. But this will give him access to the message and he might then subsequently be able to "sell" it to two different persons by encrypting it simultaneously with their respective keys. Hence this must be prevented. This requires tamper resistant hardware, perhaps a chipcard, or a hardware protected PC. In the following this hardware will be called the B/L Token.

The functionality we require here from the B/L Token is that once the owner has agreed to sell a particular B/L to another registered user, they both have to enter a protocol which allows the B/L to be exported from his B/L Token to the B/L Token of the buyer, but he cannot subsequently export it again, except if at a later point in time he has imported it as it has been repurchased.

An END is generated electronically, for instance by using non-repudiation of origin, in a B/L Token. It is essential that the signature thus appended to provide the non-repudiation never appears in the clear. It will of course suffice to represent the END by a hash value inside the B/L Token.

An END is transferred from one B/L Token to another, through a public unprotected network, in such a way that

a) It can only be transferred to one particular B/L Token to change ownership.
b) Recovery is possible, if the transfer is unsuccessful.
c) The protocol cannot be executed by any other device than an authorised B/L Token.

Each B/L Token must have a public key pair for signature generation and one for encryption. They do not have to be different e.g. if RSA is being used. But it is paramount that the secret keys are not even known to the owner of the B/L Token. The private keys cannot be exported except for back-up solutions - representing another interesting challenge which will not be addressed in full in this exposition.

Now, the challenge is to ensure that B/Ls can only be released (through the trading process) from one genuine B/L Token to another. So the initial question is, how can one B/L Token identify another?

This works well with traditional PKI. A Certification Authority CA, authorises all B/L Tokens in the following manner: The public key of the CA is installed on the B/L Token, and cannot be replaced except through an appropriate protocol. (This will not be further addressed here, but would be completely standard). Moreover, when an END, or rather the defining signature of an END is entered into the B/L Token, the B/L Token must ensure that it can only be released again encrypted for an alternative B/L Token. The point behind this is that it will prevent the use on a non-authorised B/L Token to get access to the vital signature which defines the particular END. In particular this encrypted message is useless except when imported in a certificated B/L Token holding the corresponding decryption key.

Furthermore, and this is an essential assumption, such an encryption of a particular END on an individual B/L Token can take place only once, or rather, once a public key has been selected for export, it is impossible at any later stage to go through the same procedure with another public key and the exact same END, unless of course said END has been properly imported through another sales negotiation at a later point in time.

4.2 The Required Functionality

Here are the basic concepts we need to appreciate in order to design the protocols.

- Creation of a B/L: The B/L is generated on a work station, and the hash is forwarded for the B/L Token. The signature may then be generated by the B/L Token at the will the owner of the B/L Token. Again, once generated, the signature will NEVER appear in the clear outside a B/L Token
- Ownership: If a B/L signature has been imported to a B/L token, it is always marked with a flag which indicates whether it may be negotiated or whether it has been sold.
- Export of a B/L: If the owner of a B/L has ownership of a negotiable B/L, he may choose to export (= sell) it to a business partner - the buyer - who in turn

is the owner of a registered B/L Token as well. To do that, the seller imports the public encryption key of the B/L Token of the buyer simply be receiving the certificate and verifying it. Selling is achieved by encrypting the defining signature under the public encryption key of the buyer and then forwarding it, say over the Internet. It is then marked as "sold" by his B/L Token.

- Import of a B/L: The buyer imports the encrypted signature and verifies the content. If he accepts, it is imported in the sense that the signature is stored by the B/L Token and marked by the B/L Token as "negotiable". If he does not accept, which is the exception rather than the rule, as that as a scenario that will only happen if something went wrong, an off-line TTP has to be involved to reset the B/L as "negotiable" in the B/L token of the seller.

Consequently, the B/L Token must provide the following essential functionality:

a) Creation of an electronic B/L. The content of the B/L is generated in a normal office environment, and once generated, a hash is imported by the generating B/L Token. The B/L Token will add appropriate data for identification and the prevention of double-spending (selling the same B/L to two different buyers) and then sign the concatenation of this and the hash. We will consider this in further details later. It may be appropriate only to allow certain B/L Tokens to have this creation facility for commercial reasons.
b) Exporting (= selling) a B/L. The actual negotiation with a buyer is done out of band in the same fashion as with paper B/Ls. But once agreed to trade,
   1. The certificate of the public encryption key of the B/L Token B of the Buyer is forwarded to the B/L Token A of the Seller.
   2. A verifies the certificate of B, and if this is successful,
   3. The owner of the B/L Token specifies which B/L he wants to export. He can only choose B/Ls which are exportable.
   4. A encrypts the data identifying the electronic B/L, incl. defining signature with the public encryption key of B and forwards this to B. The B/L is the listed by A as exported, i.e. non-negotiable.

Notice all this only makes sense with an application that actually forwards and receives the actual B/Ls with the full content and not only the defining encrypted data.

c) Importing (= buying) a B/L. This matches "exporting" above in the obvious manner:
   1. B decrypts with its corresponding secret key
   2. B verifies the signature and reports the findings to the owner.
   3. The owner accepts.
   4. The B/L is accepted by B and flagged as negotiable.

The Buyer is now the owner of the B/L.

4.3 Additional Properties

We have mentioned the possibility of re-importing an END which has previously been exported from a B/L Token. This can be achieved as follows:

A counter will be incremented for each completed negotiation in such a way that a B/L Token may re-purchase any END at a later point in time, without violating ii). This counter will allow the B/L Token to distinguish between the END previously sold and therefore flagged as non-negotiable from the same END later received again through a proper negotiation-phase and then flagged as negotiable.

Back-up: Each B/L Token may have a back-up B/L Token. Alternatively, a record may be kept of all purchased END's (i.e. encrypted under the public key of the Buyer) by the owner of the Buyer or by the TTP, if this is involved in tracing, and the TTP may have a back-up copy of the secret key. If an owner announces a particular END for lost, e.g. because his B/L Token has broken down, the TTP waits until the expiration of the validity period. If the END has not been claimed by someone else by then, the TTP will handle this by the back-up copy of the secret key of the Buyer. Alternatively, which is the most realistic commercial solution, the Owner may recover immediately by giving a guarantee or similar.

Further Mechanisms and Protocols

a) Generation of an END

As discussed above, the content of an electronic negotiable document is generated freely in an unprotected environment with a time stamp indicating the time of issue. A hash value of a fixed bit length of this is given as input to a B/L Token. The B/L Token automatically, at the initiation of the user, adds

1) Its device number, which is in one-to-one correspondence with the public key of the device used for verification of digital signatures generated by the device. This could be reflected in the certificate, or the device number could be the upper bits of the public key, after the most significant 1.
2) A serial number which identifies the END uniquely in the system.

This is all beyond the control of the user.

The concatenation of all of this is then signed by the signature key of the B/L Token. The numbers appended by the device are public and appears as output from the B/L Token, after which they are appended to the plaintext content of the END as well. This information and the digital signature are stored on the B/L Token for later negotiation in its own record. To this information is further appended the number 0, called the counter, which indicates that the END has been negotiated 0 times, and a flag which indicates that the END may be negotiated.

b) Negotiation of an END

A negotiation involves a seller (Sl) and a buyer (Br). It may involve a TTP as well for passive logging only for recovery purposes.

1. Sl decides to sell a certain END, which is stored on his B/L Token (either because it has been issued on the B/L Token, or because it has previously been purchased by the B/L Token) to a certain Br.
2. Sl requires the certificate of the encryption key of the B/L Token, which is forwarded through a public channel to the B/L Token of the Sl.
3. The B/L Token of the Sl verifies the validity of the certificate and extracts the public key of the B/L Token of the Br. The Sl specifies the serial number of the

END to be negotiated. If the flag of the End indicates that the END is non-negotiable (i.e. has already been negotiated), further action is denied. Otherwise, step 4 is carried out.

4. The full record in Sl's B/L Token of the END is encrypted by means of the public key of the B/L Token of the Br (possibly using a hybrid system) and transmitted, together with the content of the END through a public channel to Br.
5. The status of the record (i.e. the END and the counter) is flagged non-negotiable.
6. The B/L Token of the Br decrypts the received information with its secret key.
7. The B/L Token of the Br requests the certificate of the Issuer, the hash value of the original content of the END and verifies the signature and the device number of the Issuer. If accepted, an acknowledgement such as a digital signature on the concatenation of the serial number of the END, the generating signature and the counter is returned to the B/L Token of the Sl, together with the certificate of the B/L Token of the Br. A copy may be returned to the Issuer as well for tracing.
8. The B/L Token of the Sl verifies the acknowledgement and outputs the result for information to the Sl. (Likewise at the Issuer if applicable.)
9. The received information, i.e.
   1) the hash value of the content of the END
   2) the device number of the generating END
   3) the serial number of the END
   4) the generating signature
   5) the counter, subsequently incremented by 1

A flag will indicate that this document, with this particular counter, is now negotiable.

c) Recovery
Various back-up procedures are possible as mentioned earlier. In one solution, the TTP will have a copy of the appropriate keys protected by a key only known to the owner of the B/L Token. As the Buyer always received the END's encrypted under the public key of his device, he may keep copies of the received encrypted information for later recovery my means of the TTP.

Alternatively, each B/L Token could have a back-up identical device with identical public key pairs, certificates and device number, which furthermore is identified as a back-up device. Whenever a negotiation takes place, the protocol is duplicated with the back-up B/L Token. If the primary B/L Token breaks down, the back-up may sell to the issuer after expiration time of the END's which could not be negotiated, or special procedures could take over.

4. Splitting a B/L
This is a functionality that had previously not been available in the paper based world.

In the application where each B/L Token is an issuer, a END may be split by any Seller into two (or more for that matter if this is meaningful) END's, where the sum of the values of the new END's is that of the old, e.g. if the cargo is oil. This is secured by the B/L Token. The Seller's B/L Token then signs each of the End's, which basically is the original signature together with a new value, and forwards its certificate in the trading protocol. The B/L Token of the Buyer verifies the original signature of the Issuer as well as the new signature.

5. Settlement
In the end, the cargo is collected, and the B/L is finally returned to the issuer with the payment. This is again handed out of band.

If the END is digital cash or an electronic cheque, perhaps with the electronic watermark of the bank, the bank will cash in the END and mark the END non-negotiable. Thus the settlement follows the same principle as an ordinary negotiation.

In 1994 we filed a world wide patent [10] on these fundamental protocols, which has now been granted almost everywhere with the exception of USA, where the difference of this approach and the BOLERO approach with a central registry still appears to be incomprehensible to the patent reviewer. The first live world-wide solution has now been employed by e-Title Authority Pte Ltd.[3], and we wish them good luck.

5 2nd Scenario: Light Weight PKI in Vehicle Communication

This is more speculative and has not been deployed yet. Nor has it been patented.

It is now a well established fact that information technology is the driving force behind a number of innovations in the automobile industry, and a number of papers have analysed the various threats and security measures that may become relevant, see for instance [1], [4] and [12].

A major challenge is to avoid providing so much functionality that the security requirements become intolerably complicated and expensive. Of course, it will not really be possible to design the appropriate and optimal secure architecture until we know exactly which functionality we need from the system - something researchers often forget. Until then, the object we want to secure, the automobile, easily becomes a moving target (!).

It is pretty obvious that once we start building solutions, there will be a number of fundamental, difficult challenges such as timeliness, robustness and scalability: To have whichever information is required at any time at any party available, regardless of the density of the traffic, and highly trustworthy at the same time.

The ideas presented here go back to [8]. Before we can even start thinking about the appropriate security architecture, there are a number of generic challenges we will have to address, such as

1) How can we take the maximal advantage of the fact that each vehicle will have a tamper resistant device installed? For instance, will that enable us to come up with light weight authentication mechanisms, which are an order of magnitude faster to generate and verify than the standard ones – even considerably faster than elliptic curve digital signatures?
2) How do we provide an appropriate PKI solution with as little overhead as possible? E.g. can we avoid revocation lists – which is the nightmare of most PKI solutions and if so how?

5.1 A Lightweight Digital Signature

We start by recalling a new proposal for a lightweight Digital Signature we presented some years ago in [7] – which alas can be broken if implemented in software but at a speed which is an order of magnitude slower than it takes to generate or verify it and then move on to explain how to make use of it. We will then discuss how to enhance this using tamper resistant hardware.

In the following, "$a \approx b$" mod k means a and b have the same residue modulo k. The following observations can be found in greater detail in [6]:

P. Fermat observed that if p is a prime, then

(1) $p \approx 1$ mod 4 iff there exist a, b with $p = a^2+b^2$

i.e. p factors to the primes $(a+ib)(a-ib)$ in the Gaussian ring **Z**[i] (which happens to be a unique factorization domain).

An equivalent, although not entirely obvious, statement is that

(2) $p \approx 1$ mod 4 iff -1 is a square root modulo p

Now, let $n = pq$ be an RSA modulus, where p and q are primes which both are 1 mod 4.

Then the multiplicative subgroup of **Z**/n**Z** has exactly 4 square roots of -1, say +/− δ and +/− ε, where say

(4) $\delta \approx \varepsilon$ mod p and $-\varepsilon$ mod q

Then obviously $\gcd(\delta\varepsilon+1, n) = p$, i.e. knowing δ and ε is equivalent to knowing p and q. However, as argued in [6], knowing only one of them and then calculating the other from that is likely as hard as factoring.

Assumptions
Let $n = pq$ be an RSA modulus, where p and q are primes which both are 1 mod 4.

Prover knows a square root γ of -1 mod n
Verifier knows Prover's public RSA key

Signature generation
Let m be a message (or the hash of another message) with $m < n$.
Choose a random and calculate b to satisfy $\gamma ab = m$ mod n.

Set $r := (a+\gamma b)/2$, $s := (\gamma a+b)/2$

Then (r,s) is called the γ-signature of m with respect to a.

Signature verification
The verifier V receives m, r and s, and verifies that

$r^2+s^2 = (a^2-b^2+2\gamma ab-a^2+b^2+2\gamma ab)/4 = \gamma ab \bmod n = m$

Notice here that if a is revealed, too, then calculating rs reveals γ:

$4rs = (a+\gamma b)(\gamma a+b)/4 = \gamma(a^2+b^2)$, as we know a^2 as well as b^2 (from γb).

It is thus essential that a is chosen at random and kept secret.

So why does this look like a good idea? For m a message, let the pair (r,s) be the digital signature. All that is required to calculate it is about 2 modular exponentiations, and verification is equally easy, if RSA with small exponents are being used. So indeed, we have come up with a scheme that is dramatically better than any other known scheme!

Unfortunately, as explained in [7] as well, it can be broken:

Find a number x such that $y = x^2m \bmod n$ is a prime which is 1 mod 4. This is relatively easy using say the Rabin primality test.

Use Cornacchia's algorithm (see [8]) to find a, b with $y = a^2+b^2$

Then $m = (x^{-1}a)^2+(x^{-1}b)^2 \bmod n$, and we have broken the scheme. In fact in doing so, we discovered that you do not even need the assumption that n be the product of two primes equivalent to 1 mod 4 to write m as a sum of two squares modulo n.

We notice though that breaking it is considerably more time demanding than generating it in a legitimate way in that it requires at a minimum a modular exponentiation +, compared to a couple of modular squarings.

So doesn't that kill our proposal for a light weight digital signature? Not necessarily so! That depends entirely on the details of the architecture and this is where HSMs come in handy. Just as an example, assume the following holds:

1. The signature is always generated by a tamper resistant device which has been certified for use in a system. Such a device will come with a very limited set of commands, and it cannot include any other means of generating a light weight signature.
2. Vital parts of the communication is encrypted or otherwise protected to prevent outside attacks exploiting the attack described overleaf.
3. A certificate has been generated by a recognised CA for the system on the corresponding public key.

Under the assumption that all communicating unit are tamper resistant and authorised and certified with the alleged functionality, any falsified digital signature obviously must originate from a third party. Whether that will be a realistic threat depends on the nature of the communication, but we should of course try to minimise the probability that this can happen at all. One possibility, just for the sake of the argument, is to encrypt all messages. In order to avoid overhead, this would basically require that communicating units already share a key, but even though this would be possible, the security of the whole system would rest with the protection of that key, which perhaps would be considered unduly risky.

A more sophisticated approach is the following example:
One recipient B of a message signed as proposed above by A, and to whom the message is deemed critical by some application, could return a randomly chosen challenge c RSA-signed by B encrypted under the public key of the alleged transmitter A. The RSA signature by B could easily be pre-calculated, even with a timestamp which is fairly accurate, and encryption under A's public key is cheap for RSA with small exponents. A verifies the RSA signature (which proves that c was chosen by B, hence is random, as B is tamper resistant). By choosing c smaller than the smallest prime

divisor of the public key of A (e.g. say 500 bits, where all primes used are around 512 bits), decryption at A is faster by a factor 2 – 4 compared to e.g. an RSA signature generation, depending on implementation, as A need only decrypt modulo the smallest prime divisor. So with one initial RSA message from B, A can now authenticate itself to B using light weight signatures as follows.

Let $a(m) = a$ denote the random number chosen above along with the message m. A then chooses $a(m) = c$, which he just received from B, and calculates $b = b(m)$ from $\gamma cb = m \bmod n$, to return the signature $r := (c+\gamma b)/2$, $s := (\gamma c+b)/2$. B can now verify that indeed the signature was calculated using a square root of – 1 modulo the modulus of the public key of A, *which rules out the attack described above*. Of course this means that B will learn of the secret of A, but as it is a tamper resistant device, it can make no use of it and dispose of it if programmed so. At the same time, any other recipient can verify that the message m has been signed by means of a light weight signature, but of course cannot rule out the attack. For a subsequent message, m_1, A could use $a(m_1) := h(c)$, where h is a known hash function, and so on, so for all subsequent signed messages, B can verify that A indeed did sign using a square root of -1 modulo its public key modulus.

Notice that if a number of vehicles are communicating, B could now confirm to all other parties that it has verified the signature as genuine with its own light weight signature on said message m, which indicate that either the original signature received from A is genuine, or A and B are both allied with a malicious third party. We leave it to the reader to generalise this to a scheme where a small, but reassuring number of vehicles authenticate each others messages for the benefit of all other communicating vehicles.

Notice that it will still be possible for other applications – if need be – to generate ordinary RSA signatures on the same device.

So far for the basic protocol. But we need a PKI.

5.2 Transparent PKI for Vehicle Communication

Before we discuss this further, we want to recall that there are a number of large scale transparent PKI solutions out there: In EMV (debit- and creditcards), TPM (Trusted Platform Module) and DRM (Digital Rights Management), and on the whole they work very well. What we propose here is yet another transparent PKI protocol.

Avoiding Revocation Lists

One of the most challenging problems has been identified as that of how to possibly avoid revocation lists (see e.g. [2] and [5]).

First of all, it is reasonable to ask: Under which conditions should the certificate of a particular vehicle be revoked? We can think of dozens of reasons why a driver should be banned from driving, and we can also think of reasons why a particular vehicle should be banned from getting on the road (e.g. as it failed its MOT test because of a serious defect). But if the vehicle contains a tamper resistant unit which only transmit automated messages, we can only think of a couple of reasons why the corresponding private key should be revoked: Sufficient suspicion that the private key has been compromised in spite of the tamper resistance, or that the device has successfully been removed from its vehicle, or – in the scenario where lightweight signatures are being used, that the vehicle is known to collude with a malicious third party.

It could thus be quite unlikely that we need to revoke a key pair. Nevertheless, if we feel strongly about this, here is a way forward that would still avoid revocation lists: Basically, the idea is the following: The CA only issues very short term certificates, say every hour or even every 15 minutes depending on capacity, just for the sake of the argument. This would automatically revoke any certificate much faster than if revocation lists were to be used. If the vehicle is in a remote area it does not matter, this is not were the technology will be of any use worth mentioning. On highways and in densely populated areas, it would be easy to have the certificate renewed in an automated fashion once a base station is passed, and we shall make a more detailed proposal below. Obviously, if a vehicle key pair needs to be revoked, it just needs to be communicated to the appropriate CA not to reissue a new certificate at the next automated request. The impact of this will be quite dramatic, as suddenly certificates become very trustworthy – which is very important for the proposal above on light weight signatures.

Renewing a certificate: As mentioned earlier, electronic communication between automobiles is much more important in densely populated areas. At the leisure of the appropriate authorities, there could be base renewal stations at regular intervals in sufficiently densely populated areas that carries out the renewal in a completely automated manner as the vehicle passes. In addition, we could imagine such a unit installed in every police car as an option. Once a vehicle passes within communication range of such a renewal station, it automatically submits a request for renewal, and the CA for the region is contacted for an online generation of a certificate provided the existing cer has not been revoked. So there would be revocation lists, but not at the end user level, only at the regional CA level, a bit like in the EMV world.

So each region has a CA, and when a vehicle is in that region, this is the only CA that matters. Of course the regional CA needs to be aware of basically all other CAs in the world, e.g. through certificate paths just as in the EMV environment, as a vehicle could arrive e.g. by ship from another country, but once it has been verified that the certificate should be renewed, it might as well be with the private key of the local CA.

6 3rd Scenario: Mobility in Digital Signatures

Whereas digital signatures really are designed - and mostly deployed - for automated processes, there is one exception, electronic commerce, which is often conducted from an insecure – or at least untrusted – workstation and gives a lot of headaches, not least legal ones.

To introduce a device that the user can carry around - like a chipcard or a USB token - is a very appealing way of protecting the secret key, but in the case of the chipcard, one additionally needs a connected chipcard reader, and in both scenarios there are further challenges with this approach.

For an electronic message which an individual is about to sign, he is forced to believe that whatever he sees is a true representation, or "transcription" of what his is digitally signing. He normally has no means of verifying, but has to rely on the interpretation of the codes, he himself typically cannot read or comprehend.

The functionality he needs is what we introduced in [9] and named WYSIWYS, “What You See Is What You Sign”. The real issue of course is whether the hash to be signed really is calculated on the message displayed to the user on the screen prior to signature generation.

6.1 Protection of Signing Keys

Apart from WYSIWYS the other paramount issue in electronic commerce or banking - namely proper key protection – touches a vulnerable point of traditional digital signature solutions: From a security perspective the best solution is to keep the signing key in protected hardware of some kind as discussed above. But usability considerations have promoted software-based solutions, and today these are by far the most widespread for cost as well as convenience.

Thus the traditional solution is a compromise between security and usability, and in reality a software solution ties the users' access to e.g. a particular Net-banking application to one specific work station. But mobility is an absolute requirement of most Net-bank users.

Since the weakness is storage and handling of the signing key, the straightforward solution is to free the user from this responsibility. Instead the keys are generated and stored centrally on a secure server enhanced by appropriate HSMs - preferably handled by an independent third party - and the signature key never leaves this protected environment, yet it is controlled entirely by the user through secure channels at all times. This has some evident advantages:

- Optimum physical security: The key is far better protected on a secure server than on the user's hard disk or on a smart card.
- Protection against theft: It is practically unfeasible to steal a key from a secure server – even for the system administrator.
- Mobility: The owner can use the key from any terminal with Internet access.
- Logging: Access and use of the key is logged which makes it easy to detect misuse.

6.2 Two Factor Authentication

With the approach described above, the security challenge has now been reduced to protection of key usage, and again this new approach has superior functionality. For software solutions, the key is only protected by a (static) password on the user's hard disk. On a signature server the security level is easily increased with a well-known technique: authentication through two independent channels – typically something you *know* and something you *possess*. In Net-banking a natural solution is a (static) password combined with at one-time password, which can be delivered as an SMS to the user's mobile phone, printed on a card, generated by a token, etc. To access the key the user has to prove that he (1) knows his password and (2) is in possession of his phone/card/token.

So to handle this one server, the signature server, stores the private keys on behalf of all users and at their full control, while another, the authentication server, is responsible for proper authentication of some sort do the users. These two servers are connected through a secure channel, and a re best operated by independent parties, although for less secure solutions , the two server functionalities may be handled by one ubiquitous server only.

6.3 Signing

Signing a transaction then looks as follows:

1. The user logs on to the Net bank or the application.
2. The user wishes to perform a transaction, which requires a digital signature. The static password and a hash of the transaction, required to generate the signature, is sent by the authentication server to the signature server through the secure tunnel.
3. The user has received/generated the one-time password controlled and verified by the authentication server.
4. The user sends the one-time password to the signature server via the secure tunnel.
5. The server verifies that the information received from the user and the authentication server match and if so returns the signature on the transaction (and a certificate if required).
6. For particularly sensitive or valuable data, the content of the signed message may even be forwarded to the user through a different channel, e.g. on a mobile phone or a pda, and the user confirms the content via that channel. This is one simple but very efficient way of addressing the WYSIWYS issue, which completely thwarts MITM attacks.
7. The transaction, signature, (and certificate) are sent to the Net bank or appropriate application.
8. The Net bank or appropriate application validates and processes the transaction.

Of course there are lots of details to add before the solution is sufficiently secure to offer non-repudiation in the legal sense. Again, we have filled several patent applications to protect our technology, and a European patent was obtained in 2006 [11], and a number of solutions have already been deployed throughout Europe.

7 Summary

We have presented two solutions from the real world (first and third scenario) where the challenge was to combine PKI with the use of tamper resistant hardware not only to protect keys, but to enforce a certain and limited functionality that enables the special requirements that were imposed by the system architecture. We have moreover presented what we believe is a viable and practical way forward for the use of an innovative new PKI system for communication between moving vehicles which employs tamper resistant hardware that would enable a new light weight digital signature we have proposed. This is based on RSA but is an order of magnitude faster than RSA in that both signature generation and verification is a few exponential modular squarings. As PKI in vehicle communication is still something of the future, this has not been implemented yet, in contrast to the two other scenarios.

Note: This paper was produced to explain in greater details what was presented by the author as the invited keynote speaker at EuroPKI 2008. The author is grateful to the referees for their feedback which helped making the paper more accessible.

References

1. Blum, J., Eskandarian, A.: The threat of intelligent collisions. IT Professional 6(1), 24–29 (2004)
2. Bolero's Trusted Platform: http://www.bolero.net/solutions/tradeplatform/titleregistry.html
3. E-Title, http://www.e-title.net/
4. Hubaux, J.-P., Capkun, S., Luo, J.: The security and privacy of smart vehicles. IEEE Security and Privacy Magazine 2(3), 49–55 (2004)
5. Landrock, P.: A New Realisation of Negotiable instruments. In: Cambridge Workshop on Security Protocols, April 1996, The Isaac Newton Institute, Cambridge (1996)
6. Landrock, P.: A new concept in protocols: verifiable computational delegation. In: Christianson, B., Crispo, B., Harbison, W.S., Roe, M. (eds.) Security Protocols 1998. LNCS, vol. 1550, pp. 137–145. Springer, Heidelberg (1999)
7. Landrock, P.: Security Protocols – Who Knows What Exactly. In: Herbert, A., Jones, K.S. (eds.) Computer Systems: theory, technology, and applications, Monographs in Computer Science, Springer, Heidelberg (2003)
8. Landrock, P.: Timely Authentication in Vehicular Communication, presentation at ESCAR, Berlin (2006)
9. Landrock, P., Pedersen, T.P.: WYSIWYS? What you see is what you sign? Information Security Technical Report, vol. 3(2). Elsevier, Amsterdam (1998)
10. Patent No. WO9624997, Electronic negotiable Documents, Inventor: Landrock, P. (first filed 1994)
11. Patent no. EP13645081 B., Data Certification Method and Apparatus. Inventors: Landrock P. and Tuliani, J. (filed 2002)
12. Raya, M., Papadimitratos, P., Hubaux, J.-P.: Securing Vehicular Communications – Accepted in IEEE Wireless Communications Magazine. [LCA-ARTICLE-2006-015]
13. TPM, https://www.trustedcomputinggroup.org/groups/tpm/

Validation Algorithms for a Secure Internet Routing PKI

David Montana and Mark Reynolds

BBN Technologies
10 Moulton Street, Cambridge, MA 02138 USA
dmontana@bbn.com, mreynold@bbn.com

Abstract. A PKI in support of secure Internet routing was first proposed in [1] and refined in later papers, e.g., [2]. In this "Resource" PKI (RPKI) the resources managed are IP address allocations and Autonomous System number assignments. In a typical PKI the validation problem for each relying party is fairly simple in principle, and is well defined in the standards, e.g. RFC 3280 [3]. The RPKI presents a very different challenge for relying parties with regard to efficient certificate validation. In the RPKI every relying party needs to validate every certificate at fairly frequent intervals (e.g., daily). In addition, certificates on the validation path may be acquired from multiple repositories in an arbitrary order. These dramatic differences motivated us to develop performance-optimized validation algorithms for the RPKI. This paper describes the software developed by BBN for the RPKI, with a special focus on this optimized validation approach.

Keywords: Border Gateway Protocol, Resource PKI, Internet Routing PKI, Route Origination Attestation.

1 Background

The Border Gateway Protocol (BGP) [4] is a critical routing protocol in the Internet. Routers exchange Autonomous System (AS) path information between themselves using BPG UPDATE messages. Unfortunately the current implementation of the BGP protocol does not provide any method for determining if such path information is valid. Path information may be invalid due to configuration errors, or due to malicious BGP spoofing [5, 6].

Several proposed alterations to BGP provide for additional security to the path information [7, 8, 9, 10, 11]. All are predicated upon the existence of some form of PKI that binds AS# and IP-address block resources to the entities to which they have been allocated. These proposals have not been adopted due to the changes required to routers and the infrastructure requirements imposed. The most recent proposal for creating the requisite infrastructure is described in [1], an approach based on a new, digitally signed object, the Route Origination Attestation (ROA), together with a PKI to validate, manage and process such objects. Relying party software for use with this "Resource" PKI (RPKI) was implemented by BBN, and is described in this paper. Of particular interest is the set of validation algorithms that were developed for the RPKI.

In a typical PKI [12] the validation problem for each relying party is fairly simple in concept, although it may be complex in practice. Typically a relying party receives an End Entity (EE) certificate which must be validated prior to verifying the signature on an object. The relying party may be provided with additional Certificate Authority (CA) certificates needed to complete the certificate path to one or more Trust Anchors (TA) employed by that relying party. The validation of a certification path from a TA to a EE certificate, including processing of revocation status data, is well defined and specified in standards. The non-standard part of the process is the discovery of a suitable certificate path.

Given this typical task for a relying party, strategies for optimizing the performance of certificate validation have been developed. They are based on the assumption that a relying party will, within a reasonable time interval (say, 24 hours), validate only a very small fraction of all the certificates issued in the context of the specific PKI. This is a reasonable assumption for most PKI applications, e.g., secure email provided via S-MIME, VPN security using IPsec, or secure web access via TLS/SSL [13].

The RPKI presents a very different challenge for relying parties with regard to certificate validation. In this RPKI it is anticipated that every relying party (e.g., every participating ISP) will need to validate every certificate within (roughly) a 24 hour interval. This dramatic difference in validation methodology motivated the development of a novel performance-optimized approach to certificate validation. This paper focuses on the validation algorithms developed as part of our work on implemented the RPKI. Our approach to validation makes use of a relational database containing only validated signed objects (e.g., certificates) to avoid duplicative validation processing.

2 RPKI Software

The goal of the BBN RPKI (relying party) software is to process data from repositories operated by the five RIRs (and their subordinate certification authorities) in order to create a single set of text files that can be used to generate BGP filters. These filters describe which Autonomous Systems are authorized to originate routes for specified IP address prefixes. The BBN RPKI software system finds all the available ROAs, and their associated certificates and certificate revocation lists (CRLs), verifies the ROAs using these certificates and CRLs, and creates simple text files containing AS#/IP prefix pairs that can be used to create BGP filters.

As noted above, this process is different from the traditional use of a PKI to support applications. Previously, an application received one or a small number of signed objects that required validation. The application would gather the certificates required to form a certificate chain to a Trust Anchor and then perform the appropriate certificate path checks. Our application needs to determine the validity of all the available ROAs, certificates and CRLs across multiple repositories. To do this quickly and with efficient use of system resources requires a different approach than that of PKI software supporting a traditional application, since it will require processing the same certificates and CRLs many times. Therefore, our software extracts relevant information from local copies of certificates and CRLs, and creates relational database records containing that information. This approach provides a simple, efficient, and highly structured way to access the applicable information for each object without having to repeatedly acquire and process certificate and CRL data. We first discusses how the data is stored locally, and then describes the different programs that operate on the data and transform it, eventually resulting in output files that can be used to create BGP filters.

Our software suite consists of a set of programs that typically run continuously, in the background, rather than a set of programs that are executed once from the command line. This is done for the sake of efficiency; as the ROA's validation state changes over time; our software updates files incrementally, so as not to require reprocessing all the data.

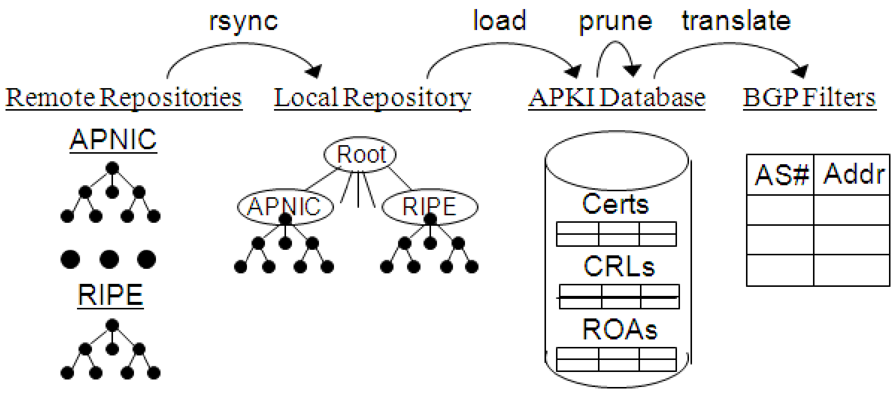


Fig. 1. Transforming the data

The different forms of the data and the transformations between them are shown in Figure 1. We now describe each component in the process.

Remote Repositories - The raw data (certificates, CRLs, and ROAs) are stored as files on a distributed system of servers provided by the CAs that comprise the RPKI. Each of the five RIRs will attempt to cache data from its subordinate CAs on its repository server, so it is expected that our software will be able to find most of the data it needs on the RIR servers. However, there will be some cases where the data is not cached at a RIR and the application will need to retrieve data directly from a subordinate CA repository. Data from these repositories can be located using the Subject Information Access (SIA) extensions present in RPKI certificates [14].

Local Repository - Our application retrieves files from repository servers and caches them locally. It uses rsync, an existing freely available utility, to keep the local repository synchronized with the remote repositories in a directory structure that mirrors the originals. Because most of the files retrieved from the remote repositories will need to be accessed many times as part of ongoing processing, it is more efficient to copy these files once (particularly with the efficient copying of rsync) rather than to continually access the remote versions. Furthermore, rsync outputs a record of what changes occurred in which files, which allows our application to process only the modifications, and thereby minimize the work done with each incremental data update.

Relational Database - MySQL [15] was used as the relational database underpinning our relying party software. The database has three main tables, one for each type of object of interest: certificates, CRLs and ROAs. Each object type has a different set of data fields in addition to its signature. The database includes only those fields required to search for and/or identify the different objects/rows in the tables. (It is easier and more efficient to leave seldom-used information in the corresponding file.) For example, some of the fields included in the certificate table are: subject key identifier (SKI), subject, authority key identifier (AKI), and issuer. This supports an SQL query requesting the certificate with a particular SKI and subject, or a query requesting all certificates with a particular AKI and issuer. (The significance of these queries is discussed in the section on validation algorithms.) Such database queries allow rapid location of objects of interest.

The database does not include all objects. If an object has been determined to be invalid, that object is either not loaded into the database, or is deleted from the database if it had previously been valid or awaiting validation. Note that the underlying file remains in the local repository until its remote copy has been deleted. (Henceforth we will always use the term "repository" to denote the local collection of files, unless otherwise noted.) An object can be determined invalid for a variety of reasons including expiration, revocation or a signature that does not verify. An object will be placed into the database if it is valid, or if there is not yet enough information to determine whether or not it is valid. The most common reason for an inability to determine the validity of an object is when there is a missing link (ancestor) in the certificate path to a Trust Anchor. An object of undetermined validity stays in the database until it expires (and hence is know to be invalid) or it is deleted from its remote repository (and hence also from the local repository).

There is one column in each object table containing the validation state. This field can have three possible values: validated, awaiting-validation, and CRL-stale. Validated and awaiting-validation represent the obvious meanings;

CRL-stale is explained next. Each CRL has a field next-update that tells the latest time to expect an updated version of the CRL, which may enumerate a potentially different set of revoked certificates. When the current time becomes later than the next-update time for a CRL, then any *sibling* certificate of this CRL (i.e., any certificate that shares the same issuer and hence could be revoked, as discussed below) enters an uncertain state. This uncertain validity extends to all its *descendants*, i.e., those certificates and ROAs whose certification path includes this certificate. All sibling certificates and descendants have their validation state set to CRL-stale until the expected update to the CRL arrives.

BGP output files - The goal of the application is to derive this set of files, which are just text files consisting of IP address prefixes and their associated AS numbers. These pairs are specified in the ROAs and our software accumulates the pairs from all validated ROAs. An issue is what to do with those ROAs in the CRL-stale validation state. The application provides the user with three options: include them, exclude them, or put their pairs in a separate table/file where a human can decide how to handle them. (This last option is the default.)

The software is composed of a set of application programs that operate on the data. There is a natural sequential order to these programs which mostly follows the sequence of data transformations shown in Figure 1. Typically, one would expect them to be run in sequence at least once a day. However, there may be times when the programs need to be run separately, which is also possible. The program components are described next.

Synchronizer - This program executes rsync for each of the five top-level RIR repositories in order to create a local repository copy that contains the same data as the remote repositories. The program rsync is an open-source utility that efficiently synchronizes files and directories between two systems. The local repository has at least five parallel subtrees, one for each of the remote repositories. The synchronizer can synchronize with remote repositories other than these five main ones and generate additional subtrees if it is known beforehand (or discovered subsequently) that the data from some RPKI CAs is not cached at one of the five RIRs. The rsync program outputs messages about everything it does, including all files that it has added, updated or removed; the synchronizer saves this output to a log file to tell the loader which files need to be handled.

Loader - This program looks at the logs generated by the synchronizer and loads the data from all new or modified files into the database; it also deletes the data corresponding to those files that have been removed. Before the loader puts a record corresponding to a new object into the database it checks whether this object is invalid and refrains from loading such files. If the object can be validated, it sets the validation flag for that object, and then determines which other objects already in the database need to have their validation state changed, or need to be deleted, as a consequence. This process of validating an object and then determining the effects of this change on other objects in the database is discussed in detail in Section 3. When there are multiple remote repositories, the loader is executed in parallel with the synchronizer. After the synchronizer is done updating from one repository and has proceeded to a second repository, the loader can be loading the data from the first repository. Because both the synchronizer and loader spend much of their time doing input/output, parallel execution provides true speedup.

Pruner - This program looks for changes in the state of objects due to the passage of time, in particular certificates and CRLs that have expired (but which were valid at the time they were loaded). Finding expired objects in the database is easy, requiring only two database queries, one for certificates and one for CRLs. Propagating the effects of an expired certificate follows the process described in Section 3.

Chaser - This program is invoked when not all the certificates are cached at the RIR repositories, and hence accumulating all the required data necessitates retrieving data from lower-level CA repositories. Each certificate has a field that points to the location of its parent (the Authority Information Access or AIA), one pointing to sibling certificates (the Subject Information Access or SIA), and one pointing to the CRL in which the certificate will be listed if revoked (the CRL Distribution Points or CRLDP). The chaser accumulates a list of alternative repositories from which to retrieve data and executes the synchronizer and loader on these repositories in the same way they are initially executed on the RIR repositories. Currently there is no way to check which objects from these lower-level repositories have already been cached by an RIR. In this case those objects will already have been copied and processed locally. When the duplicates are copied from a lower level CA repository, the fact that it they are duplicates will be noted, and they will not be entered into the database twice.

Translator - This utility creates the output files. The primary content of a ROA is a mapping from IP address prefixes to AS numbers; the translator combines the data from all the validated ROAs into a single large list. Note that the translator is part of a more general query tool for extracting information from the database, which allows easy visibility into the database for the expert user.

BBN has developed, integrated, tested, and released this software as a free, open source project. It may be downloaded from [16].

Work on the software is ongoing. In particular, the evolution of the resource certificate project [17] has grown to encompass a new type of object: a file manifest. BBN is currently working on extensions to the RPKI software to process manifests, as well as integrating them into the overall database architecture. (A manifest is a list signed by an EE representing a CA, enumerating all of the files currently published under that CA. The manifest will be used to detect unauthorized deletion or substitution of (older, valid) files in a repository.) BBN also plans to continue work on improving the efficiency of the software suite. As described below, there are cases in which caching a previously computed validation result can result in a significant performance improvement. In addition, the chaser (and the entire rsync transfer process) can be made more efficient by adding a mechanism for determining which objects in which repositories have already been retrieved, so that duplicate files are transferred less frequently. A number of other performance optimizations are also being considered. Finally, as standards for secure internet routing continue to evolve [18], we anticipate that the RPKI software will continue to be improved to track those changes.

3 Validation Algorithms

The process of keeping the (local) database current with respect to the validation state of all objects (certificates, CRLs, and ROAs) has two main parts. The first is validating or invalidating individual objects. The second is propagating the consequences of an object's change of validation state throughout the database. We discuss each of these steps, in turn, after providing a quick overview of the various objects relationships.

The primary relationship between objects is the parent-child relationship. The parent object is always a certificate. If the child is a certificate or a CRL, the parent is the certificate of the CA that signed the child. For a ROA or a manifest, the parent is an EE certificate. Hence, a parent certificate is required to validate an object. For the signature to be acceptable, the parent must itself be validated, and so a certification path is required back to a Trust Anchor. A Trust Anchor is a self-signed certificate that is inherently trusted; the default Trust Anchors in the RPKI are self-signed certificates issued by the RIRs and by IANA.

The parent-child relationship between two objects is dictated by two simple rules:

– A certificate is the parent of another certificate or of a CRL if the parent's SKI equals the child's AKI and the parent's subject equals the child's issuer.
– A certificate is the parent of a ROA (or a manifest) if the two objects have the same SKI.

The sibling relationship between CRLs and certificates is also important, because a CRL can revoke any certificate that is its sibling, i.e., has the same parent. A certificate is the sibling of a CRL if the two objects have the same AKI. A validated CRL revokes a sibling certificate if the serial number of the certificate is in the list of serial numbers of the CRL.

Having a relational database makes it easy and fast to find objects that satisfy these relationships. For example, the following single SQL query produces all the certificates that are the children of a certificate with SKI=foo and subject=bar:

SELECT id, ski, subject FROM certificates WHERE aki="foo" AND issuer="bar";

We now discuss the different ways that an object can change its state as part of single object validation. These are changes in validation state that are not the result of some other object changing state. Such initial events can potentially start a chain reaction of validation state updates.

When the loader acquires a new or modified object, because the synchronizer has added or modified a file in the local repository, it performs a set of checks to determine if it should create an object corresponding to that file and add it to the database. The loader performs the following checks:

– If the syntax of the object does not follow the specifications for that type of object, the object is not added to the database.
– If the object is a certificate or ROA and has expired, it is not added to the database.
– If a certificate is revoked by a validated CRL already present in the database, then the certificate is not added to the database.

Before loading an object into the database, the loader queries the database to determine if a validated parent of the object is present. If no such parent exists, the object is written to the database and put in the awaiting-validation state, for subsequent processing by the *deferred validation algorithm*, described below. If a validated parent does exist, then the object can be tested against a potential certification path. If the parent's key validates the object's signature, then the object is added to the database and marked as validated, potentially starting a chain of validation updates as described in the next section. If the parent's public key is inconsistent with the object's signature, then the object is not added to the database.

If the synchronizer removes a file that corresponds to an object that is still in the database, the loader removes the corresponding object from the database. This can possibly start a chain of validation updates. If the pruner determines that a ROA or certificate has expired, it deletes the object from the database. When a certificate is deleted, this can potentially start a chain of validation updates. If a CRL expires, then the CRL is placed in the CRL-stale validation state, and this can also start a chain of updates.

Note that once a potential path to a Trust Anchor exists, the actual validation of certificates and CRLs is done using existing publicly available software (OpenSSL [19] and cryptlib [20]) that performs the certification path validation checks and verifies the signatures all the way back to the Trust Anchor. An approach that cached the validation state could offer a potential performance improvement, since all but the final link in the chain has already been validated at that point. Note that this potential inefficiency does not exist for ROAs, since we had to write custom code to validate them.

When an object changes validation state this change can propagate through the database. Because of the need for certification path validation, a new certificate being validated or invalidated can ripple to its descendants, i.e. all those objects that use the certificate as part of the path used to determine its own validation state. The algorithms for propagating validation state efficiently and incrementally, including the deferred validation algorithm, are described below.

There are four conditions that can lead to propagation of validation state through the database. These conditions along with the actions they require are:

1. If a certificate moves from the awaiting-validation state to the validated state, then all of its children are tested to see whether they are now valid. (Note that the certificate cannot even reach the awaiting-validation state

unless syntax checks on the certificate have already been successfully performed; therefore the checking being refer to here has to do with signature verification only.) Generally, these children will be in the awaiting-validation state until the missing parent is validated. Testing the signature of a child against the newly valid parent's key either proves the child valid, and hence changes its state to validated, or proves it invalid, and causes it to be removed from the database.

2. If a previously valid certificate is deleted or invalidated, then each of its child objects is declared invalid unless the child object has been re-parented by another certificate (during key rollover, for example). If the parent is deleted or its new state is awaiting-validation, then the state of each child is modified to be awaiting-validation. If the parent's state is CRL-stale, then the state of each child is modified to be CRL-stale.
3. If a CRL is validated, then each of its sibling certificates is tested to see if its serial number is in the CRL's list of revoked certificates, and if so, the certificate is revoked. If the newly valid CRL replaces one that is stale, then each of its sibling certificates should be removed from the CRL-stale state.
4. If a validated CRL becomes stale, its sibling certificates are placed in the CRL-stale state.

Conditions 1 and 2 constitute the core of the deferred validation algorithm. In a typical PKI path discovery propagates *upward*, from a child object to its parent objects. In the RPKI, path discovery propagates *downward*, from a parent object to its children, when new objects arrive. This type of validation is essential for the operation of the RPKI, since ultimately all objects must either be validated or invalidated, even though their order of arrival in the local repository is completely arbitrary. These operations are complicated by the fact that objects can interact with one another. We provide an example scenario of such propagation below.

Table 1. Initial state of example data on 01-JAN

Type	**ID**	**SKI**	**AKI**	**Serial #**	**Expires**	**Validation State**
Cert	1	AB:00	AB:00	123	31-DEC	Validated
Cert	2	13:B5	C1:8D	2	01-FEB	Awaiting
Cert	3	EE:23	C1:8D	345	01-FEB	Awaiting
Cert	4	7D:62	13:B5	3	01-FEB	Awaiting
Cert	5	28:2C	EE:23	7	15-JAN	Awaiting
ROA	1	EE:23			01-FEB	Awaiting
ROA	2	28:2C			01-FEB	Awaiting

The following scenario contains far fewer objects than would be in a real system, with the quantity of objects limited for the purposes of illustration. However, it is otherwise realistic, and is similar to scenarios we used for initial tests of our software. The initial state of the database is provided in Table 1. It does not show all the database fields, just those critical to determining validation state propagation. Certificate 1 is the single Trust Anchor and initially is also the only validated object, since the others do not yet have a certification path back to the Trust Anchor.

Table 2. Missing link in trust chains arrives on 02-JAN

Type	ID	SKI	AKI	Serial #	Expires	Validation State
Cert	1	AB:00	AB:00	123	31-DEC	Validated
Cert	2	13:B5	C1:8D	2	01-FEB	Validated
Cert	3	EE:23	C1:8D	345	01-FEB	Validated
Cert	4	7D:62	13:B5	3	01-FEB	Validated
Cert	5	28:2C	EE:23	7	15-JAN	Validated
ROA	1	EE:23			01-FEB	Validated
ROA	2	28:2C			01-FEB	Validated
Cert	6	C1:8D	AB:00	23	01-FEB	Validated

Table 2 shows the state after a new certificate arrives. This certificate provides the missing link in the certification paths for all the objects. Using the deferred validation algorithm described earlier, the validation state propagates from the new certificate first to its children and then to their children and so on.

Table 3. Certificate 5 expires and its child ROA is invalidated on 15-JAN

Type	ID	SKI	AKI	Serial #	Expires	Validation State
Cert	1	AB:00	AB:00	123	31-DEC	Validated
Cert	2	13:B5	C1:8D	2	01-FEB	Validated
Cert	3	EE:23	C1:8D	345	01-FEB	Validated
Cert	4	7D:62	13:B5	3	01-FEB	Validated
ROA	1	EE:23			01-FEB	Validated
ROA	2	28:2C			01-FEB	Awaiting
Cert	6	C1:8D	AB:00	23	01-FEB	Validated

Table 3 shows the state after certificate 5 expires. Certificate 5 is deleted from the database, and its child, ROA 2, is invalidated and placed in the awaiting-validation state.

Table 4 shows the state after a CRL arrives. This causes one of its sibling certificates, certificate 2, to be revoked, and its child, certificate 4, to be placed in the awaiting-validation state.

Table 5 shows the state after CRL 1 expires. Certificate 3 is a sibling of CRL 1 and is therefore placed in the CRL-stale state. This propagates to its child, ROA 1, which is also placed in this state.

Table 4. CRL 1 arrives on 16-JAN, revoking certificate 2 and invalidating its child

Type	ID	SKI	AKI	Serial #	Expires	Validation State
Cert	1	AB:00	AB:00	123	31-DEC	Validated
Cert	3	EE:23	C1:8D	345	01-FEB	Validated
Cert	4	7D:62	13:B5	3	01-FEB	Awaiting
ROA	1	EE:23			01-FEB	Validated
ROA	2	28:2C			01-FEB	Awaiting
Cert	6	C1:8D	AB:00	23	01-FEB	Validated
CRL	1		C1:8D	2,33	20-JAN	Validated

Table 5. CRL 1 expires on 20-JAN, causing other objects to enter the CRL-stale state

Type	ID	SKI	AKI	Serial #	Expires	Validation State
Cert	1	AB:00	AB:00	123	31-DEC	Validated
Cert	3	EE:23	C1:8D	345	01-FEB	CRL-stale
Cert	4	7D:62	13:B5	3	01-FEB	Awaiting
ROA	1	EE:23			01-FEB	CRL-stale
ROA	2	28:2C			01-FEB	Awaiting
Cert	6	C1:8D	AB:00	23	01-FEB	Validated
CRL	1		C1:8D	2,33	20-JAN	CRL-stale

4 Testing and Experimentation

While large-scale testing and experimentation with the application has been limited, we have been able to do some testing with real data. The RIRs have cooperated to supply data in order to test our software. All certificates and CRLs from the five RIRs and their subordinates have been cached on a single server, with the cache updated intermittently. Noticeably absent from this data are any ROAs, since ROAs are objects that have only recently been defined and therefore are not currently used by ISPs as a means to express origin AS authorization. Therefore we have generated our own ROAs in compliance with the current specification [21].

There are two different scenarios of interest from the viewpoint of performance. The first is the initial synchronization and loading, when the software starts from a clean state and does a full read of all the data. The second is an incremental update, where the software starts with a local repository and database that reflects the state at the time of last execution, and then reads only the changes to the current state. The amount of work required for an incremental update depends on the number of objects added, modified, or deleted since the previous update, which in turn depends in large part on the time since the previous update. We anticipate that an update will be performed roughly once a day; since the number of objects updated in this time period is typically only a fraction of the total number of objects, we focused on the initial synchronization and load as the performance bottleneck. We also evaluated the performance of the load operation when cryptographic validation was omitted, in order to give us an estimate of how much time was spent in validation.

Our test for the initial synchronization and load for the non-ROA objects involved 22,633 certificates and 10,528 CRLs. The total time required was 20 minutes and 2 seconds. This is divided into the time for synchronization with the remote repository, which required 426 seconds (about 3.1 minutes), and the time for the load, which required 776 seconds (12.9 minutes). The synchronization time depends strongly on the network throughput, while the load time depends on the speed of the local computer. (Our tests were done on a single CPU running at 1.6 GHz.) Note that if we had broken the data into five separate remote repositories, 80% of the time for synchronization could have been executed in parallel with the load, hence reducing the time by around 340 seconds (about 5.6 minutes). We anticipate that an incremental synchronization and load would be well under a minute, although we were not able to test this because the repository was not being updated when we performed our experiments. Without cryptographic validation, the time for the load was 226 seconds (about 3.7 minutes); since the average validation path was three or four certificates long, we could have saved roughly 400 seconds (about 6.6 minutes) by validating only the last link in the chain (knowing that the remainder of the chain has already been validated).

We generated 10,000 ROAs to be loaded (but not synchronized because we stored them locally). We do not know how many ROAs will be in real system if this approach is adopted, but this should be within an order of magnitude of the actual number. In a real system, after the RIR and occasional NIR tier, one would expect each CA certificate to be accompanied by at least one EE certificate and one ROA. The number 10,000 is consistent with our 22k certificates and 10k CRLs, on the basis of this reasoning. Loading these ROAs required 281 seconds (about 4.7 minutes).

The remaining question then is how long it takes to generate a file containing the BGP output values specified by these 10,000 ROAs, since this operation will need to be performed every day. The answer is 297 seconds (about 5 minutes). We have already identified a potential modification to the ROA validation step (involving putting more of the ROA data fields in the database), however, and anticipate that this time can be dramatically reduced. Initial experiments indicate that if this proposed modification is implemented, the total amount of time to process all 10,000 ROAs could be reduced to less than ten seconds.

It is worthwhile to note that an alternate implementation of RPKI validation has been carried out at ISC [22]. This code contains an implementation of validation as an integrated part of rsync, their combined version being called "rcynic". It also contains a variety of Python scripts for manipulating certificate and CRL information in a database. While this might appear similar to our RPKI software, BBN's focus has been on generating an end-to-end solution that is optimized for ROA validation and BGP output file generation, and thus we believe that the two are not directly comparable.

5 Conclusion

This paper has described BBN's implementation of a software suite for a resource PKI in which the resources are certificates, CRLs, and, most importantly ROAs. The RPKI software performs all the syntactic and semantic validation steps necessary in order to arrive at a set of trusted AS# to IP-address block assignments that can be used to generate BGP filters. In the course of creating the RPKI software, a novel deferred validation algorithm was developed. The algorithm was optimized for the "validate everything" paradigm of the RPKI. Performance testing indicates that even for very large repositories it will be possible to perform a complete filter generation run on a daily basis. The RPKI continues to evolve as aspects of the RPKI itself evolve.

Acknowledgements

The authors would like to thank Dr. Stephen Kent of BBN, who developed the RPKI concept, and whose guidance was critical to the accomplishment of this project. The authors would also like to thank George Michaelson of APNIC for hosting the BBN RPKI software.

The BBN RPKI software was written by Charlie Gardiner, Joshua Gruenspecht, Marla Shepard, Peiter "mudge" Zatko, and the authors, under the management of Karen Seo.

The development of BBN's RPKI software was partially funded under U.S. Government contract FA8750-07-C-0006. However, the opinions stated in this paper are solely the responsibility of the authors.

References

1. Kent, S., Lynn, C., Seo, K.: Design and Analysis of the Secure Border Gateway Protocol (S-BGP). In: IEEE DISCEX Conference (2000)
2. Kent, S.: An Infrastructure Supporting Secure Internet Routing. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 116–129. Springer, Heidelberg (2006)
3. Housley, R., Polk, W., Ford, W., Solo, D.: Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile. RFC3280 (2002)
4. Rekhter, Y., Li, T.: A Border Gateway Protocol (BGP). RFC4271 (2006)
5. Murphy, S.: BGP Security Vulnerability Analysis. RFC4272 (2006)
6. Kent, S., Lynn, C., Seo, K.: Secure Border Gateway Protocol (S-BGP). IEEE Journal on Selected Areas in Communications 18(4), 582–592 (2000)
7. Goodell, G., Aiello, W., Griffin, T., Ioannidis, J., McDaniel, P., Rubin, A.: Working Around BGP: An Incremental Approach to Improving Security and Accuracy for Interdomain Routing. In: Network and Distributed System Security Symposium, pp. 73–85 (2003)
8. Hu, Y.-C., Perrig, A., Johnson, D.: Efficient Security Mechanisms for Routing Protocols. In: Network and Distributed System Security Symposium, pp. 57–73 (2003)

9. Wan, T., Kranakis, E., van Oorschot, P.C.: Pretty Secure BGP (psBGP). In: Network and Distributed System Security Symposium (2005)
10. Kent, S.: Securing BGP: S-BGP. The Internet Protocol Journal 6(3), 2–14 (2003)
11. White, R.: Securing BGP: soBGP. The Internet Protocol Journal 6(3), 15–22 (2003)
12. Housley, R., Polk, T.: Planning for PKI. Wiley Computer Publishers, Chichester (2001)
13. Opplinger, R.: Secure Messaging with PGP and S/MIME. Artech House Publishers (2000)
14. http://ietfreport.isoc.org/idref/draft-ietf-sidr-res-certs
15. http://www.mysql.com
16. http://mirin.apnic.net/bbn-svn/BBN_RPKI_software/trunk
17. http://mirin.apnic.net/resourcecerts/wiki
18. draft-ietf-sidr-arch-01.txt, http://ietfreport.isoc.org
19. http://www.openssl.org
20. http://www.cs.auckland.ac.nz/~pgut001/cryptlib
21. http://ietfreport.isoc.org/idref/draft-ietf-sidr-roa-format
22. http://subvert-rpki.hactrn.net

Instant Revocation

Jon A. Solworth[*]

University of Illinois at Chicago
solworth@rites.uic.edu

Abstract. PKI has a history of very poor support for revocation. It is both too expensive and too coarse grained, so that private keys which are compromised or otherwise become invalid remain in use long after they should have been revoked. This paper considers Instant Revocation, or revocations which take place within a second or two.

A new revocation scheme, *Certificate Push Revocation (CPR)* is described which can support instant revocation. CPR can be hundreds to thousands of times more Internet-bandwidth efficient than traditional and widely deployed schemes. It also achieves significant improvements in cryptographic overheads. Its costs are essentially independent of the number of queries, encouraging widespread use of PKI authentication.

Although explored in the context of instant revocation, CPR is even more efficient—both in relative and absolute terms—when used with coarser grain (non-instant) revocations.

Keywords: Public Key Infrastructure, Revocation.

1 Introduction

A *Public Key Infrastructure (PKI)* provides a means of binding a public key with the user who is the *key holder*. This binding is in the form of an identity certificate, signed by a Certificate Authority (CA).

Using the key holder's identity certificate, a third party can verify that a statement's digital signature was produced by the key holder: The public key is used to verify that the signature is for the statement and the binding identifies the user.

The party that relies on a signature, called the *relying party*, generally bears the risk if the signature is defective. This risk is reduced by ensuring that the CA is trustworthy and the identity certificate information is current. Trustworthiness comes from relying on the software and procedures for issuing certificates; perhaps the single most important component is that the CA protects its private key from disclosure or misuse. To safeguard the CA's private key, it is desirable to keep it offline, and carefully log its use. This limits how often the CA can sign certificates and hence argues for longed lived certificates, typically on the order of a year. To prevent a relying party from using stale information, it is

[*] This work was supported in part by the National Science Foundation under Grants No. 0627586 and 0551660. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of the National Science Foundation.

necessary to check that the information is still valid—that is, that the binding has not been *revoked*. (In general, an identity certificate can be revoked because the private key was exposed or because the binding is no longer valid.)

Unfortunately, the *revocation problem* has long been a difficultly in PKI, both in obtaining timeliness and efficiency of revocation mechanisms. The cost of revocations is the dominant cost for a PKI [23]. Rivest, for example, suggested short lived certificates [20], but this requires frequent signings and hence exposes the CA's private key to attack, especially if it can no longer be kept off line. Gutmann called revocation a Grand Challenge problem in PKI [6] and discusses several issues with it [5]. At a recent security architectures workshop[1], Sandhu stated PKI's critical need is for instant revocation, which he described as revocations which take place within a couple of seconds. Instant revocation is interesting since it states the problem as a requirement (recency of revocation information) rather than a mechanism for achieving it (eg. on-line queries).

We therefore consider the problem of *instant revocation*, which, following Sandhu, is defined as a revocation mechanism which can invalidate an identity certificate in no more than a second. In addition to invalidation, the revocation must be propagated to its destination to be used by the relying party. To achieve a total revocation delay of two seconds, the propagation delay must be no more than a second[2].

We introduce a new PKI revocation scheme, which we call *Certificate Push Revocation (CPR)*. Our focus here is on the design of the *Validity Authority (VA)* which revokes certificates, the *cache* which hold copies of revocations, and the relying party. (Normally, the VA would be would be part of the CA, but we separate them here to emphasize VA function.) While CPR was developed for instant revocation, it is even more efficient (and some requirements can be relaxed) when used for longer revocation intervals. CPR relies on efficiently pushing the updates towards the relying party, rather than a combination of pull and push as in other schemes.

The instant revocation problem is complicated because authentication takes place in a distributed environment amongst different organizations which may have limited trust in each other. Hence, the authentication should be robust, that is support deniability resistance[3], so that the relying party can prove after the fact that the certificate was authenticated. This is the strongest guarantee to the relying party that a VA can make, since the VA cannot later deny the status that it provided.

[1] ACM 1st Computer Security Architectures Workshop, Panel on Distributed Authentication, Panelists Angelos Keromytis, Ravi Sandhu, and Sara "Scout" Sinclair.

[2] As network latency requirements seems inherent in *any* instant revocation scheme, and are not PKI specific, we do not consider them further here. Note that even on-line checks experience a round trip delay.

[3] We use *deniability resistance* to mean that with a few assumptions (the private key is under the sole control of the key holder and the crypto is not broken) then the key holder must have signed the statement. Of these assumptions, the "sole control" is the most likely to be violated. Nonetheless, deniability resistance has the minimum set of assumptions and therefore is the best basis for dispute resolution.

There are further requirements, since the above timing requirement can be trivially met with existing techniques—for example, a digital signature certifying the status can be computed in well under a second. The scheme therefore must also be efficient in its use of resources—in particular, it must be economical in its use of Internet bandwidth.

Hence, investigators traditionally measure the efficiency of certificate revocation schemes in term of their Internet bandwidth. To analyze and optimize revocation schemes accurately, it is important that the costs be accurately captured in the model. However, the network metric traditionally used is bits/day, although bandwidth is typically priced in terms of *peak* bandwidth usage, as the bandwidth provider must build out network hardware resources to address this peak demand. Hence, if there is significant difference in peak demand vs. average demand, average bit rates do not accurately reflect costs. Therefore we analyze, and our goal is to minimize, peak Internet bit rates. In particular, local network costs are inexpensive and hence ignored (although we show that local bandwidth requirements are modest).

Ideally, a revocation scheme should achieve the following goals:

1. Support instant revocation,
2. Efficiently use peak network bandwidth for the VA,
3. Efficiently use network bandwidth for the relying party,
4. Efficiently use computational resources at the VA,
5. Efficiently use computational resources at the relying party,
6. VA costs should increase minimally as revocation checks increase, and
7. VA vulnerability to attack should be minimized.

Item (1) is a requirement for PKI in high value operations. Items (2)-(5) are necessary to make the scheme economically viable[4]; Item (6) encourages use, thus increasing the value of the PKI facility; item (7) is desirable to protect the VA (and its high value keys) from attack.

CPR achieves all 7 of these goals. In contrast, existing techniques clearly fail to efficiently achieve the two-second goal. Existing techniques either have high cost per use (e.g., a digital signature) or costs which depend on the time granularity (e.g., hash chains) which make them more expensive to use for instant revocation. Techniques which attempt to aggregate unrelated information (such as revocation lists) increase transmission size and thus network costs.

CPR is also suitable for coarser grained revocation. We believe that, when applicable, it sets records as best in class for (1), (2), (3), (4), and (6); in some cases the improvements are by orders of magnitude.

The rest of the paper is organized as follows: Section 2 gives the background, Section 3 describes related work, and Section 4 characterizes VA attributes. Section 5 presents CPR. Section 6 shows the performance of CPR against OCSP

[4] Peak rate is used for the VA and average rate for the relying party. The VA's sole mission is to support PKI (and hence its peak network bandwidth is solely for the support of revocations), while the relying party caries on a variety of tasks and hence it is somewhat less sensitive to peak rates.

and CRS and describes how to combine CPR with other schemes to support low-bandwidth, intermittently connected computers. Section 7 describes security considerations and then we conclude.

2 Background

Revocation schemes have followed a common architecture, using a VA and, optionally, a set of caches. They vary in the algorithms and protocols used to provide up-to-date information. Revocation schemes are centered around the following 4 components:

VA holds the secrets (such as private keys and/or hash chain seeds) necessary for revocation and thus produces deniability resistant validity information.

Caches hold replicas of the information produced by the VA and answer queries as to the validity (revocation status) of certificates. Caches do not hold secrets and need not be trusted. Hence, for schemes in which secrets are necessary to answer queries, the VA answers queries directly and the cache is eliminated.

Signing party is the user whose private key is used to produce a digital signature and who is the subject of authentication.

Relying party is the user who receives a digitally signature, determines who it corresponds to, and validates it.

In general, each of the above entities is administered by a different organization, and hence the network traffic between them is typically routed over the Internet. These parties and the communications between them are shown in Figure 1.

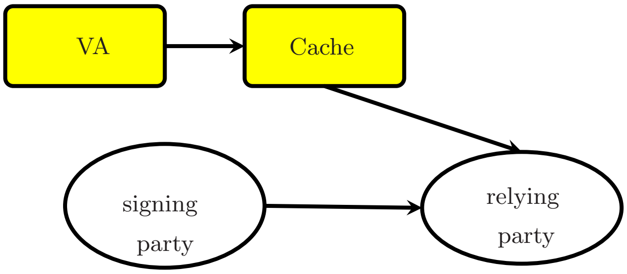


Fig. 1. Parties to a revocation scheme

Although both VA and cache costs are borne by the same entity, they are managed differently because their security needs are different; the VA must be trusted for integrity and availability while caches collectively need only supply sufficient availability. This enables the caches to be outsourced. The cost of a revocation scheme is measured by the cost of Internet traffic from VA to cache (providing the basis of authentication) plus the *queries* to the caches (or VA) as to the validity of specific certificates.

The queries can be made by the relying party or the signing party. With traditional revocations periods, such as a day, cache costs are reduced by sending information to the signing party. Given a single query, the signing party can send the validity information to multiple relying parties until the revocation period expires. However, signer caches has negligible advantage with instant revocation and thus the signing party saves network bandwidth when the cache transmits revocation information to the relying party. Moreover, this optimization does not increase cache costs. It is also more secure, as it prevents the signing party from sending dated information when she knows that her entry is about to be, or already has been, revoked (for example, just after she is fired).

3 Related Work

Initial designs for PKI centered on Certificate Revocation Lists (CRLs) which are lists of bad certificates digitally signed by a VA. CRLs are patterned after off-line bad credit card number lists which were published by credit card issuers and used by retailers; a credit card not in the list was assumed to be good. To reduce the CRL publication and use costs, two CRL variants are used: *segmented CRLs* (partitioned by certificate serial number, alternatively see [12]) or *delta CRLs* (containing the changes since the previously published CRL). As revocations are typically assumed in the literature to be about 10% [15], CRLs constitute a non-trivial share of the VA plus CA databases. CRLs are inefficient due to their size and frequent publication (needed to ensure timeliness) [15,20].

CRLs are patterned after an offline technique, so its natural that there be an on-line version. In order to provide deniability resistance, its necessary for the relying party to have positive proof that a certificate has not been revoked. To provide this property, an *On-Line Certificate Status Protocol (OCSP)* signs each revocation request [18]. OCSP provides high timeliness, but the signature increases query network bandwidth and CPU costs. The OCSP scheme we consider here does not use caches, an alternative to OCSP-based scheme uses multiple private keys to enable some secrets to be moved to the caches [10].

Micali proposed an authentication scheme called *Certificate Revocation System (CRS)* [14,15,16] which combines Lamport's hash chains with certificates. Lamport's scheme uses a cryptographic hash $H(x)$, applied n times to a *hash seed* s, $H^n(s)$ to support n authentications [11]. The hash function H and the top of the hash chain $H^n(s)$ are public information while s is secret. Authentication i is proved by providing $a_i = H^{n-i}(s)$ which can be verified by showing that $H^n(s) = H^i(a_i)$. It is computationally infeasible to determine s from $H(s)$. CRS uses a hash value for each revocation period. Hash chains have two main advantages over signing; hash values are an order of magnitude smaller, and about 10,000 times more efficient to compute, than a signature.

Hashing is also useful to summarize an arbitrary amount of data, such as some part of the VA database. If only part of the data is used or changed, then an efficient alternative to a simple hash is a tree of hash values—called a Merkle Tree [13]. As with a single hash, the root of the Merkle Tree summarizes the whole data. The tree is constructed recursively from the children; given two children containing values d_0 and d_1, the parent contains $H(d_0|d_1)$ where '|' is concatenation. Given S values, stored at the leaves, $\log S$ hash values are needed to compute the root of the tree and thus verify the value.

Certificate Revocation Trees (CRTs) uses a tree whose leaves are ranges of values [9]. A certificate is valid if its serial number falls within one of the ranges of the CRT and is otherwise revoked. Every new time period, a signed list of revoked certificates is sent to the cache, which updates (and rebalances) its tree. The drawback of this scheme is that certificate validations requires logarithmic (in the number of certificates) hash values—using a Merkle Tree—plus a signature. Hence, it reduces VA-to-cache costs while increasing query costs.

To reduce VA to cache costs versus CRS, Goyal proposed using hash chains (based on CRS) to authenticate groups of certificates [4]. The hash chain is attached to a validity certificate which has a validity bit for each certificate in the range. If one of the (previously valid) certificates became invalid, a new certificate would be issued with a new hash chain. This scheme increases revocation-to-cache costs while decreasing query costs.

We do not consider certificate chains in our performance comparisons. All of the schemes (including our own) handle certificate chain processing, but such considerations drive up the authentication cost of other schemes while having no effect on our own. The cost of chains can be reduced for other schemes using synthetic certificates [21].

We consider only revocation here, and not the reason for revocation (i.e., status), although CPR could be extended to do so. For a fuller discussion of PKI revocation and status issues see [8,7].

An alternative approach for revocation is to revoke public keys rather than the certificates which contain them. For example, security mediated PKI does this with a mediator which the relying party queries [24,25]

4 Characterization of VAs

We next describe the parameters which have the greatest impact on the performance of revocation schemes. Each certificate is assumed to have a serial number; serial numbers are consecutively issued by the CA[5]. Two of the most significant parameters of the VA are the *number of certificates*, denoted N and the *annual revocation rate*, denoted R. From these, the size of the active range of serial number, $N_r = N/R$ can be derived. Q is the number of authentication (queries) per user per day. We are interested in peak performance, the peaks are characterized relative to average authorizations and revocations:

– P_a is the ratio of authentications in the busiest second to that of the average second, and
– P_r is the ratio of revocations in the busiest second to that of the average second.

[5] It is possible to extend this scheme to allow, for example, multiple series of sequence numbers.

We consider two VA sizes, the small VA with $N = 10,000,000$ and the large VA with $N = 100,000,000$. These values are summarized in Table 1. To be conservative, we have chosen P_a ranges to be relatively modest (since that affects only traditional schemes) while chosen P_r ranges to be more aggressive (since that affects only CPR).

Table 1. Certificate Authority statistics

Parameter	Meaning	Small VA	Large VA
N	Certificates	10,000,000	100,000,000
R	Annual revocation fraction	.1	.1
N_r	Size of serial number range	N/R	N/R
Q	Number of authentications per certificate per day	1–64	1–64
P_a	Peak–ratio of authentications	1–10	1–10
P_r	Peak–ratio of revocations	1–100	1–100

Table 2. Crypto++ benchmarks on AMD Opteron 2.4 GHz processor

Operation	Time	Cycles
SHA-1 (Hashing)	0.546 μs	999 + 10.6 cycles/byte
RSA 2048 Signature	5,950.000 μs	10,890,000
RSA 2048 Verification	150.000 μs	280,000

The schemes described here use cryptographic hashing, digital signatures, and Merkle (Hash) Trees. For the purposes of presenting performance, we use 2048-bit RSA signatures and SHA-1 hashing. The cost of cryptographic operations are shown in Table 2 (for further information see `http://www.cryptopp.com/benchmarks-amd64.html`). The table's numbers are for a current but inexpensive processor.

5 Certificate Push Revocation (CPR)

CPR provides the relying party with sufficient information to validate *any* certificate. Figure 2 shows the parties to CPR. Rather than the cache being owned by the VA as in other schemes, in CPR the cache is owned by, and located at, the relying party[6]. This does 3 things

– The VA does not pay *any* cache costs, resulting in substantial savings. These savings include elimination of cache hosting and, most importantly, queries;
– The cache cost (including queries) is essentially zero, since it is co-located with the relying party which performs revocation checking; and
– The relying party, which is taking the risk, can make tradeoffs which minimize costs.

[6] We distinguish our scheme from others in which the cache must be trusted.

Because the cache has no secrets (integrity depends only on VA operations) moving it to the relying party does not reduce security. The cost to the relying party is very low.

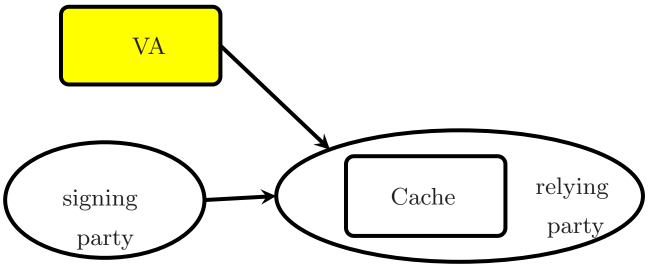


Fig. 2. Parties to CPR

Locating the cache at the relying party enables it to perform many authentications per second for not much more than the price of a single authentication. But more importantly, it is the relying party which needs the assurance that the information is valid (and thus will invest appropriately to protect it).

Finally, the relying party can chose when to take or mitigate risks depending on its business model. For example, it might choose a lower cost (and availability) system at the price of increased fraud. Different businesses are subject to different risks, for example a web-based retailer may choose after-the-fact verification, while an investment brokerage firm may refuse to perform a transaction without up-to-date verification. Indeed, such a tradeoff is inherent [2,3].

Because the cache is maintained at the client site, and CPR has many caches, we need to consider cache failure. Of course, other schemes need to recover from cache failure, but papers have traditionally ignored this cost because it is small in traditional schemes. Hence, we'll measure it carefully for CPR and assume it's cost is zero for other schemes.

Moreover, to ensure that we count all costs and provide maximum security for the VA, access to the VA is limited to the minimum necessary: that which performs revocation and addition of new certificate serial numbers. To enable recovery under this scenario, the VA continually broadcasts recovery information. If a cache fails, it simply listens to this broadcast channel until it is up to date.

Instant revocation relies on posting revocations once a second. In CPR, all revocations for the past second are posted to the cache. Revocations are the slowest changing part of the system, and so having updates depend on revocations is likely to be more efficient than other techniques (this is how CRT achieves low VA-to-cache costs). For example, assuming an annual revocation rate of 10%, the small VA will have 0.032 revocations per second while the large VA will have 0.317 revocations per second. This enables the *certificate validity vector*—a vector which contains a bit for each certificate indicating whether that certificate has been revoked—to be kept up-to-date at the relying party. Furthermore, although each cache holds the entire revocation database, it is a relatively small data structure, for example, for the Large VA it is less than 20 megabytes.

Our protocol is exceeding simple. Every second it sends out:

1. The summary (i.e. the root of the Merkle tree) of certificate validity vector,
2. The revocations in the last second, and
3. Part of the certificate validity vector for some range of certificates.

Item (1) is used to secure revocations. Item (2) is used to update certificate validity vector. Item (3) is used for recovery after a cache failure. (Recovery after a VA failure is provided by simply sending out the "updates" once the VA comes up).

Revoker. We first describe the revoker, which contains items (1) and (2). The revoker data structure is shown in Table 3. We have tried to be conservative in its design (no more than the number of bits specified are required), although if a few thousand more bits are needed it is of little consequence as only one revoker is issued per second.

Table 3. Revoker

Name	Bits	Purpose
time	64	Number of seconds since epoch
min	32	Minimum certificate serial number
max	32	Maximum certificate serial number
hash	160	Merkle hash root of certificate validity vector
s_{rv}	32	size of the *rv*
rv	$32s_{rv}$	Revocation vector
signature	2048	RSA 2048 bit signature
	$2368 + 32s_{rv}$	total bits

It contains the time in seconds since some epoch as a 64-bit integer; to ensure that the relying party has the latest update, it needs a reasonably accurate clock[7].

The min and max are the minimum and maximum certificate serial numbers. The max increases when new certificates are issued. The min increases when the lowest certificate serial number for the VA is either revoked or expires.

The revoker "points" to a *certificate validity vector,* a bit vector of length $\max - \min + 1$ with bit i being 1 if certificate $\min + i$ is not revoked and 0 otherwise. As certificates are issued in order, and about 10% are revoked during the year, the bit vector is of length N_r.

[7] Time synchronization protocols such as Network Time Protocol (NTP) easily provide synchronization to a small fraction of a second [17]. Other time sources include GPS and radio receivers; obtaining an accurate time source is inexpensive. Moreover, the relying party can always use a slightly older revoker (i.e., the last one received).

Table 4. Certificate Validity Segment

Name	Bits	Purpose
time	64	Number of seconds since epoch
startVec	32	starting certificate serial number of the vector
sizeVec	32	size of the validity vector
valVec	$N_r/T_{recovery}$	$1/T_{recovery}$ fraction of the validity vector
signature	2048	RSA 2048-bit signature

Finally, to enable the cache to update its certificate validity vector, the revocations that occurred in the last H seconds are sent (H can be set to 1 or larger, and is a parameter related to fault tolerance). The cost of additional revocations (due to $P_a > 1$ or $H > 1$) is quite small because the average number of revocations is very small and because a 32-bit serial number is significantly smaller than a 2048-bit signature (both are fields of the revoker).

The algorithm at the cache is as follows:

– Each second the revoker is received and used to update the local copy of the certificate validity vector (by turning off bits corresponding to the revoked certificates and adjusting the bit vector to reflect the new valid range);
– The root of the Merkle tree is computed over the bit vector;
– The computed hash is valid if it equals the hash in the revoker and the signature in the revoker is valid; and
– The bit vector can then be used to check an arbitrary number of revocations in the next second.

Certificate Validity Segment. The certificate validity segment, used to construct the certificate validity vector at the cache, is described here.

Every second a certificate validity segment containing $1/T_{recovery}$ of the certificate validity vector is transmitted; in $T_{recovery}$ seconds the entire certificate validity vector is transmitted. Hence, smaller $T_{recovery}$ results in faster recovery after a failure, but at a higher bit rate.

The fields of the certificate validity segment are shown in Table 4. The time field is as in the revoker. The current portion of the certificate validity vector is described by:

startVec the starting certificate serial number of the vector,
sizeVec the size of the vector, and
valVec the bits of this segment of the certificate validity vector.

Its possible to save some bits (the signature and some time) by combining the certificate validity segment with the revoker. However, the certificate validity segment is used only for recovery while the revoker is used to validate signatures, so there is an advantage of making the revoker small as it reduces average relying party bandwidth requirements.

Table 5. OCSP network bandwidth and CPU costs

		Auth/sec		**bits/sec**		**CPU cores**	
Q	P_a	Small VA	Big VA	Small VA	Big VA	Small VA	Big VA
1	1	115.74	1,157.41	244,444.44	2,444,444.44	0.69	6.89
4	1	462.96	4,629.63	977,777.78	9,777,777.78	2.75	27.55
16	1	1,851.85	18,518.52	3,911,111.11	39,111,111.11	11.02	110.19
64	1	7,407.41	74,074.07	15,644,444.44	156,444,444.44	44.07	440.74
1	5	578.70	5,787.04	1,222,222.22	12,222,222.22	3.44	34.43
4	5	2,314.81	23,148.15	4,888,888.89	48,888,888.89	13.77	137.73
16	5	9,259.26	92,592.59	19,555,555.56	195,555,555.56	55.09	550.93
64	5	37,037.04	370,370.37	78,222,222.22	782,222,222.22	220.37	2,203.70
1	10	1,157.41	11,574.07	2,444,444.44	24,444,444.44	6.89	68.87
4	10	4,629.63	46,296.30	9,777,777.78	97,777,777.78	27.55	275.46
16	10	18,518.52	185,185.19	39,111,111.11	391,111,111.11	110.19	1,101.85
64	10	74,074.07	740,740.74	156,444,444.44	1,564,444,444.44	440.74	4,407.41

6 Performance

The performance of revocation schemes is measured here by two metrics: The primary performance metric is Internet network bandwidth and secondarily, the cost of performing cryptographic operations.

In particular, we consider the effect of peak rate authentications and revocations[8]. This will almost certainly be significantly higher than the average rates, and the VA infrastructure must be sized to accommodate them. For example, financial markets are typically opened only a limited number of hours per day and in addition tend to see the greatest volume at opening and closing; moreover, trading volume varies widely on different days. In retail, peak sales are often related to Christmas or New Years shopping, depending on country. It is the peak load for which these systems must be sized, for example Amazon.com sizes its systems to work with extreme reliability even for its peak load during the 2 weeks prior to Christmas. It would seem that minimum peak to average ratios would be at least 5 although much higher peaks would be necessary in many scenarios. Sizing for peak loads significantly drives up costs; ignoring peak factors significantly distorts costs.

CPR outperforms traditional schemes under peak rates. But CPR's advantages do not depend on peak rates—it significantly outperforms other schemes under average rate measurements as well ($P_a = P_r = 1$). Our technique's advantages increase as the peak authentication ratio increases.

6.1 OCSP

The cost of OCSP depends on Q (average number of authentications per user per day) and P_a. In Table 5, values are given for P_a equals 1, 5, and 10 and values

[8] Although CPR performance is independent of query rate, other schemes such as OCSP and CSR are not.

Table 6. CRS network bandwidth and CPU costs

		Auth/sec		**bits/sec**		**CPU cores**	
Q	P_a	Small VA	Big VA	Small VA	Big VA	Small VA	Big VA
1	1	115.74	1,157.41	22,222.22	222,222.22	5.46	54.6
4	1	462.96	4,629.63	88,888.89	888,888.89	5.46	54.6
16	1	1,851.85	18,518.52	355,555.56	3,555,555.56	5.46	54.6
64	1	7,407.41	74,074.07	1,422,222.22	14,222,222.22	5.46	54.6
1	5	578.70	5,787.04	111,111.11	1,111,111.11	5.46	54.6
4	5	2,314.81	23,148.15	444,444.44	4,444,444.44	5.46	54.6
16	5	9,259.26	92,592.59	1,777,777.78	17,777,777.78	5.46	54.6
64	5	37,037.04	370,370.37	7,111,111.11	71,111,111.11	5.46	54.6
1	10	1,157.41	11,574.07	222,222.22	2,222,222.22	5.46	54.6
4	10	4,629.63	46,296.30	888,888.89	8,888,888.89	5.46	54.6
16	10	18,518.52	185,185.19	3,555,555.56	35,555,555.56	5.46	54.6
64	10	74,074.07	740,740.74	14,222,222.22	142,222,222.22	5.46	54.6

for Q of 1 to 64. When $Q = 1$ and $P_a = 1$, the result is 116 authentications per second for the small VA and 1157 per second for the large VA. As each authentication requires 2112-bits, (a 32-bit certificate serial number, a 32-bit time, and a signature). Minimum bit rates for this scheme are about 244,000 bits/second for the small VA and 2,444,000 bits/second for the large VA. They increase linearly with the number of queries per day. For the large VA, they exceed over a billion bits per second in a system which has $Q = 64$ and $P_a = 10$.

The cost of computing signatures is also significant. While very low authentication and peak rates can be performed with just a few processing cores[9], high rates result in the need for hundreds or thousands of cores (or custom cryptographic hardware). Furthermore, extra expense is incurred for these cores to suitably protected the highly sensitive private key of the VA.

6.2 CRS

CRS achieves a significant savings over OCSP because hash values are so much smaller than signatures (160 bits vs. 2048 bits) and because the time is implicit in CRS (by the hash value's location in the hash chain). CRS's scheme gains a factor of 11 in network bandwidth. The corresponding CRS performance in shown in Table 6. Nevertheless, CRS requires significant network bandwidth of up to 142,000,000 bits/second.

There are also significant savings in CPU cores for CRS vs. OCSP, although the time to compute the hash chains needed is non-trivial; 5.46 processor cores for the small VA and 54.6 cores for the large VA. This under counts the number of cores, since it does not count run-time computation needed to recompute these hash values.

[9] Most desktop, notebook, or server processor chips today have 2-4 cores, or independent CPUs, and thus able to execute 2-4 independent programs simultaneously.

6.3 CPR

In CPR, the VA transmits a revoker and certificate validity segment every second. The Internet Protocol's multicast, enables the same packet to be delivered to an arbitrary number of Internet destinations. Two multicast groups are used, one for revoker and another for certificate validity segment traffic. A relying party can join and leave these multicast groups independently. When a relying party is running, it is always part of the revoker multicast group; it is only part of the certificate validity segment multicast group when recovering from a failure. The latter is needed for making CPR reliable.

We consider first the revoker. Its only variable component is the size of the rv containing revoked serial numbers (see Table 3). The average number of revocations/second is .032 for the small VA and .317 for the large VA. The peak number of revocations sent is $P_r \cdot H$ times the average number of revocations. We note that bandwidth usage is not too heavily related to this value for the ranges of P_r and H considered here. Moreover, CPR is totally independent of the number of queries.

The results for a variety of values of H and P_r are shown in Table 7. For example, when $H = 10$ and $P = 100$ (a very conservative choice), the average number of bits during peak load is 3,382 for a small VA and 12,512 for a large VA. These are the only updates needed during normal operation.

Table 7. Revoker bits

		avg. bits/second				avg. bits/second	
H	P_r	Small VA	Large VA	H	P_r	Small VA	Large VA
1	1	2,369.01	2,378.14	10	1	2,378.14	2,469.44
1	5	2,373.07	2,418.72	10	5	2,418.72	2,875.20
1	10	2,378.14	2,469.44	10	10	2,469.44	3,382.40
1	50	2,418.72	2,875.20	10	50	2,875.20	7,440.00
1	100	2,469.44	3,382.40	10	100	3,382.40	12,512.00
5	1	2,373.07	2,418.72	20	1	2,388.29	2,570.88
5	5	2,393.36	2,621.60	20	5	2,469.44	3,382.40
5	10	2,418.72	2,875.20	20	10	2,570.88	4,396.80
5	50	2,621.60	4,904.00	20	50	3,382.40	12,512.00
5	100	2,875.20	7,440.00	20	100	4,396.80	22,656.00

The cost of cryptography at the VA is extremely modest (it is even less at the relying party).

- Two RSA signatures—one signature for the revoker and one for the certificate validity segment (11.90 milliseconds),
- Update of the Merkle tree for a Small/Large VA (39/347 microseconds), using the maximum $P_r = 100$.

That is, the cryptographic cost is a small fraction of a single core.

The cost to the relying party is significantly smaller than to the VA, as it verifies a signature (150 μs) rather than signs (5,950 μs). Hence, the CPU cost/second at the relying party is less than 1 millisecond (339/647 μs).

We next consider the certificate validity segment. The peak network bandwidth on this multicast channel is dependent on $T_{recovery}$. We consider recovery periods from one to five minutes. Our goal is to measure the bandwidth from two points of view, the cache and the VA. The bandwidth requirements depend on $T_{recovery}$ and are a bit more than a T1 line[10] (each 1.536 MBits/s) at 60 second recovery and only a few hundred thousand bits/second at a 5 minute recovery.

Table 8. Instant revocation recovery information network costs

	bits/second	
$T_{recovery}$	Small VA	Big VA
60	187,361.19	1,854,127.85
120	94,768.59	928,201.93
300	39,213.04	372,646.37

CPR is designed for instant revocation. However, it is the only scheme analyzed whose network cost *decreases* with longer revocation intervals (mostly due to fewer signatures). All of the other scheme's network cost depend only on the number of queries, and are therefore independent of the revocation interval. Hence, CPR's absolute and relative performance advantages *improve* with longer revocation intervals, making it advantageous to use at all revocation intervals.

6.4 Blended Schemes

CPR is clearly very inexpensive for the VA, so we now consider its effect on the relying party. For a relying party with a high speed Internet connection, CPR's bandwidth use is modest, especially given ISP practices[11]. However, instant revocation may be impractical for computers connected intermittently or at low-speed. These schemes can also be used to avoid firewall limitations in large companies, by setting up a directory outside the firewall.

In general, cache schemes are either trusted or untrusted. A *trusted cache* means that the relying party cannot verify that the information provided by the cache is the same as that provided by the VA. For example, a trusted cache might be maintained by one's employer. An *untrusted cache* means that the cache must provide a deniability resistant proof that the cache information is the same as that provided by the VA.

[10] The standard unit of commercial Internet bandwidth.

[11] Cable/DSL ISP's typically have a *gap* between actual vs. advertised bandwidth, due to insufficient Internet bandwidth. However, CPR is very low cost in terms of Internet bandwidth due to multicast, and hence its bandwidth may be almost free (decrease the gap) rather than slowing down other connections.

There are three *no-cost cache extensions*—i.e., they do not increase VA bandwidth or computational costs—which enable intermittent or low-speed computers to be connected:

trusted cache. Each query to a trusted cache can be answered with "revoked" or "non-revoked". This combines Gutmann's on-line query with deniability resistance from the trusted cache. An example of a trusted cache is the webDAV cache [1].

untrusted cache (ISP). A cache provided by the relying party's ISP does not consume additional Internet bandwidth (only bandwidth between the relying party and ISP)[12].

untrusted cache (signing party). Cache information can be provided by the signing party, assuming the signing party is not low bandwidth.

Untrusted caches for CPR can be implemented most easily with CRT-style query responses as no change is needed in VA-to-cache traffic. The query response includes the latest revoker, the 512-bit part of the certificate validity vector containing the certificate's validity bit and a logarithmic number of hash values (3072-bits plus the revoker). The relying party can then verify that the query response was provided by the VA [13].

Table 9. Comparison of different schemes

	Costs				increased revoc period	
	Crypto	Network	Deniability	Uses	Reduced	Reduced
Technique	Cost	Cost	Resistance	Cache	Networking	Crypto
OCSP	High	High	yes	no	no	no
CRS	Medium	Medium	yes	no	no	yes
CRT	Low	High	yes	yes	yes	yes
CPR	Low	Low	yes	yes	yes	yes
trusted dir.	Low	Low	no	yes	no	no

If the above techniques are not sufficient for revocations, CPR can be combined with other schemes to reduce overall system costs. It is profitable to do so, because instant revocation minimizes query costs which dominate overall VA costs in traditional schemes; hence satisfying even a fraction of the queries by instant revocation will save Internet bandwidth.

Therefore, the VA can also do OCSP or CRS style revocation. This would increase VA query costs, which need to go out over the Internet. To a first order approximation, if the percentage of requests which can be answered by CPR, the

[12] This is advantageous to the ISP as it reduces Internet query costs. Fielding of such a service is analogous to providing DNS servers, which all ISP's do.

[13] Further optimization can be achieved by splitting the revoker into two components, one with the Merkle tree root and the other with the revoked certificate serial numbers.

ISP, or a trusted cache is p, then p is the savings in bandwidth over a non-blended approach. The differences in the various schemes are summarized in Table 9.

We believe that with the no-cost cache extensions above, the vast majority of revocations can be answered without the VA incurring Internet query costs.

7 Security Considerations

Since the VA only countersigns certificates from the CA, its security implications are limited. A compromised VA can:

– Revoke certificates which are valid or
– Fail to revoke certificates which are invalid.

Such security violations can be detected by auditing at any relying party.

Any VA is going to depend on a correct stream of new certificate and revocation information. Similarly, any VA is subject to denial-of-service attacks on the VA or VA-to-cache path.

Other than that, the VA is the only trusted entity for the integrity of the system. The VA signs both the changes and the hash of the certificate validity vector. Hence, assuming that its private key is kept private, the signature and hashing scheme are not broken, and that its rather simple calculations are performed correctly, the integrity of the system is ensured.

Further, we assume that revocations are not confidential. Knowledge of certificate validity is public, and hence its disclosure does not violate security.

8 Conclusions and Future Work

We have presented, CPR, a revocation scheme which is capable of instant revocation and is efficient. This is the first practical PKI revocation scheme which meets this goal.

It is efficient in network traffic as it reduces external network traffic to little more than the rate of revocations, the slowest changing part of the revocation system. It shows improvements in Internet bandwidth of 100s to 1000s times over well known and widely used techniques. It is so efficient that it is trivial to add in redundancy so that the scheme is robust.

Additionally, it requires very little computational resources and can be 10s to 1000s of times more efficient than other schemes. For the case of low bandwidth, intermittently connected devices CPR can be blended with other techniques. Several of these blends do not increase VA costs over a pure instant revocation scheme.

CPR eliminates the VA's need to provide caches—and the queries made by signing or relying parties against them—by co-locating the cache with the relying party. It does this at very low cost to the relying party. Finally, it allows the relying party to make tradeoffs of availability vs. risk of fraud by using somewhat slightly older revocation information.

Although our goal was to meet requirements of instant revocation, the scheme is perfectly suitable for revocations which need not be as timely, as it is even more efficient for longer revocation intervals than it is for instant revocation.

We have started to build a CPR prototype and associated infrastructure using a simplified but powerful certificate system called sayAnyting [22]. These mechanisms are to be used, among other things, as part of an enterprise-wide authentication system [19].

References

1. Chadwick, D.W., Anthony, S.: Using webDAV for improved certificate revocation and publication. In: López, J., Samarati, P., Ferrer, J.L. (eds.) EuroPKI 2007. LNCS, vol. 4582, pp. 265–279. Springer, Heidelberg (2007)
2. Fox, A., Brewer, E.A.: Harvest, yield and scalable tolerant systems. In: Workshop on Hot Topics in Operating Systems, pp. 174–178 (1999)
3. Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33(2), 51–59 (2002)
4. Goyal, V.: Certificate revocation using fine grained certificate space patitioning. In: Financial Cryptography and Data Security Conference (2007)
5. Gutmann, P.: PKI: It's not dead, just resting. IEEE Computer 35(8), 41–49 (2002)
6. Gutmann, P.: Drawing lessons. In: 3rd PKI workshop (2004)
7. Iliadis, J., Gritzalis, S., Spinellis, D., Cock, D.D., Preneel, B., Gritzalis, D.: Towards a framework for evaluating certificate status information mechanisms. Computer Communications 26(16), 1839–1850 (2003)
8. Iliadis, J., Spinellis, D., Gritzalis, D., Preneel, B., Katsikas, S.: Evaluating certificate status information mechanisms. In: CCS 2000: Proceedings of the 7th ACM conference on Computer and communications security, pp. 1–8. ACM, New York (2000)
9. Kocher, P.C.: On certificate revocation and validation. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 172–177. Springer, Heidelberg (1998)
10. Koga, S., Sakurai, K.: Proposal and analysis of a distributed online certificate status protocol with low communication cost. IEICE Transactions 88-A(1), 247–254 (2005)
11. Lamport, L.: Password authentification with insecure communication. Commun. ACM 24(11), 770–772 (1981)
12. Lopez, J., Mana, A., Montenegro, J.A., Ortega, J.J.: PKI design based on the use of on-line certification authorities. Int. J. Inf. Sec. 2(2), 91–102 (2004)
13. Merkle, R.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)
14. Micali, S.: Efficient certificate revocation. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA (1996)
15. Micali, S.: Efficient certificate revocation. In: Proceedings of the RSA Data Security Conference (1997)
16. Micali, S.: NOVOMODO: Scalable certificate validation and simplified PKI management. In: 1st PKI Workshop (2002)
17. Mills, D.L.: Network Time Protocol (version 3) specification, implementation and analysis. Internet Request for Comment RFC 1305, Internet Engineering Task Force (March 1992)

18. Online certificate status protocol, version 2. Working document of the Internet Engineering Task Force (IETF)
19. Radhakrishnan, M., Solworth, J.A.: Netauth: Supporting user-based network services. In: Usenix Security (2008)
20. Rivest, R.L.: Can we eliminate certificate revocations lists? In: Financial Cryptography, pp. 178–183 (1998)
21. Russell, S., Dawson, E., Okamoto, E., Lopez, J.: Virtual certificates and synthetic certificates: new paradigms for improving public key validation. Computer Communications 26(16), 1826–1838 (2003)
22. Solworth, J.A.: What can you say? and what does it mean? In: Workshop on Trusted Collaboration, IEEE, Los Alamitos (2006)
23. Stubblebine, S.: Recent-secure authentication: Enforcing revocation in distributed systems. In: Proceedings 1995 IEEE Symposium on Research in Security and Privacy, May 1995, pp. 224–234 (1995)
24. Vanrenen, G., Smith, S.W., Marchesini, J.: Distributing security-mediated PKI. Int. J. Inf. Sec 5(1), 3–17 (2006)
25. Yang, J.-P., Sakurai, K., Rhee, K.H.: Distributing security-mediated PKI revisited. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 31–44. Springer, Heidelberg (2006)

Optimized Certificates – A New Proposal for Efficient Electronic Document Signature Validation

Ricardo Felipe Custódio[1], Martín A. Gagliotti Vigil[1], Juliano Romani[1], Fernando Carlos Pereira[2], and Joni da Silva Fraga[2]

[1] Laboratório de Segurança em Computação (LabSEC)
Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil
{custodio,vigil,juliano}@inf.ufsc.br

[2] Programa de Ps-Graduação em Engenharia Elétrica
Universidade Federal de Santa Catarina (UFSC)
{fernando,fraga}@das.ufsc.br

Abstract. Optimized certification is a new method for efficient certificate path verification and digital signing. The basic idea is to issue special certificates (called optimized certificates) for an electronic document to replace the signer's certificate. Optimized certificates are issued to be only valid for a specific time, i.e., the fields *notBefore* and *notAfter* of the certificate are the same. Therefore, certificate revocation are not a requirement as it is no longer necessary to request the status of certificates from a certification authority repository to validate signatures.

1 Introduction

Electronic documents are normally signed using public key cryptographic algorithms [1,2,3] and digital certificates issued by certification authorities [4]. The signature, however, can only be considered valid if the verifier can trust the signer's certificate.

A final CA normally issues certificates to a signer, that is part of a certification chain. The Root CA of this chain is a trust anchor, which both the signer and the verifier have to trust. To validate the user's certificate it is necessary to validate all certificates in this chain.

The validation of a digital certificate is made both when the signer and the verifier of the electronic document sign and later verify it. It is necessary for the signer to verify if the certificate is still valid in order to be able to generate a valid signature. Moreover, each time the verifier checks the integrity and the authenticity of a document, they need to validate the certificate of the signer.

This work presents an new way of producing artifacts for use in the validation of the electronic document signatures. These artifacts are a new kind of certificate called **optimized certificate**. They are a replacement for the signer's certificate. Additionally, the optimized certificate can be also used as a timestamp of the signature or to show the time when the signature was certify as valid. The proposed method is ideal for computational systems under special conditions of limited energy (applications in wireless, ad-hoc nets, etc.).

Section 2 presents a discussion about validating digital certificates. This is important for understanding the advantage of optimized certificates. Section 3 presents in detail the concept of optimized certificates, its benefits and main applications. Section 4 introduces the optimized certificate certification authority and adopted procedures for the issue of optimized certificates. Section 5 deals with the validation of digital signatures using optimized certificates. Section 6 presents a comparison between optimized certificates and traditional ones and Sect. 7 contains the conclusion of the paper and some suggestions for future works.

2 Digital Certificate Validation

In general, certification path processing consists of two phases: *path construction* and *path validation* [5]. Path construction is the determination of one or more paths between signer certificate and root CA certificate resulting in *certificate chains*. The validation phase consists of verifying each certificate contained in a chain. It includes checking its integrity, the certificate's period of validity, the existence of any revocation information and certificate policies [6].

The more certificates in a chain, the more expensive it is to verify the signature. Levi et al. [7] improved this process by using nested certificate-based PKI that reduce the amount of computer power needed for certificate chain verification. However, this process comes with a high processing cost to CAs, mainly in CAs where the number of certificates issued is high in a short period. Revocation status checking is one of the highest costs to verify the signature. This is considered an obstacle to use of signed electronic document on a large scale [8]. This has motivated researchers to pursue alternatives and more efficient verification schemes to establish the revocation status of certificates [9,10,11,12,13,14,15]. Some defend the idea of eliminating certificate revocation [16], through the use of short lifetime certificates. However, there is no consensus about the best way of verifying a certificate's validity.

Moreover, to validate a signature it is necessary not only to verify digital certificate revocation status but also each signature of certificate in the path certification. It can be argued that the ideal PKI model would be a tree with only one CA: the root. The Root CA would be the self-signed certificate and common trust anchor for all users within this tree. This CA would issue all end user certificates as illustrated in Fig. 1. The figure shows a root CA and three end user certificates. The user certificate u has a validity period v from t_1 to t_2. This scenario would configure the ideal PKI from the point of view of the user, since it represents the simplest form of a hierarchical PKI in which only two levels exist [17].

On the other hand, from the point of view of CA managements, a one-level tree is inadequate, because only one CA is responsible for issuing all certificates. In order to provide scalability, it is necessary to increase the number of CAs to distribute the certificate issuance load. These CAs normally are intermediate CAs, which issue certificates to another intermediate CA or end users.

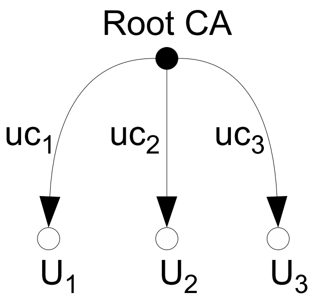


Fig. 1. One level PKI

The major problem with certificate validation is its high cost. Furthermore, this operation is inefficient and it often needs to be performed. To prevent this repetitive process, certificates can be issued moments before their use. This reduces the likelihood of revocation and eliminates the need for revocation schemes [16]. However, this is not common practice, as this leads to frequent certificate re-issue.

The next section describes an optimized certificate to replace the signer certificate to give faster and less costly processing of certificate validation.

3 Optimized Certificates

Optimized certificates (OC) aim to improve the validation of electronic document signatures, reducing the redundant processing necessary in the traditional process. With OCs, it is possible to minimize or eliminate the necessity of checking the revocation state of the signer certificate. The OC is on a basis of the information about the conventional signer certificate, the signature of the document and the validation of its certificate chain. The OCs are issued only once for each document and substitute the traditional signer certificate as illustrated in Fig. 2.

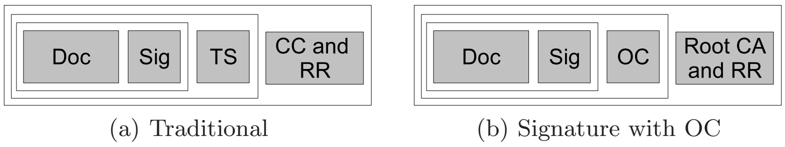


Fig. 2. Digital signature of electronic documents

In this figure, **Doc** is the electronic document, **Sig** is a digital signature, **TS** is the timestamp, **CC** is the certification chain, i.e., all certificates from signer certificate to root CA certificate, **RR** is the list of revoked certificates that allow the verification of the revocation state and **CA Root and RR** are the Root CA certificate and the respective list of revoked certificates. Comparing the figures 2(a) and 2(b) it can be shown that the **TS** and **CC and RR** used in the structure of a traditionally signed document are substituted by two certificates: the Root CA and the OC.

In contrast from a traditional certificate, the OC is valid only for one specified instant of time k, i.e., $t_1 = t_2$ of validity period, where t_1 is the beginning of validity and $t_2 =$ is the end of validity. This characteristic eliminates the necessity of any revocation method associated with this certificate, as it does not make sense to revoke it; if it was issued, it was valid.

The OC contains all information necessary to validate an electronic document signature, this is the same information as in traditional signer certificate. It is practically a copy of this certificate, while adding a set of extensions which carry information related to the hash code of the document, the validity type of optimized certificate and others parameters for verification of the validity certificate sender.

Despite the OC not needing a revocation scheme, it is still necessary to validate the CA certificate that issued it. This CA, called Optimized Certificate Certification Authority (OCCA), uses NOVOMODO [18] to include in all OCs issued an extension that will allow the users to verify its revocation state without consulting the CA that issued its certificate. The OCCA, when issuing an OC, request a proof about validity of its own certificate to the Root CA. If it is valid, the OCCA inserts it into the OC. This means that the OC is inherently valid, because it was verified when it was issued, it possess in its body a proof that the OCCA certificate was valid at the same moment when the OC was created.

The proof of validity of the OCCA certificate is the parameter Y_{n-k} of NOVOMODO that allows the verification of the state of a certificate at instant k.

4 Optimized Certificate Certification Authority

4.1 The OCCA

The OCCA must provide services on-line, once it can receive a request certificate for issuing an OC. Based on this requirement, on the necessity of scalability and the reducing number of certificates in the certification chain, the OCCA is proposed to be subordinated directly to a root CA.

The method NOVOMODO proposed by Silvio Micali [18] eliminates the necessity of using CRL or to consult the CA when the status revocation of a certificate is verified. In this method, the knowledge of few simple parameters can be used to determine if the certificate was valid at one instant of time k. This parameter and other information needed by NOVOMODO are available in OCCA's certificate and issued OCs.

The OCCA certificate is a conventional certificate that contains data related to NOVOMODO in its extensions. The Root CA chooses a hash function F, a period of time t, a resolution l of validity and two random values Y_0 and N_0, keeping these two last private. The resolution of validity l is the interval of time between one revocation and another one and $n = t/l$ the number of time intervals. The Root CA applies n times the value Y_0 to the function F resulting in the value Y_n. N_0 is applied once to F, resulting in N_1. Y_n, N_1 and the hash function F are available in OCCA's certificate through inclusion of an extension named $Novomodo - Ca$.

In addition, there is some information left which is needed for checking the OCCA's certificate through NOVOMODO validation method. It is data extracted from certificate revocation status service (CRS) [18] and it corresponds to parameters Y_{n-k}, n and k - discussed in Sect. 3 - which are available in OCs through inclusion of an extension named $Novomodo - Oc$. Its fields $certificateStatus$, $last$, $instant$ correspond respectively to the parameters Y_{n-k}, n and k. Moreover, there's a field called *digestAlgorithm* which is related to hash function F mentioned later.

In order to verify if the OCCA certificate is valid at the determined time interval k, the value of Y_{n-k} of the CRS must be checked. If $F_k(Y_{n-k}) = Y_n$, where Y_n is obtained from the OCCA certificate, then it is guaranteed that certificate was valid in k. The revocation can be verified using the equation $N' = F(N_0)$, where N_0 must be in the CRS. If $N' = N_1$, where N_1 is obtained from the OCCA certificate, then this certificate has been revoked.

If a determined period of time k does not possess the corresponding value Y_{n-k} in the CRS, OCCA cannot prove that its certificate is valid in this period. The OCCA must verify if N_0 is in the CRS. If this has happened, the OCCA certificate has been revoked. If the OCCA cannot prove that its certificate is valid, it cannot issue new certificates.

If the OCCA does not have proof that its certificate is valid, the OCCA cannot issue OCs.

The OCCA also must presents other necessary functionalities for issuing of OCs. An important field of an OC is its validity. To be able to include the date and time, the OCCA must have its internal clock synchronized with a trustworthy time source. The best known method to do this is the NTP [19]. However, many critics of the impossibility of assuring, with a reasonable cost, that the date and time are really used by the OCCA and that this is in fact synchronized with trustworthy sources of time. To resolve this problem Haber and Stornetta [20] argue that the documents should be interlinked in a chain to confirm that they are dated. This method is known as *relative timestamp.* Thus, the OCCA must link the OCs, and insert an absolute date and time into the OCs.

The interpretation of OC validity depends on the type of requirer, who can be a signer or a verifier of the signature. This information is contained in the extension called *validityType*, which is included in the OC and is shown in Sect. 3.

Figure 3 shows an OPKI with five levels and an OCCA. In this figure CRL_i is the certification revocation list issued by CA_i. CRL_1 is issued by the Root CA. U_i is the signer certificate of user i. v is the validity of the certificate. h is the hash of the document. OC_i is an optimized certificate for the signer certificate U_i. Y_n and Y_{n-ki} are parameters, which latter is obtained from CRS. These parameters are used to assure that the OCCA certificate was valid when the OC was issued. f is the parameter that indicates the validity type of the OC.

In this figure, the numbers 1 indicates the CRLs which should be consulted by the user to verify the CA certificates. 2 represents where information about

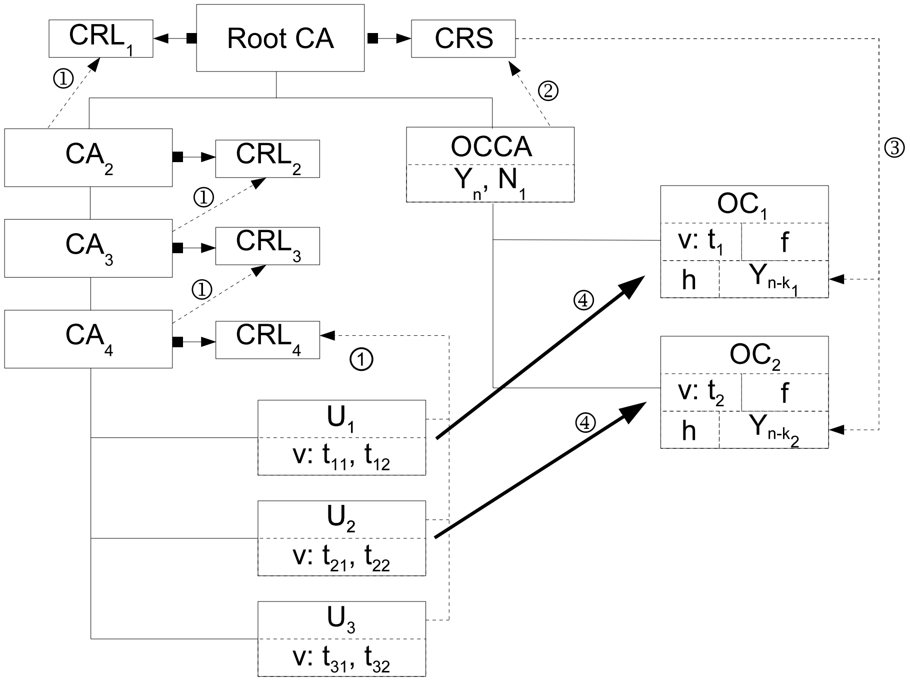


Fig. 3. Optimized PKI

OCCA revocation is published. 3 shows that OCCA certificate status data is obtained from CRS and it is included in OCs. Finally, 4 illustrates the exchange of a traditional certificate to a optimized certificate.

4.2 Attainment of Optimized Certificates

The parameter f of the OC indicates how the field **Validity** must be interpreted for the document to which it is related. This parameter can assume four different values, as is shown in the Table 1. Parameter f is equal to f_1 when the request for a new OC is made by the signer and is equal to f_2, f_3 or f_4 when it is made by a verifier.

When $f = f_1$ this means that the validity of the OC must be interpreted as the date and time when the document was signed. Therefore, only a signer could request an OC with $f = f_1$. The signer sends the document's hash and its classical certificate C to the OCCA. The OCCA, through a challenge-reply protocol, verifies if the requester has ownership of key pairs. If this is confirmed, the OCCA verifies the validity of C, the entire certification chain and the revocation status, and checks if the whole process conforms to certification policies stated. If all the information is valid, the OCCA issues the OC and sends it to the signer. The signer has the responsibility of confirming the OC and attaching it to the document.

Table 1. Parameter f

f	**Interpretation**
f_1	The date and time of the OC correspond to the time when the document was signed and also it is a proof that the document existed at this moment.
f_2	The document signature was valid when the OC was issued and the document already existed at this moment.
f_3	The document signature was valid at the date and time requested by the verifier.
f_4	The document signature was valid at the date and time of the signature's timestamp and it is evidence that the document already existed at this moment.

When $f = f_2$, f_3 or f_4, the request for an OC can be made by the signer or a verifier. The objective of the verifier is to replace the original signer certificate with an OC to optimize future validation of the signature document. In this case, the OCCA is responsible for checking the validity of documents submitted at the instant of time contained in the request.

When $f = f_2$ and f_4, the OCCA initiates the verification procedure at the moment of receipt of the request and when f_3, the request is processed at a moment defined by the user. If f_4, the request contains the TS of the original signature. The OCCA validates the TS and uses this date and time in the *validity* field of the OC.

The OCCAs register all operations carried when the OCs are issued. This permits auditing procedures by a third party. These records are bound to a precise time using NOVOMODO in the same way as shown by Haber and Stornetta [20] to timestamp an electronic document.

The initial value of the chain is the value Y_1. Each issued OC is linked with the previous one keeping all OCs connected in a chain. This chaining is made during the period l, for example, one day. After this period if a new element is added to the chain, the next value will be Y_2. This continues successively until the period n, or when the Root CA publishes N_0.

5 Digital Signature Validation

An OPKI is a PKI that issue optimized certificates. OPKI compares favorably to a traditional PKI in analysis of performance validation processes, as the number of signature validation during the life time of an electronic document is greatly reduced.

In contrast to the signature generation process which occurs only once, the signature verification is normally repeated innumerable times. This validation can be made by different entities at different times. Thus, in evaluation terms, the cost functions of a conventional signature and signatures using OCs are in terms of the computational effort demanded by verifiers and signers.

Most of the processing power in traditional PKI deployments is used to validate digital certificates. With OPKIs, only two additional certificates need to be verified, the OCCA certificate and the OC. The OCCA certificate is verified by the user through the hash function of NOVOMODO, by $F_k(Y_{n-k})$, where k is a specific instant of the lifetime n.

The value Y_n is contained in the OCCA certificate and Y_{n-k} is included the OC, as discussed in Sect. 3 and 4. After simple calculations these can be easily compared. This is sufficient to verify the signature without consulting any additional information. The OCs do not need to be verified about revocation since it's only validity at an instant.

6 Considerations

The signer must validate his digital certificate when he issues the signatures of electronic documents. And, the recipients must validate the signatures each time they need to verify the documents' authenticity. These tasks require many cryptographic signature validation. A reasonable strategy to avoid an overload on the recipients' computers would be to delegate those computations to an external service.

The delegation of the digital certificates validation to a third party entity is not a new idea. Peifang Zheng [21] calls this entity a verifier. The protocol Online Certificate Status Protocol (OCSP) [22] is considered a verifier. OCSP is an Internet protocol used for obtaining the revocation status of a digital certificate. It was created as an alternative to certificate revocation lists (CRL), specifically addressing certain problems associated with using CRLs. However, this protocol cannot be used as evidence that a certificate was valid in the past without additional services like timestamping. The verifier could be used either by the signer or by the recipients to improve the efficiency of certificate and signature validation. Although this strategy may appear attractive, there needs to be at least one server inquiry for each certificate or signature validation.

The use of OCs in electronic documents eliminates the necessity of consulting an external entity for this task. The verification is made by the OCCA, which already is part of the optimized PKI. Each time the OCCA needs to issue a new OC, it requests from the Root CA a proof of the validity of its own certificate, as described in Sect. 4. This proof is inserted into the OC. Then, when the recipient verifies the OCCA certificate, he only uses that proof, without consulting a database or even the CA. Additionally, an OC is issued for a specific date and time, it is valid only for this date and there is no sense in revoking it.

The main advantage in the use of an OC is to get a self-sufficient electronic document, where there is no necessity to consult any external service to validate its signature. The OC itself is a proof that the certification chain was validated by a trust entity.

The fact of the OC was issued with validity only for one k instant gives it the same positive features described by Rivest [16], where a certificate has low probability of revocation soon after its emission. Moreover, the method does not present the problems of short term certificates [23] with generation of new cryptographic keys and certificates each time that the subscriber needs to renew its certificate.

The publication of a CRS involves the use of the Root CA private key to authenticate the values Y_{n-k}. However, this check may be waived, as the authentic values for CRS are kept private by the CA. The maintenance of the signature assures that even discovery of values Y_0 and N_0 is not enough to compromise this scheme.

A positive point of OPKI approach is that even having the OCCA's private key at any instant of l_i, an attacker can only issue false OCs until l_i ends since he can not generate and include in an OC a fake proof of OCCA's certificate validity in l_{i+1}. Therefore, to compromise an OPKI an attacker needs to compromise OCCA's private key and the secret value Y_0, which is stored in root CA, that is likely kept offline.

This means that the recipient does not need to check the Root CA certificate status during signature verification. This is a great advantage of this proposal compared to the classical approach.

In the case of any fraud that should occur, the attacker may use a compromised root CA private key to create a false OCCA. However this problem is easily countered by publication of the hash of the true OCCA certificate in a public directory.

Users should register the trusted OCCAs by taking hashes from the public directory and saving them into their computers before beginning to use the OCAs.

7 Conclusion

This paper proposes optimized digital certificates as a way to efficiently sign and verify signature of electronic documents. Optimized certification and the corresponding subject certificate verification methods improve certificate path verification time and signature in signed electronic documents.

An optimized certificate is a certificate valid for a specific time and date and it can be used to substitute the signer's certificate. Optimized certificates are not revoked as they are valid only for that time and date. These certificates are issued by an optimized certificate certification authority (OCCA) and contain all the information needed to turn the electronic document into a self-verified signed document, i.e., it is not necessary to consult a certification authority or a database to verify the certification chain as in the classical PKI. By using optimized certificates, it is possible to have efficiently verifiable electronic documents.

In this paper, the design of the OPKI has been presented. This incorporates both classical and optimized certificates. The OPKI includes an new optimized certificate certification authority which is a service directly subordinate to the Root CA. The OCCA issues optimized certificates whenever it is request to. Before the OCCA issues the certificate, it verifies the signer's classical certificate.

The optimized certificate is bound to the electronic document through its hash and includes proof of validity of its own certificate. This proof is obtained from the Root CA. The proof is a hash from a chain as specified in the NOVOMODO validation scheme. Both verification efficiency and the revocation advantage of optimized certificates make them suitable for hierarchical PKIs of wireless applications using signed electronic documents where wireless end users have limited bandwidth and processing power.

Next steps will be design and implement a prototype of an OCCA and a formal analysis of its protocols.

References

1. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978)
2. Diffie, W.: The first ten years of public-key cryptography, pp. 510–527 (1988)
3. Zhou, J., Deng, R.: On the validity of digital signatures. SIGCOMM Comput. Commun. Rev. 30(2), 29–34 (2000)
4. Adams, C., Farrell, S., Kause, T., Mononen, T.: Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard) (September 2005)
5. Lloyd, S.: Understanding certification path construction. PKI Forum, 1–14 (September 2002)
6. Russell, S., Dawson, E., Okamoto, E., Lopez, J.: Virtual certificates and synthetic certificates: new paradigms for improving public key validation. Computer Communications 26(16), 1826–1838 (2003)
7. Levi, A., Caglayan, M.U., Koc, C.K.: Use of nested certificates for efficient, dynamic, and trust preserving public key infrastructure. ACM Trans. Inf. Syst. Secur. 7(1), 21–59 (2004)
8. Ferguson, N., Schneier, B.: Practical cryptography. Wiley, Chichester (2003)
9. Perlman, R., Kaufman, C.: Method of issuance and revocation of certificates of authenticity used in public key networks and other systems. Technical report, United State Patent 5,261,002 (1993)
10. Cooper, D.A.: A model of certificate revocation. In: ACSAC 1999: Proceedings of the 15th Annual Computer Security Applications Conference, Washington, DC, USA, pp. 256–264. IEEE Computer Society, Los Alamitos (1999)
11. Micali, S.: Efficient certificate revocation. Technical report, Cambridge, MA, USA (1996)
12. Kocher, P.C.: On certificate revocation and validation. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 172–177. Springer, Heidelberg (1998)
13. Cooper, D.A.: A more efficient use of delta-crls. In: SP 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 190. IEEE Computer Society, Los Alamitos (2000)
14. Naor, M., Nissim, K.: Certificate revocation and certificate update. IEEE Journal on Selected Areas in Communications 18(4), 561–570 (2000)
15. Gassko, I., Gemmell, P., MacKenzie, P.D.: Efficient and fresh cerification. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 342–353. Springer, Heidelberg (2000)
16. Rivest, R.L.: Can we eliminate certificate revocations lists? In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 178–183. Springer, Heidelberg (1998)

17. Adams, C., Lloyd, S.: Understanding PKI: Concepts, Standards, and Deployment Considerations. Addison-Wesley Longman Publishing Co, Boston (2002)
18. Micali, S.: NOVOMODO: Scalable Certificate Validation and Simplified PKI Management. In: Proceedings of the 1st Annual PKI Research Workshop, NIST, Gaithersburg MD, USA (April 2002)
19. Mills, D.: Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard) (March 1992)
20. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 437–455. Springer, Heidelberg (1991)
21. Zheng, P.: Tradeoffs in certificate revocation schemes. SIGCOMM Comput. Commun. Rev. 33(2), 103–112 (2003)
22. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard) (June 1999)
23. McDaniel, P., Rubin, A.D.: A response to can we eliminate certificate revocation lists? In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 245–258. Springer, Heidelberg (2001)

An Efficient and Provable Secure Identity-Based Identification Scheme in the Standard Model

Ji-Jian Chin[1], Swee-Huay Heng[2], and Bok-Min Goi[1]

[1] Center for Cryptography and Information Security (CCIS),
Faculty of Engineering, Multimedia University, Cyberjaya, 63000 Selangor, Malaysia
{jjchin,bmgoi}@mmu.edu.my

[2] Center 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. We present an efficient and provable secure identity-based identification scheme in the standard model. Our proposed scheme is secure against impersonation under passive attack based on the Computational Diffie-Hellman assumption, and secure under active and concurrent attacks based on the One-More Computational Diffie-Hellman assumption.

Keywords: identity-based cryptography, identification scheme, standard model.

1 Introduction

Background. An identification scheme allows a prover to prove himself to a verifier without revealing any information of his private key. For conventional identification schemes, a certificate is used to bind the public key to a user in order to prove the authenticity of a user's ownership of his public key. In 1984, Shamir introduced the concept of identity-based cryptography [18]. The idea was to eliminate public key certificates by using a public key that is bound to a users identity like an identity (ID) string of the entity involved (e.g. email address, telephone number, etc.). A private key generator (PKG) is in charge of computing and distributing the private keys corresponding to the ID from a master secret key.

Shamir proposed an identity-based signature (IBS) in the same paper [18]. However, the first provable secure identity-based encryption (IBE) scheme in the random oracle model was not constructed until Boneh and Franklin in [5] and later by Cocks [8]. Subsequently, several other IBS schemes were constructed in the random oracle model [7,9,15]. Boneh and Boyen in [4] first attempted to construct an IBE to be proven secure in the standard model, but unfortunately it was weaker in security and impractical. Finally, Waters succeeded in constructing a fairly efficient IBE scheme without random oracles [19].

While work in the areas of IBE and IBS schemes flourished and still continues to do so, identity-based identification (IBI) schemes remained untouched. It was not until the year 2004 that IBI was given rigorous definition in two independent papers [1] and [10]. In their work in [10], Kurosawa and Heng proposed transforming any standard digital signature (DS) scheme having 3-move honest verifier zero-knowledge proof of knowledge protocol to an IBI scheme. They also proved that the newly derived IBI scheme is secure against impersonation under passive attack if the underlying DS is existentially unforgeable under adaptive chosen message attack. Taking a different approach from [10], [1] provided security proofs of attacks for a large number of IBI and IBS schemes defined either explicitly or implicitly in existing literature. [1] first provides security of an underlying standard identification scheme before proving the security of IBI or IBS schemes. Also, [1] does not touch IBI and IBS schemes in the standard model.

A random oracle produces a bit-string of infinite length that can be truncated to a desired length. It is used in cryptographic proofs when there are no practical functions that provide sufficient mathematical properties to satisfy the proof of security [2]. Bellare and Rogaway originally proposed the random oracle model to be used in providing proof of security for cryptosystems. Open to be accessed by all parties, good and bad alike, a random oracle acts as an idealistic hash function will return a random response if queried for the first time, and provide the same corresponding response for a repeated query.

Canetti et al. demonstrated, however, there exist certain cryptosystems that while proven secure using the random oracle model, is totally insecure when the oracle is replaced by a conventional hash function [6]. Therefore, although it is still widely accepted that to have a proof in the random oracle model for a cryptosystem is better than to have no proof of security at all, it would still be better to prove cryptosystems secure in the standard model without using random oracles.

Recently, Kurosawa and Heng proposed two IBI schemes in the standard model, one secure against impersonation under passive attack and the other secure against impersonation under active and concurrent attacks [11]. The two schemes were based on the Strong Diffie-Hellman assumption. Another scheme was proposed by the same authors also based on the Strong Diffie-Hellman assumption [12] and [13]. The latest work by Yang et al. was to come up with a general methodology to construct IBI schemes secure against active and concurrent attacks, but the framework is also shown to work in the random oracle model only [20].

Our Contribution. In this paper, we propose an efficient IBI scheme based on the Waters signature scheme [19]. We prove the scheme secure against impersonation under passive attacks based on the intractability of the Computational Diffie-Hellman Problem. We proceed to show the scheme secure against impersonation under active and concurrent attacks based on the intractability of the One-More Computational Diffie-Hellman Problem.

The rest of the paper is organized as follows. We provide some preliminary background information and definitions in Section 2, followed by a review of the definition of IBI schemes in Section 3. In Section 4 we show our scheme construction and verify its correctness. In Section 5, we analyze the efficiency and security of the proposed IBI scheme. We conclude in Section 6.

2 Preliminaries

Let k denote the security parameter. We say that $\varepsilon(k)$ is negligible if $\varepsilon(k)$ approaches zero faster than $\frac{1}{k^c}$ for any constant $c > 0$.

2.1 Bilinear Pairings

Definition 1. *Let G and G_T be finite cyclic groups of prime order p and let g be a generator of G. The map $e : G \times G \rightarrow G_T$ is said to be an admissible map if it satisfies the following three conditions:*

1. *Bilinearity. $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.*
2. *Non-degeneracy. $e(g, g) \neq 1$.*
3. *Efficiently computable.*

2.2 Computational Diffie-Hellman Problem (CDHP)

Let G be a multiplicative cyclic group generated by g with prime order p. The CDHP is described as follows: Given (g, g^a, g^b), calculate g^{ab}.

Definition 2. *We say that the CDHP is (t', ε')-hard in G if $\Pr[A$ calculates $g^{ab}]$ $\leq \varepsilon$ for any A that runs in time t'.*

2.3 One-More Computational Diffie-Hellman Problem (OMCDHP)

The intractability of the strong One-More problems were first introduced in the area of identification schemes in [3] to prove the security of the GQ and Schnorr identification schemes. The OMCDHP was first used to prove pairing-based IBI schemes secure against active and concurrent attacks in the random oracle model in [1] and [10].

Let G be a finite cyclic group of large prime order p and let g be a generator of G. The OMCDHP is described as the following game played by an adversary A. A is a probabilistic polynomial time (PPT) algorithm that is given input $g, g_1 = g^a$ where $a \in \mathbb{Z}_p$ is randomly chosen. A is given access to a challenge oracle $CHALL$ and a computational Diffie-Hellman oracle CDH. $CHALL$ when invoked (without input) will return a random challenge point $W \in G$. CDH when invoked with input $h \in G$ returns $h^a \in G$. To win the game, A must invoke $CHALL$ to get $W_0, \ldots, W_{q_{CDH}} \in G$ challenge points, and output the solutions to all the challenge points $W_0^a, \ldots, W_{q_{CDH}}^a \in G$ by using strictly less than $q_{CDH} + 1$ queries to CDH.

Definition 3. *We say that the OMCDHP is $(t'', q_{CDH}, \varepsilon'')$-hard in G if $\Pr[A$ wins$] \leq \varepsilon''$ for any A that runs in time t'', where q_{CDH} is the total number of queries made to CDH.*

3 Formal Definitions of IBI

3.1 Definition for IBI

An IBI scheme consists of four probabilistic polynomial time algorithms (Setup S, Extract E, Prove P and Verify V)

1. **Setup(S):** S takes in the security parameter 1^k. It publishes the master public key mpk and keeps the master secret key msk to itself.
2. **Extract(E):** E takes in the public identity ID and msk, and returns the corresponding user private key usk.
3. **Identification Protocol(P and V):** P receives mpk, ID and usk as input while V receives mpk and ID. The two will then run an interactive protocol where V will decide whether to accept or reject P after each iteration. The interactive protocol consists of the following steps:
   (a) **Commitment:** P sends a commitment CMT to V.
   (b) **Challenge:** V sends a challenge CHA randomly chosen from a predefined set.
   (c) **Response:** P returns a response RST where V will either choose to accept or reject.

3.2 Security Model for IBI

The goal of an impersonator towards an IBI scheme is impersonation. An impersonator is considered successful if it interacts with the verifier as a cheating prover with public identity ID and is able to be accepted by the verifier with non-negligible probability.

We describe three types of adversaries for IBI schemes:

1. **Passive Attacker.** This type of adversary merely eavesdrops on conversations between an honest prover and verifier and attempts to extract information from their transcripts.
2. **Active Attacker.** This type of adversary interacts with honest provers sequentially as a cheating verifier several times to extract information before the impersonation attempt.
3. **Concurrent Attacker.** This type of adversary is similar to the active attacker with the only exception being it can interact with multiple provers at the same time.

Two differences between security for IBI and conventional identification schemes is that firstly an impersonator can choose a public identity ID of his choice to impersonate as opposed to a random public key. Secondly, it is also assumed that the impersonator already has some private keys of some other users in his possession. This definition allows the impersonator to obtain a private key associated with any identity of his choice besides the one being attacked.

The impersonation attack between an impersonator I and the challenger is described as a two-phased game as follows:

1. **Setup.** The challenger takes 1^k as input and runs setup. The result of system parameters mpk is given to the impersonator I while msk is kept to itself.
2. **Phase 1.** I issues some extract queries ID_i to S. The challenger responds by running the extract algorithm to generate and returns the private key usk, corresponding to the identity ID_i to I. The queries may be asked adaptively. I issues transcript queries for passive attacks or requests to act as a cheating verifier (identification queries) corresponding to some ID_i for active and concurrent attacks.
3. **Phase 2.** Finally, I outputs a challenge identity ($ID \neq ID_i$) which it wishes to impersonate. I can still continue to issue extract and transcript/ identification queries on condition that the challenge identity ID is not queried. I now acts as a cheating prover to convince the verifier based on information gathered in Phase 1 and I wins the game if it is successful in doing so.

Definition 4. *We say an IBI scheme is* (t, q_I, ε)*-secure under passive or active/concurrent attacks if for any passive or active/concurrent impersonator I who runs in time* t*,* $\Pr[I \text{ can impersonate}] < \varepsilon$*, where I can make at most* q_I *extract and transcript/identification queries.*

4 Construction

Let G and G_T be a finite cyclic group of large prime order p and let g be a generator of G. Let $e : G \times G \rightarrow G_T$ be an efficiently computable bilinear map. Use a collision-resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ to hash identities of arbitrary length to a bit string of length n.

1. **Setup:** Randomly select a secret $a \in \mathbb{Z}_p$, a generator $g \in G$, random values g_2, u' and an n -length vector $\langle u \rangle$ whose elements $u_1, \ldots, u_n \in G$. Set $g_1 = g^a$. The master public key msk is $(G, G_T, e, g, g_1, u', \langle u \rangle, H)$. The master secret key usk is g_2^a.
2. **Extract:** Let ID be an n-bit identity string with d_ι denoting the ι^{th} bit of ID. Let $ID \subseteq 1, \ldots, n$ be the set of all ι in which $d_\iota = 1$. Randomly select $r \in \mathbb{Z}_p$ and construct the user secret as
$$usk = (S, R) = (g_2^a(u' \prod_{\iota \in ID} u_\iota)^r, g^r)$$
3. **Identification Protocol:** Prover P and Verifier V do the following:
   (a) P chooses a random $z \in \mathbb{Z}_p$, computes
   $$X = (u' \prod_{\iota \in ID} u_\iota)^z, Y = g_2^z$$
   and sends X, Y, R to V.
   (b) V picks a random challenge $c \in \mathbb{Z}_p$ and sends to P.
   (c) P calculates $Z = S^{z+c}$ and sends Z as a response to V.

(d) V accepts if

$$e(Z,g) = e(Yg_2^c, g_1)e(X(u' \prod_{\iota \in ID} u_\iota)^c, R)$$

To verify the correctness of the identification protocol, we have:

$$\begin{aligned} e(Z,g) &= e((g_2^a(u' \prod_{\iota \in ID} u_\iota)^r)^{(z+c)}, g) \\ &= e((g_2^{a(z+c)}, g)e((u' \prod_{\iota \in ID} u_\iota)^{r(z+c)}, g) \\ &= e((g_2^z g_2^c, g^a)e(((u' \prod_{\iota \in ID} u_\iota)^z (u' \prod_{\iota \in ID} u_\iota)^c), g^r) \\ &= e(Yg_2^c, g_1)e(X(u' \prod_{\iota \in ID} u_\iota)^c, R) \end{aligned}$$

5 Security Analysis

5.1 Security Against Passive Attacks

Theorem 1. *The above IBI scheme is (t, q_I, ε)-secure against impersonation under passive attacks in the standard model if the CDHP is (t', ε')-hard where*

$$t' = t + O(\rho(2n(q_I) + \tau(q_I)), \varepsilon \leq \sqrt{4q_e(n+1)\varepsilon'} + \frac{1}{p}$$

where ρ represents the time taken to do a multiplication in G, τ is the time taken to do an exponentiation in G, q_e represents the number of extract queries made, q_i represents the number of transcript queries made and $q_I = q_e + q_i$.

Proof. Assume there exists an impersonator I who (t, q_I, ε)-breaks the IBI scheme. Then we show that there is an algorithm M who (t', ε')-solves the CDHP with the help of I. In order to do this, we utilize parameters from [16]. M will be given a group G, a generator $g \in G$, and elements $g^a, g^b \in G$. M will then attempt to simulate a challenger for I as follows.

1. *Setup.* M sets $l = 2q_e$ and randomly chooses $k \in \mathbb{Z}_n$. Assume that $l(n+1) < p$ for the given values of q_e and n. Furthermore, M randomly chooses $x' \in Z_l$, a vector $\langle x \rangle$ of length n with $x_\iota \in \mathbb{Z}_l$ for all ι, a randomly selected integer $y' \in \mathbb{Z}_p$ and a vector y of length n with $y_\iota \in \mathbb{Z}_p$ for all ι. Define the following functions:

$$F(ID) = x' + \sum_{\iota \in ID} x_\iota - lk, J(ID) = y' + \sum_{\iota \in ID} y_\iota$$

M now sets $g_1 = g^a$ and $g_2 = g^b$. M also sets $u' = g_2^{x'-lk} g^{y'}$ and a vector $\langle u \rangle$ of length n consisting of elements $u_\iota = g_2^{x_\iota} g^{y_\iota}$. M publishes mpk as

$(g, g_1, g_2, u', \langle u \rangle)$. Note that through the functions $F(ID)$ and $J(ID)$ we have

$$u' \prod_{\iota \in ID} u_\iota = g_2^{F(ID)} g^{J(ID)}$$

2. *Extract Queries.* When I queries M for the private key of identity ID_i, I checks if $F(ID_i) = 0 \bmod l$ and aborts if it is. This is because given the assumption, $l(n+1) < p$ implies $0 \leq lk \leq p$ and $0 \leq x' + \sum_{\iota \in ID_i} x_\iota \leq p$. Therefore $F(ID_i) = 0 \bmod p$ implies that $F(ID_i) = 0 \bmod l$ and the simulator aborts because it is unable to construct a private key. Otherwise, if $F(ID_i) \neq 0 \bmod l$, M constructs the private key by randomly selecting $r_i \in \mathbb{Z}_p$ and computing *usk* as

$$(\widetilde{S_i} = g_1^{\frac{-J(ID_i)}{F(ID_i)}} (u' \prod_{\iota \in ID_i} u_\iota)^{r_i}), (\widetilde{R_i} = g_1^{\frac{-1}{F(ID_i)}} g^{r_i})$$

   To I, all private keys generated by M will be indistinguishable from those generated by a true challenger.
3. *Transcript Queries.* There are two scenarios when I queries M for a round of interaction on ID_j, namely if $F(ID_j) = 0$ mod l or not.We assume without loss of generality that I will not issue an identification query on an identity that it has already issued an extract query on.
   (a) If $F(ID_j) \neq 0 \bmod l$, M can simulate a cheating prover for I by the running same extract algorithm and then using the valid private key to produce a valid transcript for I.
   (b) If $F(ID_j) = 0 \bmod l$, M generates a valid transcript for I by choosing random $r_j, c_j, z_j \in \mathbb{Z}_p$ and sends the transcript to I as:

$$(X = (u' \prod_{\iota \in ID_j} u_\iota)^{z_j}, Y = \frac{g^{z_j}}{g_2^{c_j}}, R = g^{r_j}, c_j, Z = g_1^{z_j} (u' \prod_{\iota \in ID_j} u_\iota)^{r_j(z_j + c_j)})$$

Impersonation phase. After some time, I outputs an identity $ID^* \neq ID_i$ that it wishes to be challenged on. M will abort if $F(ID^*) \neq 0 \bmod p$. I can still issue some extract and transcript queries on the condition that it cannot query the challenge identity ID^*. I now plays the role of the cheating prover trying to convince M that it knows the user secret of ID^*. Thus M is able to obtain two valid transcripts (X, Y, R, c_1, Z_1) and (X, Y, R, c_2, Z_2) by resetting I to where it sent X, Y, R after the first interaction. Based on the Reset Lemma [3], M can extract S from two conversation transcripts with probability more than $(\varepsilon - \frac{1}{p})^2$. M extracts the secret key by calculating $S = (Z_1 Z_2^{-1})^{(c_1 - c_2)^{-1}}$ and outputs the solution to the CDHP by calculating

$$\frac{S}{R^{J(ID^*)}} = \frac{g^{ab}(u' \prod_{\iota \in ID^*} u_\iota)^r}{g^{J(ID^*)r}} = g^{ab}$$

This completes the description of the simulation.

It remains to analyze the probability of M winning the game and solving the CDHP. First off we have the probability that M can extract S from two conversation transcripts as $\Pr[M \text{ computes } g^{ab}|\neg\text{abort}] \geq (\varepsilon - \frac{1}{p})^2$ by the Reset Lemma. Note that upon extraction S, A is able to compute g^{ab}. Therefore the probability of M winning the game by solving the CDHP is given by

$$\begin{aligned}\Pr[M \text{ solves CDHP}] &= \Pr[M \text{ computes } g^{ab} \vee \neg\text{abort}] \\ &= \Pr[M \text{ computes } g^{ab}|\neg\text{abort}]\Pr[\neg\text{abort}] \\ &\geq (\varepsilon - \frac{1}{p})^2 \Pr[\neg\text{abort}]\end{aligned}$$

It remains to calculate $\Pr[\neg\text{abort}]$. We define the following events:

(a) Let Event A_i: M answers all extract queries where $F(ID_i) \neq 0 \bmod l$.
(b) Let Event A^*: I outputs challenge ID^* where $F(ID^*) = 0 \bmod p$.

The probability of event A^* occurring can be calculated by

$$\begin{aligned}\Pr[A^*] &= \Pr[F(ID^*) = 0 \bmod p \vee F(ID^*) = 0 \bmod l] \\ &= \Pr[F(ID^*) = 0 \bmod l]\Pr[F(ID^*) = 0 \bmod p | F(ID^*) = 0 \bmod l] \\ &= \frac{1}{l}(\frac{1}{n+1})\end{aligned}$$

We also have that

$$\begin{aligned}\Pr[\bigcap_{i=1}^{q_e} A_i | A^*] &= 1 - \Pr[\bigcup_{i=1}^{q_e} A_i | A^*] \\ &= 1 - \sum_{i=1}^{q_e} \Pr[\neg A_i | A^*] \\ &= 1 - \frac{q_e}{l}\end{aligned}$$

Therefore the probability of M not aborting is given by

$$\begin{aligned}\Pr[\neg\text{abort}] &= \Pr[\bigcap_{i=1}^{q_e} A_i \wedge A^*] \\ &= \Pr[A^*]\Pr[\bigcap_{i=1}^{q_e} A_i | A^*] \\ &= \frac{1}{l(n+1)}(1 - \frac{q_e}{l}) \\ &= \frac{1}{4q_e(n+1)}\end{aligned}$$

since $l = 2q_e$ as in the simulation. Hence we have

$$\begin{aligned}\Pr[M \text{ solves CDHP}] &\geq (\varepsilon - \frac{1}{p})^2 \frac{1}{4q_e(n+1)} \\ \varepsilon' &\geq (\varepsilon - \frac{1}{p})^2 \frac{1}{4q_e(n+1)} \\ \varepsilon &\leq \sqrt{4q_e(n+1)\varepsilon'} + \frac{1}{p}\end{aligned}$$

The time complexity of M is dominated by exponentiations and multiplications performed in the extract and transcript queries. We can approximate the time complexity of M as $t' = t + O(\rho(2n(q_I)) + \tau(q_I))$ since there are $O(n)$ multiplications and $O(1)$ exponentiations in both the extract and transcript stages, ρ representing the time taken to do a multiplication in G, τ is the time taken to do an exponentiation in G , q_e represents the number of extract queries made, q_i represents the number of transcript queries made and $q_I = q_e + q_i$. □

5.2 Security Against Active and Concurrent Attacks

We prove our proposed IBI scheme secure against active and concurrent attacks using the OMCDHP. Most of the parameters are analogous from the proof against passive attacks.

Theorem 2. *The above IBI scheme is (t, q_I, ε)-secure against impersonation under active and concurrent attacks in the standard model if the OMCDHP is $(t'', q_{CDH}, \varepsilon'')$-hard where*

$$t'' = t + O(\rho(2n(q_I) + \tau(q_I)), \varepsilon \leq \sqrt{4q_e(n+1)\varepsilon''} + \frac{1}{p}$$

where ρ represents the time taken to do a multiplication in G, τ is the time taken to do an exponentiation in G, q_e represents the number of extract queries made, q_i represents the number of identification queries made and $q_I = q_e + q_i$.

Proof. Assume there exists an impersonator I who (t, q_I, ε)-breaks the IBI scheme. Then we show that there is an algorithm M who $(t'', q_{CDH}, \varepsilon'')$-solves the OMCDHP with the help of I. In order to do this, we use some parameters from [16]. M will be given a group G, a generator $g \in G$, an element $g^a \in G$ and access to oracles $CHALL$ and CDH. M will then attempt to simulate a challenger for I as follows.

1. *Setup.* M sets $l = 2q_e$ and randomly chooses $k \in \mathbb{Z}_n$. Assume that $l(n+1) < p$ for the given values of q_e and n. Furthermore, M randomly chooses $x' \in Z_l$, a vector $\langle x \rangle$ of length n with $x_\iota \in \mathbb{Z}_l$ for all ι, a randomly selected integer $y' \in \mathbb{Z}_p$ and a vector y of length n with $y_\iota \in \mathbb{Z}_p$ for all ι. Define the following functions:

$$F(ID) = x' + \sum_{\iota \in ID} x_\iota - lk, J(ID) = y' + \sum_{\iota \in ID} y_\iota$$

M now sets $g_1 = g^a$, queries $CHALL$ for the initial challenge W_0 and sets $g_2 = W_0$. M also sets $u' = g_2^{x'-lk} g^{y'}$ and a vector $\langle u \rangle$ of length n consisting of elements $u_\iota = g_2^{x_\iota} g^{y_\iota}$. M publishes mpk as $(g, g_1, g_2, u', \langle u \rangle)$. Note that through the functions $F(ID)$ and $J(ID)$ we have

$$u' \prod_{\iota \in ID} u_\iota = g_2^{F(ID)} g^{J(ID)}$$

2. *Extract Queries.* M responds to extract queries by I the same way as it does in section 5.1.
3. *Identification Queries.* There are two scenarios when I queries M for a round of interaction on ID_j, namely if $F(ID_j) = 0 \bmod l$ or not. M starts with $m = 1$. We assume without loss of generality that I will not issue an identification query on an identity that it has already issued an extract query on.
   (a) If $F(ID_j) \neq 0 \bmod l$, M can simulate a cheating prover for I by the running same extract algorithm and then using the valid private key to engage in an interaction with I.
   (b) If $F(ID_j) = 0 \bmod l$, M simulates a cheating prover for I as follows
      i. M queries $CHALL$ for a random point W_m and lets $\widetilde{Y_j} = W_m$. M randomly selects $r_j \in \mathbb{Z}_p$ and computes

$$(\widetilde{X_j} = g^{J(ID_j)} = (u' \prod_{\iota \in ID_j} u_\iota), \widetilde{R_j} = g_1^{r_j})$$

      M sends $\widetilde{X_j}, \widetilde{Y_j}$ and $\widetilde{R_j}$ to I.
      ii. I issues a randomly selected challenge $c \in \mathbb{Z}_p$
      iii. M queries CDH with

$$W_m(u' \prod_{\iota \in ID_j} u_\iota)^{r_j} (W_0(u' \prod_{\iota \in ID_j} u_\iota)^{r_j})^{c_j}$$

      and sends I the reply

$$\widetilde{Z_j} = [W_m(u' \prod_{\iota \in ID_j} u_\iota)^{r_j} (W_0(u' \prod_{\iota \in ID_j} u_\iota)^{r_j})^{c_j}]^a$$

   (c) M increments m by 1.

Impersonation phase. After some time, I outputs an identity $ID^* \neq ID_i$ that it wishes to be challenged on. M will abort if $F(ID^*) \neq 0 \bmod p$. I can still issue some extract and identification queries on the condition that it cannot query the challenge identity ID^*. I now plays the role of the cheating prover trying to convince M that it knows the user secret of ID^*. Thus M is able to obtain two valid transcripts (X, Y, R, c_1, Z_1) and (X, Y, R, c_2, Z_2) by resetting I to where it sent X, Y, R after the first interaction. Based on the Reset Lemma [3], M can extract S from two conversation transcripts with probability more than $(\varepsilon - \frac{1}{p})^2$. M extracts the secret key by calculating

$S = (Z_1 Z_2^{-1})^{(c_1 - c_2)^{-1}}$ and outputs the solution to the primary challenge W_0 by calculating

$$\frac{S}{R^{J(ID^*)}} = \frac{W_0^a (u' \prod_{\iota \in ID^*} u_\iota)^r}{g^{J(ID^*)r}} = W_0^a$$

M then proceeds to solve the other m challenge points by calculating

$$\begin{aligned}\frac{Z_j}{g_1^{r_j J(ID_j)(c_j+1)} W_0^{ac_j}} &= \frac{[W_m^a W_0^{ac_j} (u' \prod_{i \in ID^*} u_i)^{ar_j} (u' \prod_{i \in ID^*} u_i)^{ar_j c_j}]}{g_1^{r_j J(ID_j)(c_j+1)} W_0^{ac_j}} \\ &= \frac{[W_m^a W_0^{ac_j} g^{J(ID_j) a r_j (c_j+1)}]}{g_1^{r_j J(ID_j)(c_j+1)} W_0^{ac_j}} \\ &= W_m^a\end{aligned}$$

where j corresponds to the identification queries on ID_j where $F(ID_j) = 0 \bmod l$. Thus M is able to solve $m+1$ challenge points of $CHALL$ with only m queries to CDH. This completes the description of the simulation.

It remains to analyze the probability of M winning the OMCDHP game. First off we have the probability that M can extract S from two conversation transcripts as $\Pr[M \text{ computes } W_0^a | \neg\text{abort}] \geq (\varepsilon - \frac{1}{p})^2$ by the Reset Lemma. Note that upon extraction S, A is able to compute W_0^a. Therefore the probability of M winning the OMCDHP game is given by

$$\begin{aligned}\Pr[M \text{ wins OMCDHP}] &= \Pr[M \text{ computes } W_0^a \vee \neg\text{abort}] \\ &= \Pr[M \text{ computes } W_0^a | \neg\text{abort}] \Pr[\neg\text{abort}] \\ &\geq (\varepsilon - \frac{1}{p})^2 \Pr[\neg\text{abort}]\end{aligned}$$

Following the same arguments in section 5.1, we have

$$\Pr[\neg\text{abort}] = \frac{1}{4q_e(n+1)}$$

Therefore

$$\begin{aligned}\Pr[M \text{ wins OMCDHP}] &\geq (\varepsilon - \frac{1}{p})^2 \frac{1}{4q_e(n+1)} \\ \varepsilon'' &\geq (\varepsilon - \frac{1}{p})^2 \frac{1}{4q_e(n+1)} \\ \varepsilon &\leq \sqrt{4q_e(n+1)\varepsilon''} + \frac{1}{p}\end{aligned}$$

The time complexity of M is similar to that of section 5.1 as well. □

Table 1. Complexity cost for each algorithm in the proposed IBI

	Multiplication	Exponentiation	Pairing
Setup	0	2	0
Extract	Max:n+2, Avg:(n/2)+2	2	0
Prove	Max:n+1, Avg:(n/2)+1	3	0
Verify	Max:n+3, Avg:(n/2)+3	2	3

5.3 Efficiency

Our scheme has a private key size of 2 group elements. The public key and parameters consists of the description of the groups G, G_T, the pairing e and $n+4$ group elements in G. It is possible to reduce the number of group elements needed by utilizing the technique of Naccache [14] and Chatterjee-Sarkar [17].

With the vector $\langle u \rangle$ of length n, the maximum number of group operations, Max, involved in the Waters hash function is $n+1$ (inclusive of operating on u'). Avg follows by dividing $n/2$ to dictate the average number of group operations for the entire set of IDs to be used for the scheme.

Table 2. A Comparison with other IBI schemes in the standard model

	Efficiency ($Prove$ and $Verify$)	imp-pa Assumption	imp-aa/ca Assumption
HKIBI05a[11]	6G,6E,4P	q-SDH	Unknown
HKIBI05b[11]	12G,12E,6P	q-SDH	q-SDH
HKIBI06[12,13]	9G,11E,3P,1SOTSS	q-SDH	q-SDH
Proposed Scheme	(n+4)G,5E,3P	CDH	OMCDH

Legend: G-Group Operations, E-Exponentiations, P-Pairings, SOTSS-Strong One-Time Signature Scheme, imp-pa:passive attack, imp-aa/ca:active/concurrent attack.

We provide the complexity costs for each algorithm in Table 1 and provide a comparison with other existing IBI schemes in the standard model in Table 2. We emphasize on the **Prove** and **Verify** computational costs as they will be iterated whenever users wish to participate in the identification protocol. Based on the analysis in Table 2, the efficiency of our proposed scheme is significantly better than existing schemes that are secure against active and concurrent attacks as it utilizes far lesser pairing and exponentiation operations. We claim our scheme is efficient as the computational cost for a group multiplication is negligible when compared to that of an exponentiation. It is also known that pairings cost even more than exponentiation. Therefore, with reduced exponentiations and pairings in our scheme, even though we have more group multiplications, our scheme is still more efficient when compared to schemes in [11] and [12,13].

Another merit over the other schemes is that we manage to provide a direct security proof to a hard problem rather than just proving it in contrast to the security bound of a digital signature scheme or conventional identification scheme. Thus we are able to derive a more concrete security bound for the newly proposed scheme.

6 Conclusion

We presented an efficient and provable secure IBI scheme based on the Waters signature scheme. The proposed scheme is secure against impersonation under passive attack based on the Computational Diffie-Hellman assumption, and secure under active and concurrent attacks based on the one-more computational Diffie-Hellman assumption. Our construction is shown to be efficient and also the first IBI scheme proven secure in the standard model by direct reduction to a hard problem.

It still remains an open problem to be able to construct a scheme that is provable secure in the standard model against active and concurrent attacks using a weaker assumption like the Computational Diffie-Hellman or Discrete Logarithm assumption.

Acknowledgements

This research was supported by the Malaysian E-science Fund (01-02-01-SF0032).

References

1. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 268–286. Springer, Heidelberg (2004)
2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: 1st ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
3. Bellare, M., Palacio, A.: GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (2002)
4. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
5. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Canetti, R., Goldreich, O., Halevi, S.: The random oracle model, revisited. In: 30th ACM Symposium on Theory of Computing STOC 1998, pp. 209–218. ACM Press, New York (1998)
7. Cha, J.-C., Cheon, J.-H.: An identity-based signature from gap Diffie-Hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2002)
8. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
9. Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)

10. Kurosawa, K., Heng, S.-H.: From digital signature to ID-based identification/signature. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 248–261. Springer, Heidelberg (2004)
11. Kurosawa, K., Heng, S.-H.: Identity-based identification without random oracles. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganá, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 603–613. Springer, Heidelberg (2005)
12. Kurosawa, K., Heng, S.-H.: The power of identification schemes. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 364–377. Springer, Heidelberg (2006)
13. Kurosawa, K., Heng, S.-H.: The power of identification schemes. International Journal of Applied Cryptography 1(1), 60–69 (2008) Inderscience Publishers
14. Naccache, D.: Secure and practical identity-based encryption. Cryptology ePrint Archive, Report (2005), `http://eprint.iacr.org`
15. Paterson, K.G.: ID-based signatures from pairings on elliptic curves. IEEE Communications Letters 38(18), 1025–1026 (2002)
16. Paterson, K.G., Schuldt, J.C.N.: Efficient identity-based signatures secure in the standard model. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006)
17. Sarkar, P., Chatterjee, S.: Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 424–440. Springer, Heidelberg (2006)
18. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 0196, pp. 47–53. Springer, Heidelberg (1985)
19. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
20. Yang, G., Chen, J., Wong, D.S., Deng, X., Wang, D.: A more natural way to construct ID-based identification schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 307–322. Springer, Heidelberg (2007)

Trust-Rated Authentication for Domain-Structured Distributed Systems

Ralph Holz, Heiko Niedermayer, Peter Hauck, and Georg Carle

Wilhelm-Schickard-Institut für Informatik
University of Tübingen, Germany
lastname@informatik.uni-tuebingen.de

Abstract. We present an authentication scheme and new protocol for domain-based scenarios with inter-domain authentication. Our protocol is primarily intended for domain-structured Peer-to-Peer systems but is applicable for any domain scenario where clients from different domains wish to authenticate to each other. To this end, we make use of Trusted Third Parties in the form of Domain Authentication Servers in each domain. These act on behalf of their clients, resulting in a four-party protocol. If there is a secure channel between the Domain Authentication Servers, our protocol can provide secure authentication. To address the case where domains do not have a secure channel between them, we extend our scheme with the concept of trust-rating. Domain Authentication Servers signal security-relevant information to their clients (pre-existing secure channel or not, trust, . . .). The clients evaluate this information to decide if it fits the security requirements of their application.

Keywords: Authentication, Protocols, PKI, Trust-Rating, Multi-Domain, Distributed Systems, Peer-to-Peer.

1 Motivation

Emerging technologies like Peer-to-Peer networks or Identity Federation have revived interest in concepts for distributed authentication. Identity Federation is a use case where authentication has to be conducted across domain boundaries, i.e. members of one domain wishing to authenticate to members of another domain. Peer-to-Peer networks are often formed in an ad-hoc manner and are decentral by definition. Thus, these networks exhibit the need for distributed authentication even more as no *a priori* context between peers may exist.

According to the work of Boyd [1], however, authentication protocols need at least one secure channel in order to be safe against a Dolev-Yao intruder [2]. Boyd proved that where *'a principal has, at some state of the system, established a secure channel with another principal, such a channel must already have existed at all previous states'* [1]. This means that principals cannot establish an authenticated session without having established keys previously in a secure manner. This can only be circumvented by introducing a Trusted Third Party

(TTP) which can mediate between the principals (e.g. by certifying their keys) – which effectively introduces secure channels by other means.

Where security is a requirement, Peer-to-Peer networks typically rely on a central entity that acts as a TTP, often as a Certification Authority (CA). Skype [3] uses such an approach. However, such a central component ('server') conflicts with the Peer-to-Peer paradigm. While practical for a single network with one administrative domain, it would probably be utopian to assume a single global entity for all Peer-to-Peer networks and, if the networks are to inter-operate, their peers. Moreover, Ellison and Schneier point out that CAs might imply more security than they actually achieve [4], in particular if a single CA is responsible for identity verification on a global scale. Given its limited resources, it may not verify identities carefully enough as a consequence.

Another option that Peer-to-Peer networks may rely on are Web of Trust-like certification chains. However, these are susceptible to the Sybil Attack [5]. The concept can be strengthened by reputation mechanisms such as those for PGP [6] or as described in [7]. While arguably suitable for low-risk use cases, this remains unsatisfactory in general.

It is an interesting observation that most of these solutions may fit well for small Peer-to-Peer networks but become unrealistic for large scenarios (do not scale, cannot verify all identities, etc.). We consider a domain-based Peer-to-Peer network to be a good compromise - a large network consisting of easier-to-secure small networks (each being one domain). Note that such domains may also reflect social and trust relations. This is similar to so-called darknets where peers only connect to their human friends, thus avoiding direct contact with attackers.

Yet, similar to Identity Federation, domain-based solutions still need inter-domain authentication. Of course, Boyd's theorem applies to inter-domain communication as well. Most protocols for inter-domain authentication thus assume the existence of a secure channel to provide their service. The drawback is that they do not cover the use case of domains without previous knowledge of each other. This dilemma can only be mediated, but never resolved completely.

Our contribution is an authentication scheme that explicitly addresses this situation. We present a 4-party authentication protocol for domain scenarios, and extend it with a *Trust-Rating Mechanism*. The novelty in our scheme is that it covers the case of non-secure channels or untrusted systems by supplying the authenticating principals with means to assess the situation and make a well-founded authentication decision. Terminology-wise, our scheme can also be viewed as a PKI for domain-structured systems without *a priori* knowledge of each other. We have implemented a prototype, but put the focus on the concepts in this paper.

This paper is organized as follows. We describe the scheme and the underlying ideas in Section 2. Section 3 contains the formal description of protocol and protocol goals for the scheme. We verify that the goals are achieved in Section 4 and discuss protocol security in Section 5. Section 6 discusses further aspects. Related work is presented in Section 7.

2 Authentication Scheme with Trust-Rating

2.1 Domain Authentication Servers and Trust Tokens

In the following, we use the term ‘domain’ generically to indicate a group of principals that share a common context and desire communication with principals in other domains to be authenticated.

We introduce TTPs in each domain, which we call Domain Authentication Servers (DAS). We structure communication in our scheme by distinguishing between intra-domain and inter-domain communication. The latter takes place on one of two channels: either between two DAS or between two clients. An authentication process between two principals B and A includes activity on the part of their DAS, S_B and S_A. If the channel between S_A and S_B is secure, our protocol (see Section 3) will provide secure authentication, with part of the process delegated to the DAS.

This scheme is now extended with the Trust-Rating mechanism, which is somewhat orthogonal to the authentication. The idea is that the DAS, at one point of the protocol, also communicate their knowledge about the inter-domain channel and about the other domain to their clients. This takes the form of a *Trust Token* that the DAS sends to its client.

We explain this by example. Let us assume an authentication process between B in domain D_B and A in D_A. Let S_B be the DAS for B and S_A the DAS for A. The information that S_B can pass to B can be described with the following two categories. First, whether there exists a secure channel with S_A. Second, collected ‘knowledge’ about S_A, D_A and A itself.

The first is a simple yes/no statement. Where the DAS of two domains have exchanged their keys in a secure manner, the *Trust Token* will signal to the client that there is a secure channel. The protocol flow ensures that secure authentication is possible in this case. Where the *Trust Token* signals no such secure channel (and the DAS have to exchange their keys on an insecure channel), the decision whether to proceed remains with the client principals[1]. For this case, we have information of the second category.

Knowledge about S_A is delivered in the form of a set of properties, for example information about the channel on which the key exchange between S_B and S_A has taken place, or recent information from observations, e.g. whether S_A has acted faithfully so far or whether there are negative reports from other client principals (feedback). The only requirement is that there is a domain-specific configuration that defines the set of properties which both B and S_B must be able to interpret. The software may evaluate the *Trust Tokens* automatically or delegate it to a human user (which makes sense in interactive settings). Information about D_A may take a similar form, e.g. whether members of D_A have been known to have conducted fraud etc. The evaluation is again a domain-specific process. If S_A has not been acting faithfully, this will usually mean that the description of D_A will change, too. Information about the responder peer, A, is given in the same style.

[1] The DAS could theoretically also stop the authentication process, of course.

An example of a Trust Token may thus look like this:

SecureChannel	no
KeyExchange	SinglePathLookup
OtherDomain	SelfCertifyingID: no
OtherDomain	PriorContacts: yes(10)
OtherDomain	KnownFrauds: no(0)
OtherDomain	HumanFeedback: yes(3)/no(0)
OtherPeer	NoInformation

Upon receiving it, a client will know that the DAS have looked up each other via a standard look-up without further hardening and exchanged their keys during this process. The other DAS can also not prove its identity as it is not bound to a public/private key pair. However, there have been 10 prior contacts with that domain before, and no frauds reported yet by other clients. The DAS received feedback for the other domain from 3 clients, all positive and based on human interaction, e.g. from voice sessions like in Zfone [8] (see Section 7). There is no information about the other peer.

A final remark: Note that even in the case of a secure channel and successful authentication of some entity A, this does not imply that A will necessarily be honest and act faithfully – merely that its identity has been ascertained. Information of the second category can thus also be useful in the case where the Trust Token indicates a *secure* channel between S_A and S_B.

2.2 Strengthening the Insecure Channel

Although not the primary focus of this paper, it is worthwhile to observe that there are measures to address the situation where DAS have no secure channel. The methods we propose are the following.

First, in the Peer-to-Peer case, we propose to strengthen the Peer-to-Peer system against network-related attacks. The other domain is given as an identifier within the system (a name, a number, etc.). The Peer-to-Peer system is used to look-up the other domain and its DAS. An attacker could set up a man-in-the-middle attack. To strengthen the communication, the DAS may attempt to communicate over multiple paths in the hope that a man-in-the-middle cannot control all of the paths. The multiple path aspects can be divided into two mechanisms: a) storage of location information at multiple places, b) multiple independent paths towards a target. Furthermore, we need to ensure that the paths and the corresponding peers are diverse.

Second, where possible, we suggest self-certifying IDs for the DAS to avoid man-in-the-middle attacks and impersonation. A self-certifying identifier is e.g. $ID = hash(PublicKey)$. With the knowledge of the private key a peer can prove ownership of the ID. This is quite useful and does not need a central authority, but it does not solve all problems. We can bind an ID to an entity with a key, but we cannot bind a name to an ID. If we need to resolve a human-readable or application-specific name, we still have no guarantee to reach the correct domain.

Third, where possible, we suggest to use user feedback to the DAS to report errors in authentication and misbehavior (although this can also be a flaw if users purposefully inject false information). Only user and application are able to judge if the authentication decision of an insecure session was correct or not. Furthermore, this is also true for the behavior of other entities.

None of these methods can solve the underlying dilemma, but the DAS can include information about which methods were used in the *Trust Token* to aid the client to some degree. This does not provide us with a secure channel, but rather acts as means of a risk assessment.

2.3 PKI Aspects

From the perspective of key distribution, our scheme represents a PKI with the following properties. It is distributed over several domains, yet can operate without a single global authority. This makes it more flexible and eliminates the need for manual interaction, yet allows manual setup of secure channels between domains and leverages trust evaluation. We argue that our concept offers better scalability as only domains that need to communicate are involved in the authentication scheme.

3 Authentication Protocol with Trust-Rating

In the following, we give a formal description of the protocol for our authentication scheme. We begin with the notion of authentication and state the protocol goals against this background.

3.1 Defining Authentication

There are several definitions for authentication in academic literature. We use the relatively strong definition by Lowe [9], 'Injective Agreement', against which we have defined our protocol. The Model Checker that we used to verify and check our protocol with also operates on this definition. We cite it here in the form of Fresh Injective Agreement:

Definition 1. *We say that a protocol guarantees to an initiator A Agreement with a responder B on a set of data items ds if, whenever A (acting as initiator) completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was acting as responder in his run, and the two agents agreed on the data values corresponding to all the variables in ds, and each such run of A corresponds to a unique run of B. Fresh Agreement means that if any initiator A completes a run of the protocol, apparently with B, using particular values for the nonces, then A can be sure that at some time in the past, B believed that he was acting as responder in a run of the protocol with A, using the same values for the nonces.*

3.2 Protocol Goals

We now state the goals that our protocol must achieve. There are several goals in addition to Agreement.

Goal 1: Authentication as Agreement. We specify this as Fresh Injective Agreement. The principals which authenticate each other are A and B. Authentication must be mutual.

Goal 2: Key Establishment. As an outcome of the protocol, the principals A and B establish a session key. We insist that the DAS must not be able to obtain or derive this key.

Goal 3: Freshness of the Session Key. A and B must be able to verify that the session key is a new one, and not the result of an earlier protocol run.

Goal 4: Mutual Belief in Session Key. In the style of Boyd [10], we call a session key K that is only known to A and B (Goal 2) and is fresh (Goal 3) a *good key.* Mutual Belief in K is achieved if and only if B believes K is a good key for use with A and A also believes K to be a good key for use with B.

Optional Goal: Perfect Forward Secrecy. We include Perfect Forward Secrecy as an optional goal.

3.3 Notation

We use the same notation as in [10]. The identity of a principal is denoted by a capital letter. An arrow indicates the process of one principal sending a message to another. A symmetric key is denoted by the letter K with an index indicating between which principals the key is shared. We denote a public key as PK_X, where X indicates the principal to which the key belongs. Encryption with a symmetric key is written as $\{m\}_K$. Encryption with a public key is denoted by $E_X(m)$. A signature with a private key is denoted by $Sig_X(t)$: token t, signed with the private key of agent X. Nonces are denoted by a capital N with an index.

We assume that all messages are integrity-protected without explicitly adding this to our notation.

3.4 Assumptions

We require clients and DAS to execute certain actions when a client becomes a member of a domain. A client principal X and its DAS S_X must agree on a secret symmetric key K_{XS_X} when X joins. X and S_X also exchange their public keys. Both is assumed to happen securely. S_X uses the cryptographically secure hash function h to calculate $h(X.PK_X)$ and signs this with its private key. $Sig_{S_X}(h(X, PK_X))$ will be referred to as a *Public Key Token.*

Two client principals B and A can later exchange their public keys through the following message exchange:

$$B \rightarrow A: \ (B, A, PK_B, Sig_{S_B}(h(B, PK_B))) \qquad \text{(Key Exchange Query)}$$
$$A \rightarrow B: \ (A, B, PK_A, Sig_{S_A}(h(A, PK_A))) \qquad \text{(Key Exchange Response)}$$

Note that this does not achieve key authentication. If DAS exchange their keys, this corresponds to exchanging self-certified keys.

3.5 Protocol Specification

We describe our protocol and motivate each message field.

In general, we take care to follow the guidelines by Abadi and Needham [11] wherever possible. Note that we encrypt every message, which adds to security.

Step 1: Requesting a Credential. We let B from domain D_B initiate the authentication run. The first step is to acquire a *Credential* from B's Domain Authentication Server, S_B, to use for authentication with A in domain D_A.

$$B \rightarrow S_B: \ \{B, S_B, A, PK_A, Sig_{S_A}(h(A, PK_A)), N_B\}_{K_{BS_B}} \qquad \text{(Message 1)}$$

It is our protocol convention to indicate sender and receiver in the first two message fields. The message then states the responder (A), her public key (PK_A) and the *Public Key Token* from a previous key exchange. B refers to this authentication run by the nonce N_B, used for freshness and to avoid that principals mistake a message from one run for a message from a different run.

Step 2: Granting the Credential. If S_A's public key is known, S_B can verify the *Public Key Token* and thus ensure that B uses the correct public key to communicate with A. In addition, as described in Section 2, S_B's answer depends on its knowledge about the channel to S_A, S_A itself, the domain D_A and A itself. S_B creates a *Trust Token* containing at least the following information:

1. Whether S_B is in possession of the correct public key for S_A (secure channel). If the key is not known: through which operation S_B can acquire S_A's public key.
2. Information about S_A, D_A and A that is supposed to help B in estimating whether S_A and A will be 'honest' and act faithfully. These are not numerical values but string descriptions that S_B and B can interpret. Such knowledge may also be derived from previous encounters.

S_B creates a *Credential* for B to use with A. It sends the *Credential* together with the *Trust Token*:

$$S_B \rightarrow B: \ \{S_B, B, Sig_{S_B}(h(B, A, PK_B, N_{S_B})), N_B, N_{S_B}, trust_{D_A}\}_{K_{BS_B}} \qquad \text{(Message 2)}$$

The *Credential* in the third field binds and signs B's identity, A's identity, B's public key and a nonce that S_B has freshly generated for this run, N_{S_B}. This is crucial to ensure that principals cannot mistake a message from one run for a message from another. Also note that the *Trust Token* is not part of the *Credential.* From this point on, B and S_B can identify this conversation by the tuple (N_B, N_{S_B}).

Step 3: Forwarding the Credential to A. If B is satisfied with the information he obtains from the *Trust Token*, he may initiate the conversation with A. B sends:

$$B \rightarrow A: \; E_A(B, A, S_B, Sig_{S_B}(h(B, A, PK_B, N_{S_B})), N_B, N_{S_B}) \qquad \text{(Message 3)}$$

Note that this is a mere forwarding action. B indicates its responsible DAS; the nonces serve to counter replays.

Step 4: Forwarding the Credential to S_A. A cannot verify the *Credential* (because she does not have S_B's public key), thus she forwards it with necessary additional information to S_A.

$$A \rightarrow S_A: \; \{A, S_A, B, S_B, PK_B, Sig_{S_B}(h(B, A, PK_B, N_{S_B})), N_{S_B}, N_A\}_{K_{AS_A}} \qquad \text{(Message 4)}$$

B, S_B, PK_B and N_{S_B} are forwarded in order for S_A to be able to verify the *Credential.* From this point on, A refers to her conversation with S_A by nonce N_A.

Step 5: Verifying Freshness. If S_A is in possession of S_B's public key, it can immediately verify the *Credential.* Else it must acquire the public key by other, possibly insecure, means. Either will be indicated in the *Trust Token* for A later.

With the *Credential* verified, two issues remain. The first is the freshness of the *Credential.* This can only be verified with another message exchange. The second issue is that A needs a token that enables her to authenticate to B. Thus, S_A sends

$$S_A \rightarrow S_B: \; E_{S_B}(S_A, S_B, B, A, N_{S_B}, N_{S_A}) \qquad \text{(Message 5)}$$

S_A indicates the initiator B (field 3) and the responder (field 4). It uses information from the *Credential* to verify that S_B has indeed participated in this authentication run. Note that the triplet (B, A, N_{S_B}) is enough for this purpose. S_A also adds a nonce N_{S_A} by which it will refer to this conversation from now on.

Step 6: Freshness and Credential for A. S_B can identify the authentication run through the triplet (B, A, N_{S_B}). In order to answer the freshness query, it responds with nonce N_{S_A} plus an *Authentication Token* for A to use with B: $Sig_{S_B}(h(A, N_{S_B}))$. This binds A's identity to nonce N_{S_B}, which is known to B.

$$S_B \rightarrow S_A: \; E_{S_A}(S_B, S_A, Sig_{S_B}(h(A, N_{S_B})), N_{S_A}) \qquad \text{(Message 6)}$$

Step 7: Authentication Decision. S_A can verify the *Authentication Token*, and it has now verified that the nonce N_{S_B} refers to a fresh authentication run. S_A can now create a *Trust Token* for A, just as S_B did for B in the other domain.

$$S_A \rightarrow A: \ \{S_A, A, Sig_{S_B}(h(A, N_{S_B})), N_A, trust_{D_B}\}_{K_{AS_A}} \qquad \text{(Message 7)}$$

Step 8: Authentication Response. A can evaluate the *Trust Token* to decide whether to continue.

If A decides to proceed, she generates a new session key K_{AB} and sends:

$$A \rightarrow B: \ E_B(A, B, Sig_{S_B}(h(A, N_{S_B})), N_B, K_{AB}) \qquad \text{(Message 8)}$$

Note that A can safely use B's public key which it now knows to be the correct one. This establishes a secure channel between A and B.

B is in possession of S_B's public key and can thus verify A's *Authentication Token*. This completes the authentication.

3.6 Perfect Forward Secrecy

It is straight-forward to enable Perfect Forward Secrecy with a Diffie-Hellman Key Exchange. We replace K_{AB} with Diffie-Hellman values:

$$A \rightarrow B: \ E_B(A, B, Sig_{S_B}(h(A, N_{S_B})), N_B, g_A, p, DH_A) \qquad \text{(Message 8)}$$
$$B \rightarrow A: \ E_A(B, A, N_B, DH_B) \qquad \text{(Message 9)}$$

4 Verification of Protocol Goals

In the following, we verify that each of the protocol goals is achieved by our protocol.

Goal 1: Trust-rated Fresh Injective Agreement. There are secure channels between the DAS and their clients because we may assume that the respective shared keys are not compromised.

We must distinguish between the communication between S_A and S_B and the communication between A and B. The communication between A and B is clearly a Dolev-Yao channel and it is not secure until both principals have verified the authenticity of the public keys. The channel between S_A and S_B, however, is a secure channel if the DAS have been able to verify the authenticity of each other's public keys. Otherwise it is also a normal Dolev-Yao channel, and insecure. We can therefore simplify the discussion: either the channel is secure or it is not. There is no need to discuss the latter case as there is no way to achieve secure authentication. Thus it is sufficient to examine the case of a secure channel between S_A and S_B.

We examine the case of B wishing to authenticate to A. In Step 2 of the protocol, S_B generates a *Credential* for B by creating and signing $h(B, A, PK_B, N_{S_B})$. The hash function binds these four values together. The signature Sig_{S_B} can only be created by S_B (because only S_B is in possession of the necessary private key). When S_A receives this token in Step 4 and verifies the signature, it can be sure of the fact that B wishes to authenticate to A, that B is a member of D_B, that B has public key PK_B and that S_B refers to this authentication run by the nonce N_{S_B}. Note that all values that S_A needs to calculate and check the hash value are included in the message. The only thing that S_A still has to check is the freshness of the nonce N_{S_B}.

This happens in Steps 5 and 6. S_A queries S_B by referring to the tuple (B, A, N_{SB}). S_A shows that it knows the value of N_{S_B} and is referring to the correct authentication run. It also sends the identity of A, thus assuring S_B that A is a member of D_A. In Step 6, S_B answers S_A's request by replying with the *Authentication Token* $Sig_{S_B}(h(A, N_{S_B}))$. This token binds the identity of A to N_{SB}, and S_A knows that N_{SB} is fresh. After Step 6, S_A can be sure of B's identity and authenticity. It forwards this authentication information to A in Step 7, and A can accept (forward the *Authentication Token*) or refuse (stop there).

We examine the authentication of A to B. S_B creates the *Authentication Token* $Sig_{S_B}(h(A, N_{S_B}))$ for A to authenticate to B. When A receives the *Authentication Token* in Step 7 and forwards it to B in Step 8, B can be assured of A's identity. B can verify S_B's signature and knows that S_B must have created the *Authentication Token* – and thus that S_B believes the information about A to be correct.

We compare this with the definition of Agreement. The set *ds* of values that the principals need to agree on are exactly the *Credential* and the *Authentication Token*: $Sig_{S_B}(h(B, A, PK_B, N_{S_B}))$ and $Sig_{S_B}(h(A, N_{S_B}))$. The principals obviously agree on these values if and only if the protocol has been completed successfully. The condition of injectivity holds as well. When B completes his run as initiator, apparently with A, then A was acting as responder in her run, apparently with B. All principals refer to each authentication run with a nonce of their own, plus their peering principal's nonces in their replies. Thus it cannot happen that a principal mistakes a message to be from a different run, or vice versa. Thus each run of B corresponds to a unique run of A. This is Agreement. It is *trust-rated* due to the *Trust Tokens.*

For the *Freshness* property, we observe that all principals create and maintain their own nonces by which they refer to a combination of channel and run. Thus all entities can be sure that the messages they receive are fresh ones and contain fresh values.

Goal 2: Key Establishment. In message 8, A creates and sends a session key K_{AB} to B.

Goal 3: Freshness of the Session Key. In message 8, A replies with nonce N_B and a new session key.

Goal 4: Mutual Belief. A generates K_{AB} for the purpose of communicating with B. A knows that it is a good key. It is obviously fresh, and it can only be known to A and B because A encrypts it with PK_B, which A knows to be B's public key (thanks to S_A checking S_B's signature). If the authentication run was successful, B can also be sure that the session key is valid and fresh. Since A has sent the session key, B can deduce that the session key must be valid for this communication. Thus it must be a good key. In other words, B believes K_{AB} to be a good key for communication with A, and A believes K_{AB} to be a good key for communication with B. This is Mutual Belief.

Goal 6: Perfect Forward Secrecy. Perfect Forward Secrecy can be achieved with the Diffie-Hellman Key Exchange described above. The channel between A and B is secure because A's and B's public keys have been verified by their respective DAS.

5 Discussion of Security

Our security model is the Dolev-Yao model ([2]), generally considered to represent the strongest possible attacker (only limited by the cryptographic primitives). We evaluated the security of our protocol with the AVISPA Model Checker [12] (OFMC backend).

We will now describe how the specification was modeled in AVISPA, which also uses the Dolev-Yao model. Information can be passed in two ways in AVISPA. The first is as a constant (environment parameter). This value cannot be interfered with, in contrast to the second method, which is to pass information to a principal during a run (a 'variable').

The (symmetric and asymmetric) keys between the clients and their DAS plus the *Public Key Tokens* are thus passed as environment parameters. To model the secure channel between S_A and S_B, the respective public keys are also passed as environment parameters. A and B learn each other's *Public Key Tokens* during a message exchange, thus such a principal's *Public Key Token* is always a variable. The same applies to *Trust Tokens*.

The intruder I may try to impersonate any principal but he does not know their secret or private keys. I is allowed to be a member of both D_A and D_B, i.e. I has established K_{IS_A} and K_{IS_B} with the DAS and has received *Public Key Tokens*. The intruder is allowed to participate in protocol runs (under his own identity) or be a non-participant, or both, at the same time.

The protocol goals are modelled as follows:

1. Authentication as Injective Agreement can be modelled directly in AVISPA. This is applied to *Credential*, *Authentication Token*, *Trust Tokens* and K_{AB}.
2. We model Freshness by having each principal create a fresh nonce for each conversation with another principal and checking the value of this nonce when the reply arrives. We require Agreement on the value of the nonce between the principals using it.
3. We define the goal 'secrecy' for K_{AS_A}, K_{BS_B}, K_{AB}, the private keys and the *Trust Tokens* – i.e. these values must remain unknown to the intruder.

We evaluated all possible combinations of A, B and I interacting. However, due to the 'explosion of the state space', our evaluation was limited to three concurrent sessions. The evaluation showed our protocol to be secure for these scenarios. We also found during our verification that runs with three parallel sessions did not find any new attacks compared to just two parallel sessions.

A formal proof for arbitrarily-sized systems is difficult to provide and remains future work (just as for many other cryptographic protocols). One possibility to extend the model checking results could be Lowe's 'Completeness Checking' [13], which provides a proof that under certain constraints a protocol, which is secure for a small system, is also secure for arbitrarily-sized systems. However, this theory only covers secrecy and not Agreement. A different approach would be to employ other model checkers like Scyther [14], which can also deal with unbounded scenarios. However, Scyther in its current form is geared towards checking authentication as Non-Injective Synchronization [14] and does not cover Injective Agreement. The definition of Synchronization is subtly different from Agreement.

6 Further Considerations

Our scheme exhibits a few properties that are worthy of some consideration.

For example, we expect our scheme to scale reasonably well: it effectively 'decouples' domain activities. The DAS only need to know the keys of their client principals and those DAS that they have already communicated with. There is no need for *a priori* key verification, and there are no certification chains to follow. It is thus easy to introduce new domains. Where this results in an insecure channel between domains, our scheme remains flexible: clients can make an informed decision in such situations.

Key revocation can be easily added, without the need to distribute Key Revocation Lists. When a client revokes a key after *Public Key Tokens* have been exchanged, the DAS will refuse to create *Authentication Tokens*. Keys can also be updated easily: peers send a new key to their DAS (via a domain-internal protocol) and the DAS responds with a new *Public Key Token*.

One can also observe that the scheme provides a rudimentary defense against Denial-of-Service attacks. The DAS in the responder's domain D_A, for example, only becomes active if it receives a valid message from A. There is no way that other principals can trigger a computationally expensive answer, especially if they're from outside D_A. The protection for the initiator's S_B, meanwhile, lies in the fact that it can drop messages from outside its domain that do not include one of the nonces that S_B has recently created for an authentication run.

Our scheme could be extended to enable DAS load distribution within a domain: several DAS could share their (symmetric and asymmetric) keys. An anycast mechanism, possibly with proximity property, would enable clients to address their DAS.

7 Related Work

X.509 and the PGP/GPG Web of Trust are probably the best-known PKIs. X.509 uses CAs, and some argue that this concept offers the highest degree of security that is possible today. Proponents of Webs of Trust counter that a CA is not *per se* trustworthy and that the user can determine better whom to trust. A problem of PKIs is key revocation. This is usually handled with Key Servers and Key Revocation Lists, requiring great diligence. We do not intend to take sides here but rather point out that our concept establishes itself between hierarchical PKIs with CAs and 'flat' Webs of Trust.

The Kerberos protocol [15] can be used for authentication between domains. The foreign Ticket Granting Server must be registered with the local Key Authentication Server. The concept is transitive but requires manual configuration, resulting in higher maintenance.

Protocols for Identity Federation (IF) (e.g. [16]) extend this with the notion of 'Circles of Trust'. They enable the portability of an identity between domains. Keys must be exchanged *a priori* and out-of-band. Many IF protocols additionally use TLS in the underlay, thus establishing secure channels over which the actual authentication protocols are defined. Our scheme is different in both regards.

Other approaches try to eliminate the CA by distributing its functionality, e.g. by using Multiparty Computation (MPC) as presented by Narasimha and Saxena et al. in [17], [18]. They describe a decentral method for group membership control. The scheme requires all principals to execute the protocol faithfully and does not aim to counter the Sybil Attack. If intruders gain access to the group, the scheme is compromised. Defenses can be established with Verifiable Secret Sharing Schemes, e.g. as in [19], but this requires either a high number of broadcast operations or needs to introduce a CA.

If there is no secure channel yet, authentication cannot be secure. The Resurrecting Duckling [20] model assumes that the first contact is not compromised by an intruder and establishes a context for future contacts. The Voice-over-IP system Zfone [8] extends this approach by taking measures against a man-in-the-middle attack on first contact: it requires the (human) users to initiate their conversation by reading a 'Short Authentication String' (SAS) to each other that is derived from the exchanged Diffie-Hellman values.

Concerning trust, Maurer presented a model for reasoning about trustworthiness in a PKI context in [21]. Similar methods could be developed for a client's evaluation of a *Trust Token*.

8 Conclusion

We have presented a four-party authentication scheme and a new protocol for domain-based scenarios. The protocol is an attempt to design an authentication process that is largely decentralized yet remains efficient. We have evaluated its security with model checking. A rigorous proof remains future work.

Since the applications we have in mind include use cases where no prior knowledge exists, we have introduced the concept of trust-rated authentication. Trust-rated authentication uses a *Trust Token* to signal important information (pre-existing secure channel or not, trust in other server, trust in its peers, . . .)

from the authentication server to its client. The client can then decide depending on its current security requirements.

Our overall scheme positions itself somewhere between hierarchical PKIs and Webs of Trust: principals trust distinguished principals in their domain.

Acknowledgments. This work was in part funded by Landesstiftung Baden-Württemberg in the scope of the Spontaneous Virtual Networks (SpoVNet) project.

References

1. Boyd, C.: Security architecture using formal methods. IEEE Journal on Selected Topics in Communications 11, 694–701 (1993)
2. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–208 (1983)
3. Skype Ltd.: Skype (homepage) (February 2008), `http://www.skype.com`
4. Ellison, C., Schneier, B.: Ten risks of PKI: What you're not being told about public key infrastructure. Computer Security Journal 16(1), 1–7 (2000)
5. Douceur, J.R.: The Sybil Attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002) (Revised Papers)
6. Zimmermann, P.R.: The official PGP user's guide. MIT Press, Cambridge (1995)
7. Jøsang, A.: An algebra for assessing trust in certification chains. In: Proceedings of the Network and Distributed Systems Security Symposium (NDSS 1999), Internet Society (1999)
8. The Zfone Project: Zfone (homepage) (2007), `http://zfoneproject.com`
9. Lowe, G.: A hierarchy of authentication specifications. In: Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW 1997), Rockport, MA, USA (1997)
10. Boyd, C., Mathuria, A.: Protocols for authentication and key establishment. Information Security and Cryptography. Springer, Heidelberg (2003)
11. Abadi, M., Needham, R.M.: Prudent engineering practice for cryptographic protocols. IEEE Transactions on Software Engineering 22(1), 6–15 (1996)
12. The AVISPA Project: Automated Validation of Internet Security Protocols and Applications (homepage) (January 2008), `http://www.avispa-project.org/`
13. Lowe, G.: Towards a completeness result for model checking of security protocols. Journal of Computer Security 7(2), 89–146 (1999)
14. Cremers, C.: Scyther - Semantics and Verification of Security Protocols. Ph.D. dissertation, Eindhoven University of Technology (2006)
15. Neuman, B.C., Ts'o, T.: Kerberos: an authentication service for computer networks. IEEE Communications Magazine 32(9), 33–38 (1994)
16. Goodner, M., Nadalin, A.: Web Services Federation Language (WS-Federation). OASIS Specification (work-in-progress) (January 2008), `http://www.oasis-open.org`
17. Narasimha, M., Tsudik, G., Yi, J.H.: 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 Protocols 2003, pp. 336–345 (2003)
18. Saxena, N., Tsudik, G., Yi, J.H.: Admission control in Peer-to-Peer: design and performance evaluation. In: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, pp. 104–113 (2003)

19. Pedersen, T.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
20. Stajano, F., Anderson, R.: The Resurrecting Duckling: security issues for ad-hoc wireless networks. In: Proceedings of the 7th International Workshop on Security Protocols, Cambridge, UK (1999)
21. Maurer, U.: Modelling a Public-Key Infrastructure. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146. Springer, Heidelberg (1996)

Levels of Assurance and Reauthentication in Federated Environments[*]

Manuel Sánchez[1], Óscar Cánovas[2], Gabriel López[1], and Antonio F. Gómez-Skarmeta[1]

[1] Department of Information and Communications Engineering
[2] Department of Computer Engineering
University of Murcia, Spain
{msc,gabilm,skarmeta}@dif.um.es, ocanovas@ditec.um.es

Abstract. This paper presents a generic proposal for improving existing IdM systems, by enabling service providers to determine whether the SSO credentials presented by a user satisfy some minimum requirements. For example, a service provider may require the users to have been authenticated using a method labelled with a particular level of assurance or a credential issued by a specific identity provider. Thus, a user initially authenticated by a username and password might not access a service that requires a stronger mechanism, such as public key certificates. Similarly, the access to some critical service may be restricted to users belonging to a specific organization. The main contribution of this paper is a generic infrastructure that defines the mechanisms to enforce access control policies based on levels of assurance and multiple identities, and it also provides the means to find and redirect the users to the appropriate authentication service when reauthentication is required.

Keywords: SSO, reauthentication, authorization, LoA, federation, eduGAIN.

1 Introduction

Nowadays organizations offer users more and more on-line services. Most of those services require some kind of authentication step in order to determine the set of resources that can be accessed, the preferred appearance and other user-dependent conditions. Usually, the factor that imposes the different level of security required to access these services is the possible consequences derived from an authentication error and misuse of credentials. That is, nobody worries too much if someone accesses the customized session of the digital newspaper he reads, but the problem is somewhat greater if an unauthorized bank transfer is carried out. Therefore these organizations are interested in the definition of the authentication strength required to assure that an entity is indeed the claimed entity, which is known as *Level of Assurance (LoA)*.

Besides, in recent years we have experienced the emergence of federated approaches to resource sharing. In these approaches, trust links are established between different

[*] This work has been partially funded by SWIFT (FP7 project, Grant Number 215832). Thanks also to the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the Fundación Séneca.

autonomous organizations in order to grant users in any of them access to shared resources with a single identity stated by the organization the user belongs to. Usually, federations make use of SSO mechanisms to allow users access to the resources offered, without the need to reauthenticate each time. In those scenarios, organizations offering resources are known as *service providers*, while organizations stating the identity of users are known as *identity providers*. Important examples of these approaches are the establishment of academic federations worldwide like InCommon [4], HAKA [2] and SWITCH [5], which are based on Shibboleth, or eduroam [13].

In this kind of scenarios, where the user usually owns several identities provided by different identity providers, there are situations where the user needs to reauthenticate. For example, when the user is at home, he connects to Internet through a commercial ISP using an identity provided by the ISP, but when he is at work, the identity used by the same person to access the corporate network may differ. A common situation is when the user is browsing the Web at home, using the identity provided by the ISP, and he wants to access some site restricted to users belonging to his organization. Consequently, he needs to authenticate against his organization and, thus, to play the corporate identity. Another slightly different situation occurs when several authentication mechanisms with different LoAs are available. For example, a mechanism based on login and password can be used during the initial access to the network or to read an e-mail, and another one based on the use of a smartcard to digitally sign electronic documents. In that case, a SSO mechanism bootstrapped from the network access after providing the username and password pair cannot be used to gain access to the other services requiring higher LoAs.

This paper presents an infrastructure to facilitate the reauthentication process in federated environments where SSO mechanisms are used and authentication is required initially to access the network. This process is necessary to manage the use of multiple user's identities and LoAs in a federated environment. Moreover, this functionality should be added without modifying the existing IdMs, such as Shibboleth. That is, the new services should be included at a confederation level, connecting different existing federations without modifying their internal protocols. In order to do this, these features are included in a confederation middleware such as eduGAIN [15], which can also be used for single federations. This work is being developed in the DAMe project [1], whose main goal is to define a unified authentication and authorization system for federated services hosted in the eduroam network. Specifically, one of the tasks of this project is the development of a global Single Sign On (SSO) service [19]. This SSO service is based on the eduroam authentication process and the eduGAIN authorization infrastructure.

The rest of this paper is structured as follows. First, related work is shown in Section 2. Section 3 describes what *level of assurance* means and which factors affect this measure. Next, Section 4 provides an overview of the infrastructure supporting the federation services which has been used in this work. The overall description of the reauthentication proposal, with in-depth study of the key elements such as those involved or the structure of the messages, is shown in Section 5. The application of this infrastructure to enable the use of LoAs is described in Section 6. Section 7 defines the trust model and the threats that can affect the infrastructure described. Finally, conclusions and some statements of direction are shown in Section 8.

2 Related Work

This section describes other works related to the management of different levels of authentication. First, a Shibboleth extension to make use of LoA is described. Second, the mechanism used by CardSpace and Higgins to specify the kind of authentication required is analyzed.

FAME (Flexible Access Middleware Extension) [6] is a Shibboleth [20] extension designed for providing multi-level user authentication, that is, the use of LoAs in Shibboleth. This system, based on the cryptographic strength of the authentication protocol used to authenticate the user, calculates a LoA value that is added to the set of user's attributes in the identity provider. Then, when the user is accessing some protected resource, the LoA is passed through Shibboleth to the authorization decision engine, together with other user's attributes, for an authorization decision to be taken in the service provider. Besides, as one of the characteristics of Shibboleth, FAME provides web SSO. There are several differences between FAME and the proposal we are going to describe in this paper. Firstly, as FAME extends Shibboleth, it is oriented to web-based resources. Therefore it provides web SSO, but it does not link the initial authentication to access to the network with the authentication in the Shibboleth IdP, so the user has to authenticate twice, when he accesses the network and the first time he accesses the IdP. In contrast, our proposal provides unified SSO, which means that SSO is bootstrapped from the network authentication. Regarding the LoA, the SP obtains the LoA related to the user after querying the idP about the attribute containing such LoA. Since our proposal includes this information in the SSO token, if only authentication and not authorization is required, there is no need for this additional exchange of messages. Finally, besides the capabilities shared with FAME, the proposal presented here also defines a mechanism to redirect the user to the appropriate authentication service if either, the LoA or the identity, are not valid to access the requested service.

CardSpace [17] and Higgins [3] are two Identity Metasystems (IMeS) which define a similar mechanism to allow service providers, or relying parties (RP), to specify the authentication requirements of the services they offer. Both IMeS are built around the abstraction of the information card, which is a standard representation of the user information. Basically, in these systems, when the user tries to access some service, the information card client installed in his computer recovers the RP policy to determine the requirements of the service, including the allowed authentication mechanisms. Based on this policy, the application displays to the user his information cards satisfying the policy requirements. When the user selects one of the cards, the information card application contacts the IdP that issued that card to get a signed token with the appropriate data. Finally, this token is sent to the RP to get access to the service. In the sense that the required LoA for a service depends on the authentication requirements for the service, the use of the RP policy in these IMeS provides the same functionality that the infrastructure that is described in this paper. But we have to take into account that the scenarios, and therefore the requirements, are different. CardSpace and Higgins are user-centric systems defined to create open user communities, where the main principle to attract people is to give the user the maximum control over thier identities and the identification process. On the other side, the scenario for our proposal is based on the existence of previously established organizations with their own users, which want to collaborate during a specific period of time. Despite these users retain certain control over their identities and the information about them, each organization is responsible for its users to the other organizations. Therefore, the organization must control the process to guarantee the existence and value of certain attributes. Consequently, the organizations maintain the control of the identification process and a solution at a confederation level should be provided.

3 Level of Assurance

An authentication level of assurance (LoA) is defined as the strength of authentication required for a relying party to be assured that an entity is indeed the claimed entity [22]. This definition implies two different factors. On one side, the degree of trust to which the credential being presented actually represents the entity named in it, called *identity proofing*. On the other side, the degree of confidence to which the represented entity actually is the entity engaging the electronic transaction, called *identity binding*.

From the identity provider point of view, these levels are discrete assurance indicators that quantify the degree of protection that the organization provides in the identity management. On the other hand, from the service provider point of view, LoAs are measures of the authentication trustworthiness required to authorize the access to services or resources. Thus, higher LoAs are required to mitigate higher levels of risk.

The UK government was the first organization interested in regulating the security of its on-line services. To do so, it published a security framework to regulate the electronic delivery of government services [18]. With this framework, the UK government provided an approach for determining the security requirements and assuring the presence and proper operation of security countermeasures meeting these requirements. This framework proposes a methodology for identifying functionality and the assurance required to minimize risk. Besides, it defines four types of LoA based on the level of control needed. In this way, services classified as level 0 do not require assurance. Level 1 does not require formal assurance, but threats and vulnerabilities should be considered. The use of level 2 recommends some form of assurance. And finally, the highest level implies that certain security profiles must be satisfied.

The US federal government, continuing with the work done in the UK, defined four different LoAs based on the potential impact of an authentication error [8]. Besides, this document specifies the criteria for determining the LoA required to identify citizens when they access on-line services provided by federal agencies. The levels defined are *minimal assurance of identity, moderate assurance of identity, substantial assurance of identity* and *high assurance of identity*. Each LoA is appropriate for a different kind of electronic transaction. For example, *minimal assurance* is suitable for a user presenting a registered identifier to a web page that offers customized contents while *high assurance* is suitable for a law enforcement officer accessing a law enforcement database containing criminal records.

Later, the National Institute of Standards and Technology (NIST) contributed with supplementary guidelines [21] specifying the technical authentication requirements for the authentication LoAs defined previously. This document specifies the requirements of each LoA related to the process of registering an identity with a *Register Authority* and which obtains the credential, the type of tokens used for authenticating a claimant's identity, the authentication protocol used by the authentication service, the assertion mechanism used to communicate the results of the remote authentication to a relying party, and how the credentials are managed in the user's home institution. For example, based on this specification, simple password challenge-response protocols belong to level 1, the use of a password through a secure authentication protocol belongs to level 2, soft cryptographic tokens belong to level 3, and finally, hard cryptographic tokens belong to level 4.

Furthermore, there is an alternative outlined in [7] that, instead of defining a fixed number of LoA, proposes developing an algorithm that models the factors involved in the identity proofing and binding. This algorithm must provide a value that represents the LoA of a specific credential.

As outlined in the previous section, this concept of LoA and the management issues it implies will be included in eduGAIN. Therefore, the next section presents this confederation infrastructure, which will be extended to enable the use of LoAs and to manage multiple identities.

4 eduGAIN

The main goal of the eduGAIN [12] middleware is to build an interoperable authentication and authorization infrastructure to interconnect different existing federations. Thus, as Figure 1 shows, eduGAIN is responsible for finding the federation which a roaming user belongs to, for translating the messages between the federation internal protocols and eduGAIN and vice versa, and guaranteeing the trust among the participating institutions. For example, as has been proved in DAMe, it can be used to interconnect a Shibboleth-based federation with eduroam.

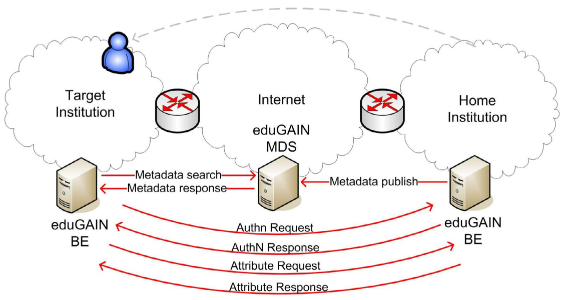


Fig. 1. eduGAIN infrastructure

This goal is achieved by defining a set of common services for the confederation, for example the Metadata Service (MDS) and a confederation-aware element called Bridging Element (BE) that is responsible for connecting the different federations to eduGAIN. Metadata related to federations are published by means of the MDS. These metadata include information for locating the authentication, attribute and authorization services of the member institutions. In this way, the home federation of a roaming user is located by the BE obtaining the information published in the MDS. Then, the appropriate authentication, attribute and authorization requests are translated and routed by the remote BE toward the user's home institution. This scheme is also valid to communicate different institutions belonging to the same federation. The specific way the authentication and authorization processes are carried out in eduGAIN is defined by different profiles [15]. Currently, a profile compatible with Shibboleth, called *Web SSO*, and another that does not require human intervention, called *Automated Client*, are defined.

After the infrastructure needed to set up the federation has been described, we propose a specific profile for eduGAIN that enables the reauthentication process when a particular authentication credential does not satisfy the requirements imposed by a particular service provider.

5 Infrastructure for Reauthentication

Starting from the eduGAIN infrastructure and taking advantage of the points of extension it provides, this section details the necessary elements, policies and protocols to implement the reauthentication process for a user belonging to an eduGAIN confederation. As we will see later, when the user is authenticated in his home institution, he obtains some kind of token, like the one described in [19], which contains data about the authentication process such as the identity provider or the LoA associated to the authentication method. Afterwards, when he tries to access a protected service, it is possible to check whether the user has been authenticated with a suitable LoA or if he plays a valid identity. When the token is valid to access to the service, the authorization process goes on, otherwise the user is redirected to the appropriate authentication service to be reauthenticated and to obtain a new token.

Specific details about the entities, policies and protocols involved in the validation and redirection processes are described in the following subsections.

5.1 Architecture

Initially, as Figure 2 shows, the scenario is composed by several organizations acting as identity provider, the ISP that provides him access to Internet at home or the organization where he works, and the service provider which offers the service the user wants to access. Identity providers are equipped with different authentication mechanisms. Furthermore, through the eduGAIN BEs, these organizations are members of the same federation or belong to different federations that form a confederation. Since the user belongs to several organizations, he can try to access to the resources offered by the service providers using different identities with different properties. Thus, the service provider must check that the user makes use of the proper identity to access the service.

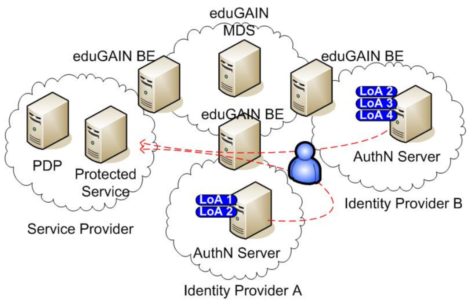


Fig. 2. Federation scenario

Furthermore, after the user's authentication token has been validated, an authorization process may be necessary to enable the access to the service.

The validation process proposed must be transparent to the service provider, that is, it only has to deal with authentication and attribute queries to the appropriate BE, and this element will be responsible for recovering the user's authentication token, validating it and redirecting the user to the appropriate authentication service if necessary. Regarding the validation step, instead of including this functionality in the BE, authorization decisions can be delegated to a *Policy Decision Point (PDP)*, which receives the appropriate data and returns the decision response. These decisions are based on a set of policies defined by the service provider. Furthermore, if the reauthentication of the user is needed, the location of the authentication services defined by an identity provider can be obtained from the confederation Metadata Service (MDS). It is important to note that the validation of the token and the reauthentication processes in the infrastructure proposed are carried out at the (con)federation level, because they depend on global agreements among all the organizations. On the other hand, each service provider can perform a local authorization process to control the access to the resources it offer, that is independent of the other organizations.

Next, as is detailed in the following section, after the appropriate authorization service is discovered, the user is redirected to that site to get a new authentication token.

5.2 Communication Profile

This subsection describes in detail the process needed to validate the user's authentication token when he tries to access a protected service, and the subsequent redirection to the appropriate authentication service if this token is not valid. However, before this process, the user has to obtain the initial token during the network authentication. The network authentication process is described in detail in [19]. Basically, as Figure 3 shows, the authentication is based on eduroam. In eduroam, access control is carried out following the 802.1X standard. That is, the user associates with the wireless access point (AP), which contacts its local RADIUS server in order to authenticate the user. But when this server identifies that the user belongs to a different domain the authentication request is forwarded through a RADIUS hierarchy to the server located in the user's home institution. Then, the user is authenticated and the response is routed back to the remote institution, where the AP enables the requested connection. Thus, at the end, the user is authenticated through an EAP method, specifically PEAP, because it provides a protected channel that can be used to send data to the user in a secure way. At this point, the eduroam authentication process is extended by means of the generation of the SSO token, which is built by an eduGAIN BE. Finally, this token is sent to the user.

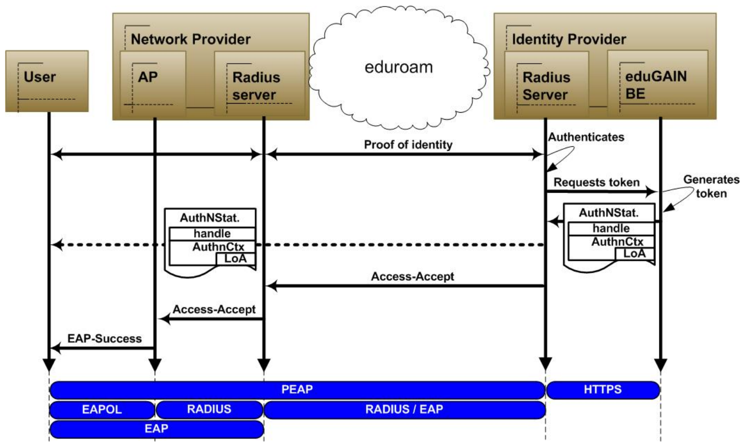


Fig. 3. Initial network authentication

Once the user obtains the authentication token he can try to access a protected service. As Figure 4 shows, this process can be split into several steps:

– *Access*. The user tries to access a protected service, such as a website, from the service provider.
– *Validation*. The user is redirected to the BE and the token is sent directly by the user's device middleware to the BE. However if he owns several tokens from previous authentications, he must select one. This selection can be based, for example, on the organization that issued the token, the LoA associated to the token or any other characteristic. The token is validated by the PDP against specific policies defined in the organization.
– *Redirection and reauthentication*. If the validation fails, the BE consults the MDS to get the list of authentication services in the federation that satisfy its requirements. Based on the information obtained from the MDS, the BE presents the user with the list of authentication services allowed that can be used to access the service. Once the user has selected one of them he is redirected and reauthenticated, obtaining a new authentication token.
– *Validation*. Finally, the user is redirected again to the BE, which takes the new token and validates it.
– *Local authorization*. Now, if the validation is successful, the user is redirected to the protected service, where additional authorization verifications can be carried out.

Specific details about the MDS mentioned above and the specific mechanism used by the BEs to contact it, are described in the next section.

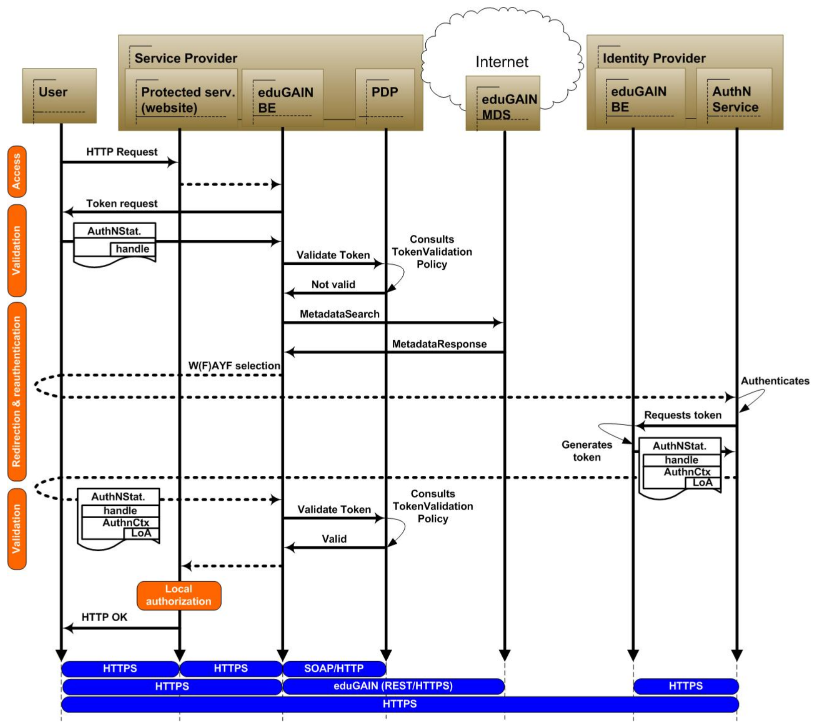


Fig. 4. LoA message profile

5.3 Management of Metadata

Metadata in eduGAIN follows the SAML 2.0 metadata specification [10]. Therefore, as Figure 5 shows, an authentication service is described by means of an XML document with an *EntityDescriptor* element. It includes information about the home institution and its authentication services using the *Organization* and the *AuthnAuthorityDescriptor* elements respectively. The latter contains an *AuthnQueryService* element for each different authentication point that the institution defines. This element specifies the location and binding of the service. Besides, this example also uses the extensions points defined in the SAML metadata specification to define the LoA provided by the authentication service.

The specification of eduGAIN [12] defines three messages to interact with the MDS: *MetadataPublish* for publishing the metadata in the MDS; *MetadataSearch* for querying the MDS for specific information; and finally *MetadataResponse* for sending the information from the MDS to the BE that is requesting that information. The *MetadataPublish* message includes, besides the XML document with the metadata, the identifier of the BE publishing the information. The *MetadataSearch* contains the search parameters, *HomeLocatorIndicators*, used to specify the information to search for. Finally, the *MetadataResponse* message contains the requested metadata if the result of the search is successful.

```xml
<md:EntityDescriptor entityID="https://lolo.inf.um.es/AuthNService"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:loa="urn:um:schemas:loa"
    xsi:schemaLocation="urn:oasis:names:tc:SAML:2.0:metadata schemas/saml-schema-metadata-2.0.xsd"
    validUntil="2007-12-31T23:59:59.0Z">
  <md:AuthnAuthorityDescriptor xsi:type="md:AuthnAuthorityDescriptorType"
      protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:AuthnQueryService Location="https://server.univeritya.org/saml/AuthNService/LoA2service"
        Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP">
      <loa:LoAValue>2</loa:LoAValue>
    </md:AuthnQueryService>
  </md:AuthnAuthorityDescriptor>
  <md:Organization>
    <md:OrganizationName xml:lang="en">OrgA</md:OrganizationName>
    <md:OrganizationDisplayName xml:lang="en">Organization A</md:OrganizationDisplayName>
    <md:OrganizationURL xml:lang="en">www.organizationA.org</md:OrganizationURL>
  </md:Organization>
</md:EntityDescriptor>
```

Fig. 5. Metadata describing the authentication service

Now the general process to validate the user's authentication token and the redirection of the user to the proper authentication service to reauthenticate have been described, the next section defines a specific profile to use the LoA of the authentication to guide the validation of the token.

6 Support for Different Levels of Assurance

6.1 LoA Validation Model

This section defines a model for the LoA validation process carried out when a user is trying to access a protected service in a federated environment. In the proposed scenario, institutions acting as service providers offer a set of services (S) to the users who belong to the federation:

$$S = (S_1..S_n)$$

A set of LoAs (L) related to the security needed in the interaction between users and services is also defined at the federation level. A precedence operator (I) is defined among these levels, based on the potential impact of an authentication error and misuse of credentials. In this way, LoAs can be arranged on the basis of the security they guarantee.

$$L = (L_1..L_m), I(L_1) < .. < I(L_m)$$

These levels are associated with both users and services. On the one hand, the credential (C_u) obtained by the user (U) after his authentication in the Identity Provider, contains the LoA (L_u) provided by the authentication method used. In this way, if the user authenticates using a different authentication method, he will obtain a credential with a different LoA. This process is shown in Figure 4, when the BE from the Identity Provider generates an *AuthNStatement* which contains the LoA attribute.

$$C_u = (U, L_u)$$

On the other hand, each institution defines a LoA validation policy (LVP), where each service (S_i) they offer is associated to the minimum LoA (L_j) required to access it. Specifically, this policy would correspond with the *Token Validation Policy* from Figure 4.

$$LVP = (S_i, L_j), \forall S_i \subset S, L_j \subset L$$

Taking into account the previous definitions when a user tries to access to some service (S_i), the system or specifically the BE from the Service Provider in Figure 4, generates a LoA validation request (R). This request is used to check if the LoA associated with the user's credential used for authentication (L_u) is valid to access the requested service.

$$R = (L_u, S_i)$$

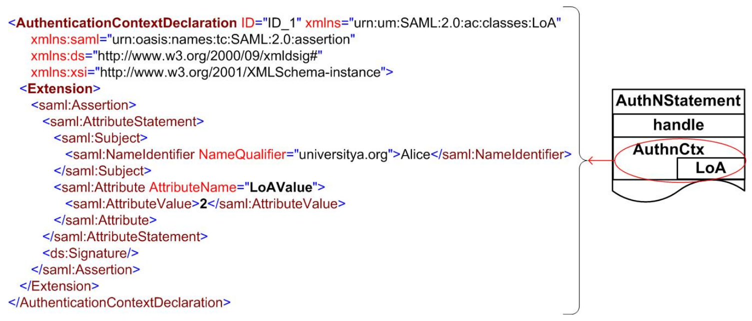
Then, based on this request, the LoA of the user (L_u) will be valid to access the requested service (S_i), if the LoA associated with the service (L_j) is lower than or equal to the user's.

$$\exists(S_i, L_j) \subset LVP/I(L_j) \leq I(L_u)$$

The specific mechanism used to include the LoA information in the SSO token is specified in the next section.

6.2 Including the LoA in the SSO Token

As previously outlined, after the authentication process in his home institution the user receives the token, which contains some information that specifies the LoA of the authentication service. According to the proposal for SSO in DAMe [19], the user's token is a signed SAML v2.0 [9] *Assertion* containing an *Authentication Statement*. Therefore, the LoA can be included using an *AuthnContext* [14] element, as Figure 6 shows.



```
<AuthenticationContextDeclaration ID="ID_1" xmlns="urn:um:SAML:2.0:ac:classes:LoA"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Extension>
    <saml:Assertion>
      <saml:AttributeStatement>
        <saml:Subject>
          <saml:NameIdentifier NameQualifier="universitya.org">Alice</saml:NameIdentifier>
        </saml:Subject>
        <saml:Attribute AttributeName="LoAValue">
          <saml:AttributeValue>2</saml:AttributeValue>
        </saml:Attribute>
      </saml:AttributeStatement>
      <ds:Signature/>
    </saml:Assertion>
  </Extension>
</AuthenticationContextDeclaration>
```

Fig. 6. Example of *AuthnContext* including LoA

The *AuthnContext* is a new element defined in SAML v2.0 to enable the authentication service to include some information related to the authentication process. This information can be used by the assertion consumer to assess the level of confidence that it can place in that assertion. SAML has defined several authentication context classes to be used with the most common authentication mechanisms, as for example X.509 public key certificates or login and password. Furthermore, it is possible to create new classes thanks to the extensibility points included in this specification. Altough it is clear that this is the appropriate element to include the information about the LoA, the way to specify it is under discussion. Due to the flexibility of this element, there are several possibilities to express the LoA information. One option is to use the existing *AuthnContext* schemas that identify the authentication mechanism used, and let the service provider to calculate the LoA associated with that authentication mechanism. Another option is to specify directly the LoA in the *AuthnContext* when it is created. This can be done by defining a new schema for each LoA. Finally, another option, following the same approach, is to include a SAML attribute specifying the LoA in the *AuthnContext*. Any of these three options can be used, and the changes introduced by the use of any of them are minimal. Therefore, although the example from Figure 6 is based on the last alternative, any of the other options could be used in this proposal.

After the eduGAIN BE from the service provider receives the authentication token, the system uses the appropriate policies to check if the user's LoA is enough to access to the resource. These policies are defined in next section.

6.3 XACML Policies to Guide the LoA Validation

The decision as to whether the user's LoA is appropriate to access a service should be based on some policies defined in the federation. These policies are based on the

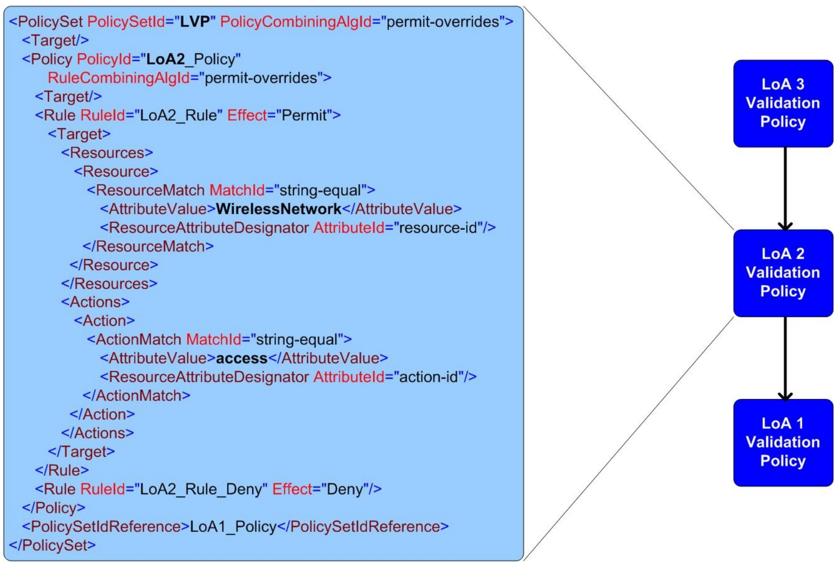

```
<PolicySet PolicySetId="LVP" PolicyCombiningAlgId="permit-overrides">
  <Target/>
  <Policy PolicyId="LoA2_Policy"
      RuleCombiningAlgId="permit-overrides">
    <Target/>
    <Rule RuleId="LoA2_Rule" Effect="Permit">
      <Target>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="string-equal">
              <AttributeValue>WirelessNetwork</AttributeValue>
              <ResourceAttributeDesignator AttributeId="resource-id"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="string-equal">
              <AttributeValue>access</AttributeValue>
              <ResourceAttributeDesignator AttributeId="action-id"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
    <Rule RuleId="LoA2_Rule_Deny" Effect="Deny"/>
  </Policy>
  <PolicySetIdReference>LoA1_Policy</PolicySetIdReference>
</PolicySet>
```

Fig. 7. LoA Validation Policy

model defined in Section 6.1. Therefore, first it is necessary to have a policy to define the different LoAs and the precedence relation among them, *LoA Definition Policy*. Second, another policy is necessary to specify the LoA associated with each service, *LoA Validation Policy*. The first one is a global policy that must be widely available for all the members of the federation, while the second is defined by each member organization to protect the its resources.

These policies can be specified using XACML [11], such as in [16]. XACML is a standard flexible XML-based access control language and, besides, there are free implementations available. Figure 7 shows an example of LoA Validation Policy defined using this language. The policy is part of a hierarchy, in such a way that each level of the policy includes the rights of the inferior levels. Specifically, this policy indicates that at least *LoA 2* value is necessary to *access* the *wireless network*.

The next section describes the trust model needed to guarantee the trust among the entities of the architecture presented and analyzes the possible threats.

7 Trust and Threats Models

The improved SSO architecture presented in this paper is based on eduGAIN. Therefore, the trust model of this proposal is mainly based on its model but including some specific details. As described in [15], the trust model of eduGAIN is based on a PKI model, so each component owns an X.509 digital certificate to identify itself. The eduGAIN confederation offers a PKI, named eduGAIN PKI[1], that can be used to provide the certificates to these entities. In this way, trust is guaranteed by the establishment of TLS connections between the peers in each interaction. XML-Sig digital signatures are also used to protect the integrity of the appropriate assertions exchanged. More specifically, we can analyze the security issues of each different interaction of the system.

The first step in the process is the publication of the metadata by the member organizations. Problems can arise if metadata are sent to the MDS by an unauthorized entity or if the metadata are modified without authorization. The former is solved by the mutual authentication performed in TLS by the eduGAIN entities before any interaction, because the MDS can be sure about the identity of the sender of the metadata. The latter is solved if the message with the metadata is digitally signed. But it is also necessary to protect the metadata in the MDS, once these data have been received.

The user authentication and the delivery of the token is the responsibility of each individual organization. Specifically, this is one of the factors that determine the LoA provided by an authentication services. But regarding the token, what can affect the rest of the federation is the protection of the token once it is on the user's device. On the one hand, the token is digitally signed to avoid possible modifications by a malicious user. On the other, the problem of the token is that, since it is for SSO, the impact of a theft of the token is bigger than with traditional credentials. Therefore, the token must be stored encrypted in the filesystem and only the necessary software elements must have access to it.

[1] http://sca.edugain.org/

The instant when the user accesses a protected resource is another tricky moment for the security of the token, because it must be sent to the service provider. This process must be properly protected to avoid the theft of the token. To do this, the channel used to send the token must be secured, but the user also must be sure that the entity receiving the token is really the entity that should receive it. Again, the identity certificates from the eduGAIN infrastructure are used, firstly, to identify the BE that is requesting the token to the user, and secondly, to create the secure channel to send it.

Finally, when the remote BE queries the MDS to obtain the information to redirect the user to an appropriate authentication service, the critical aspects are the identification of the MDS to avoid a malicious user providing false metadata, and the security of the transmission of the metadata to the requesting BE. The TLS mutual authentication and the resulting secure channel solve both problems.

Another problem arises regarding the use of the LoA to guide the token validation. In the scenario described above, the LoA information is included in the SSO token by the identity provider, and later this information is used by the service provider to determine if the user is allowed or not to access some service. In this way, the service provider must trust this information issued by the identity provider. The legal agreements of the federation mitigate in part the risk of this situation. These agreements, which must be signed by an organization when it joins the federation, must specify the security requisites that an authentication service must support to provide a specific LoA. Therefore, if these requisites are not satisfied, the organization must be expelled from the federation. This situation implies that some kind of periodic audits need to be carried out in the member organizations to verify that the requirements demanded are satisfied by the authentication services.

To end this paper, the next section shows the conclusions and some statements of direction are also outlined.

8 Conclusions and Future Work

This paper describes the existence of different situations in an SSO federated environment where it is necessary for a user to reauthenticate, for example when the identity provider from the user is not accepted by the service provider or when the authentication mechanism used to authenticate the user, that is its related LoA, is not secure enough to access the requested service.

Given this situation, this paper proposes to improve the SSO mechanism by means of an infrastructure for validation and re-generation of SSO credentials. This goal has been achieved by extending eduGAIN, a middleware for confederations, with the necessary services, protocols and policies for managing the validation of the user's identity and the redirection process. The use of this middleware also provides the advantage that different kinds of federations such as Shibboleth and PAPI can interact. Moreover, a specific profile to base the identity validation in the LoA of the user credentials is described. Furthermore, the security analysis of the infrastructure presented in this work has shown that the main security risks can be avoided by deploying the proper digital certificates. These certificates are then used to provide mutual authentication, digitally sign assertions and to establish secure communication channels.

Finally, as a statement of direction, we are currently working on a prototype of the system to include the use of LoAs in a real environment such as eduroam. We are also analyzing how to integrate this proposal into the security infrastructure of the Grid.

References

1. DAMe Project web site, `http://dame.inf.um.es`
2. The Haka federation web site, `http://www.csc.fi/suomi/funet/middleware/english/index.phtml`
3. Higgins project web site, `http://www.eclipse.org/higgins`
4. The InCommon federation web site, `http://www.incommonfederation.org/`
5. The SWITCH federation web site, `http://www.switch.ch/aai`
6. Chin, J., Goble, C., Nenadic, A., Zhang, N.: FAME: Adding mult-level authentication to Shibboleth. In: Proceedings of IEEE Conference of E-Science and Grid Computing, Amsterdam Holland (2006)
7. Alterman, P., Nazario, N., Louden, C.: White paper: e-Authentication partnership policy on levels of assurance of identity for authentication of electronic identity credentials
8. Bolten, J.B.: E-Authentication Guidance for Federal Agencies (December 2003)
9. Cantor, S., Kemp, J., Philpott, R., Eve, M.: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0, OASIS Standard (March 2005)
10. Cantor, S., Moreh, J., Philpott, R., Maler, E.: Metadata for the OASIS Security Assertion Markup Language (SAML) v2.0, OASIS Standard (March 2005)
11. Anderson, A., et al.: EXtensible Access Control Markup Language (XACML) V 1.0, OASIS Standard (February 2003)
12. López, D.R., et al.: Deliverable DJ5.2.2,2: GÉANT2 Authorisation and Authentication Infrastructure (AAI) Architecture - second edition, GN2 JRA5. GÉANT2 (April 2007)
13. Wierenga, K., et al.: Deliverable DJ5.1.4: Inter-NREN Roaming Architecture. Description and Development Items, GN2 JRA5. GÉANT 2 (September 2006)
14. Kemp, J., Cantor, S., Mishra, P., Philpott, R., Maller, E.: Authentication Context for the OASIS Security Assertion Markup Language (SAML) v2.0, OASIS Standard (March 2005)
15. López, D.R., Macias, J., Molina, M., Rauschenbach, J., Solberg, A., Stanica, M.: Deliverable DJ5.2.3,2: Best Practice Guide - AAI Cookbook - Second Edition, GN2 JRA5. GÉANT 2 (2007)
16. López, G., Cánovas, O., Gómez, A.F.: Use of xacml policies for a network access control service. In: Proceedings 4th International Workshop for Applied PKI, IWAP 2005, pp. 111–122. IOS Press, Amsterdam (2005)
17. Microsoft Corporation. A Technical Reference for InfoCard v1.0 in Windows (August 2005)
18. Office of the e-Envoy, UK online. e-Government Strategy Framework Policy and Guidelines. Version 2.0 (September 2002)
19. Sánchez, M., López, G., Cánovas, O., Gómez-Skarmeta, A.F.: Bootstrapping a global SSO from network access control mechanisms. In: Fourth European PKI Workshop (June 2007)
20. Scavo, T., Cantor, S.: Shibboleth Architecture. Technical Overview, Working Draft 02 (June 2005)
21. Polk, W.T., Burr, W.E., Dodson, D.F.: Electronic Authentication Guideline. Recommendations of the National Institute of Standards and Technology (April 2006)
22. Zhang, N.: E-Infrastructure Security: An Investigation of Authentication Levels of Assurance (LoAs), Open Grid Forum (2007)

Current Status of Japanese Government PKI Systems

Yasuo Miyakawa[1], Takashi Kurokawa[1], Akihiro Yamamura[1], and Yasushi Matsumoto[2]

[1] National Institute of Information and Communications Technology, 4-2-1 Nukui-Kitamachi, Koganei, Tokyo, Japan
{miyakawa,blackriver,aki}@nict.go.jp

[2] Information-technology Promotion Agency, 2-28-8, Hon-Komagome, Bunkyo-ku, Tokyo, Japan
y-matsu@ipa.go.jp

Abstract. In Japan, three government PKI systems are constructed as Bridge Model PKI, and they are also bridged with each other. Up to now, all of the three PKI systems have issued certificates mainly for digital signature on digital documents. Only recently, a concern to issue certificates for entity authentication has been raised. Not only "KeyUsage" but also "CertificatePolicies" and related extensions should be carefully used in Bridge Model. As a potential international issue, we have started to discuss on the migration of cryptography in PKI systems. Due to Bridge Model and vertically divided administration, difficulties in enforcing consistent policies thoroughly in these PKI systems is expected in Japan.

Keywords: eGovernment, Bridge CA, Interoperability, Policies, Level of Assurance, Migration.

1 Overview of Government PKI Systems in Japan

In Japan, there are three government PKI systems: namely Government Public Key Infrastructure (GPKI)[1], Local Government Public Key Infrastructure (LGPKI)[2] and Jumin Public Key Infrastructure (JPKI)[3].

GPKI was constructed to confirm that digital documents, which are exchanged between government and citizens via the Internet, can be validated if it was truly written by the insisting party and the content of the digital document is not altered. LGPKI is the PKI system for local authority, and JPKI is provided as a public service for citizens.

Japanese PKI systems have two characteristics:

- Trust model is Bridge Model
- They have been constructed as PKI systems for digital signature and non-repudiation on digital documents.

[1] http://www.gpki.go.jp/
[2] http://www.lgpki.jp/
[3] http://www.jpki.go.jp/

1.1 Characteristic 1: Trust Model Is Bridge Model

For all of the three PKI systems, Bridge Model [22] has been adopted as its trust model. As the background for GPKI to have adopted this trust model, the requirement that each ministries can set up theirs own CAs and prepare their specific application had been valued; i.e. vertically divided administration had been considered at the beginning. As single certificate policy is adopted for local authorities within LGPKI, LGPKI is operated in the same way as hierarchical model. But, to keep each local authority as trust point, and to implement that there is not legally superior body, Bridge Model is adopted. JPKI is also designed in the same way as LGPKI.

The bridge CA of GPKI and that of LGPKI are issuing cross certificate each other, and the bridge CA of GPKI and that of JPKI are issuing cross certificate each other. Arrows in the following figure (Fig. 1) shows the insurance of cross certificates. In this way, multi-domain PKI systems are operated.

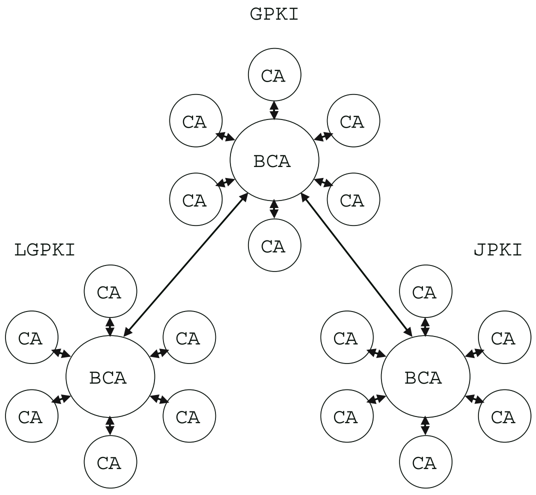


Fig. 1. Trust model of government PKI systems in Japan

As Japanese government and local authorities have adopted Bridge Model as a trust model, there have been several difficulties in keeping its interoperability regarding the path building and path validation. If Bridge Model is adopted, following extensions of X.509 must be used in cross certificates issued by CAs: CertificatePolicies (CP), PolicyMappings, BasicConstraints and PolicyConstraints. In building and validating a trust path, applications should process additional path established by the PolicyMappings in each cross certificates, from the top of respective root CA. Thus, if there were not for valid CPs, the trust path cannot be built and verified to the end.

GPKI's specification for interoperability [1], which is based on RFC 3280 [11], has been disclosed since the system was started to operate in 2000.

1.2 Characteristic 2: PKI for Digital Signature on Digital Documents

Ministries had been issuing variety of paper certificates before information technology has been utilized. It was inevitable that government have concern to digitally sign digital documents for non-repudiation purpose. Japanese government PKI systems were constructed under the circumstances, where replacing official paper documents into digital documents had been regarded top priority. For all of the three PKI systems, Certificates for non-repudiation has been issued mainly, and there have been little attention to certificates for entity authentication. In Japan, ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS [9] was enacted in April 2001. It took after German Electronic Signature Law. Under this Law and related Ministerial Ordinance, digitally signed documents were given effects equivalent to that of paper documents, which was physically signed or stamped. In Japan, stamping is common practice rather than signing. And, in this Law, time stamping to proof effective term is not provided for. But, in this Law, accrediting conditions for CAs to allow issuing certificates whose certificate has effect in terms of this law's aim is also prescribed. Of course, accredited CAs are external entity to government PKI system, and there are eighteen CAs at the end of 2007. These accredited CAs are also bridged with GPKI.

As stated above, Japanese government PKI systems had been constructed mainly to play the role of infrastructure to validate signatures on digital documents. Although certificates for Web server authentication are issued by LGPKI CA, it was not regarded important in other PKI systems. For GPKI CA, it is going to issue Web server certificates soon.

2 Current Status

2.1 Current Status 1: Interoperability in Bridge Model

Client applications for Japanese government PKI systems have to implement path building and validation function for Bridge Model PKI. It may require applications to go through multiple steps of bridged paths. So, to confirm the interoperability of the applications become difficult.

To assist the development of applications for Bridge Model PKI, we have developed a testing tool, and made it available for public use with some testing data since 2002 [6].

At this time, there is not a responsible body caring about the consistent policies between PKI systems in Japan. This kind of body is existing in US: The Federal Public Key Infrastructure (FPKI) Policy Authority[4].

2.2 Current Status 2: Digital Signature on Digital Documents

General Condition. Unfortunately, these government PKI systems are not well used widely. The only exception may be for using electronic bidding and

[4] `http://www.cio.gov/fpkipa/`

procurement systems for government. Each ministry has made it mandatory to use the system, and the system requires digital certificates. But for other applications, those systems are not well used.

There may be several causes for this. First, the measure to enforce the use may be insufficient. Second, the potential users may be limited to professionals who have rights and abilities, so, assumed users are few. (e.g.: Licensed tax accountant, Public consultant on social insurance and Administrative scrivener) Although there are a various kinds of professional jobs, those respective ministries may be different, and their policies also tend to be different. It is hard to develop aver all strategy to promote the use of PKI systems for non-repudiation purpose.

Also, CA business related to certified CAs under Law Concerning Electronic Signatures and Certification Services is not so active. In fact, it may have been difficult to be operated as dedicated CA for Non-repudiation. At present, how to promote the use of these PKI systems is a difficult issue.

Different Types of Digital Signatures. It is supposed that certificates which are compliant with ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS are issued only to natural person. But there are several PKI application systems in which different kind of certificates are issued and used.

An example is the PKI system for commercial register system[5]. On this system, certificates are issued for legal entity, enterprises.

As the second example, regarding certificates for government officials, irregular procedure is adopted: certificates for the posts, corresponding to its rolls, are issued to government officials. GPKI CAs and LGPKI CAs are mainly issuing this kind of certificates. Attribute Certificates are not used for them.

Time Stamping for Electronic Documents. ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS in 2001 has aimed to make digitally signed documents have similar effects as physically signed or stamped paper documents. There had been a fundamental problem that basic conditions regarding digital documents are lacking.

First, a lot of laws so far had been required corresponding official documents with some format to be archived as physical paper. So, there had been few electronic documents. To respond this problem, Electronic Document Law was enacted and enforced in 2005. As it is not so long since then, it is not certain if the use state for digital signature has improved.

Second, in this law, time stamping to proof effective term is not provided for. For official documents, the date must be written there. But, it is also true that, there is no requirement which need second accuracy for time on those documents. At present, although there is a semi-governmental time stamp service, Time Business Accreditation Center[6] in Japan, it is not well utilized in public sector.

As stated above, Japanese government PKI systems were constructed under the circumstances that replacing official paper documents into digital documents had been regarded top priority. But, current using status is not comfortable. As

[5] `http://www.moj.go.jp/ONLINE/CERTIFICATION/`

[6] `http://www.dekyo.or.jp/tb/`

certificates for non-repudiation are supposed to be related to tight requirement of authorizing behavior, it may be natural consequence.

Credentials for Citizens. In Japan, certificates and corresponding private keys are also issued to citizens [10], but it is only for applicant of more than fifteen years old. As a PKI system to issue certificates to citizens, JPKI is operated. And its certificates are also for non-repudiation. Those credentials, certificate and corresponding private key, are issued to whom wants to get it with some fee. The price varies from municipality to municipality.

Usually, some kind of ID number which can be used over network is included in credentials. But, within JPKI's certificate, there is no network ID number in credentials, and the following four attribute elements are included in "subjectAltName":

– commonName
– dateOfBirth
– gender
– Address

Historically, there is not ID number or ID code for citizens on this certificate, reflecting the early discussion on privacy issue. It does not mean that ID number or ID code is not contained in this smart card. Its concept itself does not exist officially in Japan. There is not Social Security No. like US.

As a device to contain the credential, smart cards are used mainly. But, there is not unified specification for smart cards. And, unfortunately, all of those smart-card specifications are not able to contain multiple credentials now. And much worse, its specification is not disclosed.

3 Recent Undertakings

3.1 Optimization for GPKI System

In current GPKI system, each ministry has its own CA, so, there are fourteen CAs around the bridge CA of GPKI. Those CAs are mainly doing similar function in distributed manor: issuing certificates for their staffs. From overall view, this issuance function can be centralized, and the system monitoring can be done more efficiently if they were centralized. Thus, project to optimize GPKI structure has begun. Responsible ministry is going to achieve it by the end of FY 2008.

In principle, CAs of ministries will be combined to so-called Common CA. Ministries will focus on RA function. In this way, GPKI, which is a complicated Bridge Model PKI, will become simpler than ever. But, Bridge CA function will still remain. Several ministries are still willing to operate special CAs to support some Government to Business service. (e.g.: Commercial register system[7] operated by The Ministry of Justice).

[7] http://www.moj.go.jp/ONLINE/CERTIFICATION/

3.2 Concern for PKI Systems for Entity Authentication

As stated above, certificate for authentication, which has Certificate Policy for authentication, have never issued from government PKI system's CA in Japan up to now. But, the first case is now under discussion.

In the latter half of 2007, it turned out that citizen's pension database system had been carelessly managed, and many data have been lost. So, it became important social issue. The requirement to make people possible to inspect their own pension record became to get attention. About the credentials to be distributed for citizens, necessity of certificate for authentication purpose became recognized. Several issues related to them also got attention as follows.

Related Issue 1: Level of Assurance for Authentication. Regarding authentication services which are provided by the Internet servers, there is a project to develop guideline documents for setting requirements of authentication systems. The basic idea was taken after from US federal government's e-Authentication[8]. Its technical requirements are specified in SP 800-63 [5] issued by National Institute of Standards and Technology (NIST) in US, which covers remote authentication of users over open networks. It defines technical requirements for each of four levels of assurance in the areas of identity proofing, registration, tokens, authentication protocols and related assertions. Under this criterion, not all of the four Levels are corresponding to PKI technology. PKI authentication is required for higher level of assurance.

In Japan, it is necessary to avoid an excessive system investment, because financial situation has deteriorated. From viewpoint to avoid excessive functional requirements, the idea to set several levels for authenticating function is supported. Basically, we support the concept of Level of Assurance such as four levels defined in SP 800-63 in this field.

In countries other than US, The University of Manchester in UK has conducted a study [25] regarding the acceptance of SP 800-63's criteria. In Japan, such a study has not been done yet.

From the viewpoint of avoiding excessive functional requirements, it may be also important to define the Levels which require hardware tokens clearly, and make it possible to determine which requirement is applied. In SP 800-63, it corresponds to Level 3 and Level 4. In Japan, smart card will continue to be typical hardware token device. It should be avoided to procure many authentication systems requiring hardware tokens, causing excessive investments.

Level 4 in SP 800-63 requires tokens to be tested under FIPS 140 series criteria [18] in Cryptographic Module Validation Program (CMVP)[9]. Regarding this, Japan also started Japan Cryptographic Module Validation Program (JCMVP)[10] hardware security testing based on FIPS 140- series and ISO/IEC 19790. So, it is potentially possible to adopt similar operation as US.

[8] `http://asc.gsa.gov/portal/template/welcome.vm`
[9] `http://csrc.nist.gov/groups/STM/cmvp/index.html`
[10] `http://www.ipa.go.jp/security/jcmvp/index.html`

Related Issue 2: Smart Card Format to Contain Multiple Credentials. Issuing certificates for authentication affects the specification of citizen's smart card format, because current specification can not contain multiple credentials, certificates and corresponding private keys. Issuing multiple cards per a person may require multiple times of expense, and it does not become even user's convenience. For smart card as a token, the specification should be appropriately defined to allow containing credentials for both non-repudiation and authentication.

Also, to make a card usable for multiple systems, and to make multiple venders can develop consistent system, inter-operability for the cards is required. The specification is expected to be open, but currently, it is not disclosed.

Related Issue 3: Proper Use of Certificate Policies (CP). Until now, Japanese government PKI systems have experienced issuance of certificates for non-repudiation and Web server authentication, but not for authentication yet. So, it is doubtful that the CPs for already issued certificates are described properly, as there was no need for that. When newly issuing certificates for authentication purpose, CPs must be distinguished properly.

In several Asian countries, there are cases in which certificates for non-repudiation and those for authentication are not distinguished. (e.g.: Korea and Singapore) But, they should be explicitly distinguished from security point of view. Here is an attack scenario as follows:

We describe a man in the middle attack if a certificate is abused; if a digital certificate for entity authentication is not explicitly described as for such a use, there is a plausible attack. Suppose that a client **C** tries to log-in a server **S** following a challenge-response authentication given in ISO/IEC 9798-3. We now consider the case that **C** has a certificate for non-repudiation, but tries to use it for challenge-response authentication. Let us recall that in a challenge-response scheme, a server **S** generates a random number r and sends it to the client **C** as a challenge and then **C** generates a signature of the random number r. The signature is returned and **S** checks whether or not the signature is generated by **C** using his/her secret key. If so, **S** accepts the user as an authentic client.

Now suppose that an attacker **E** tries to forge a signature generated by a client **C**. Suppose **C** start an entity authentication. Then **E** intercepts the message from **C** to **S** to ask a session, and generates a document $\mathcal{D}$ for which **S** want to forge a signature generated by **C**. For example, $\mathcal{D}$ may be a document saying that "I owe one million dollar to **E** and pay back by the end of the year". Note that a challenge r is supposed to be a random number and so it can be any binary bit unless the size or format of a challenge is specified. Here, we make the situation simple for easy understanding. **E** sends $\mathcal{D}$ to **C** as if it was a challenge r from **S**. If **C** is not careful enough, **C** generates a signature of $\mathcal{D}$ and sends it back while **C** believes this is a part of authentication process. Then **E** receives the signature of the document $\mathcal{D}$ generated by **C**. If the certificate for the key of **C** is specified as an entity authentication use, the signature cannot be a proof of the authenticity of the signature of the document $\mathcal{D}$.

On the other hand, if the certificate for the key of **C** is specified as a non-repudiation even though **C** uses it as an entity authentication, the signature obtained by **E** is actually generated by **C** and proves the effectiveness of the signature of $\mathcal{D}$. Then **C** has responsibility for this signature because **C** cannot repudiate it. See Figure 2.

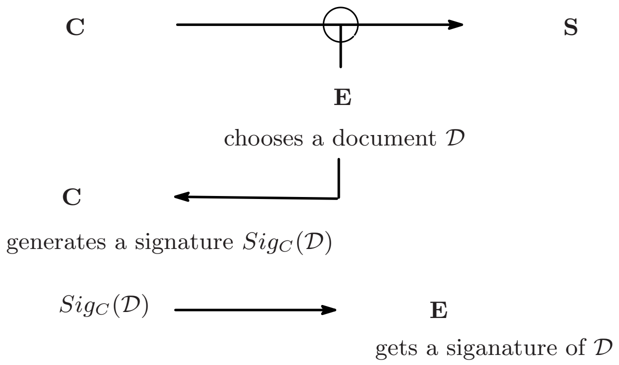

C
S
E
chooses a document $\mathcal{D}$
C
generates a signature $Sig_C(\mathcal{D})$
$Sig_C(\mathcal{D})$
E
gets a siganature of $\mathcal{D}$

Fig. 2. Plausible man in the middle attack

Therefore, it is quite important to distinguish certificates according to its use, that is, one must completely fill in the extension field “keyUsage” in the X.509. Not only “KeyUsage”, but also “CertificatePolicies” should be used appropriately in Bridge Model.

Note that the same attack can be applied if multiple certificates are issued for a secret/public key pair. Such a use is not explicitly prohibited in the current X.509 specifications, however, this type of use must be avoided. Thus, CPs should be defined clearly, and should be operated appropriately. Due to the vertically divided administration and the adoption of Bridge Model, it will be difficult to enforce the CP thoroughly.

3.3 Revising ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS and Its Ministerial Ordinance

Recently, in December 2007, discussion to revise ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS and its Ministerial Ordinance has started. As a part of the discussion, the strict description for digital signature technologies and recommended parameters is under review, and is to be revised. In current Ministerial Ordinance, cryptographic algorithms such as

- sha1WithRSAEncryption and id-RSASSA-PSS [21] whose modulus are 1024 bits length or over respectively,
- ecdsa-with-Sha1 [2] whose base point order is 160 bits length or over and
- id-dsa-with-sha1 [19] whose modulus is 1024 bits length

are strictly described. The revision discussion is aiming to include additional more secure algorithms.

At the discussion, the strategy to promote electronic signatures is also an issue.

3.4 Estimating the Improvement of Factoring Power

For the secure use of cryptographic techniques, it is important to evaluate the lower limit of computational cost. The security of RSA depends heavily on the difficulty of integer factoring problem and the general number field sieve method [16] is the fastest mathematical algorithm for solving the integer factoring problem at present. There are two main kinds of computation in the general NFS method; the sieving step and the linear algebra step.

It is extremely difficult to evaluate the computational cost to factor 1024 bits RSA type composites using the general NFS method. Last year, Cryptography Research and Evaluation Committees (CRYPTREC)[11] in Japan investigated computational complexity of the general NFS method [8].

As a standard PC for this evaluation we chose an Athlon 64 2.2GHz CPU. In terms of cost performance it is an extremely good PC for trying to break the current world record for integer factoring. For actual results for 1536 bits and 2048 bits we relied on the evaluations of Dr. T. Kleinjung [14,15], one of the current holders of the world record for integer factoring. We also used results for 1024 bits obtained using almost same procedures [13]. We follow the projected performance graph of computational power of supercomputers given in the TOP500 project[12].

The graph below (Fig. 3) indicates a time when we are able to factor RSA type composites. However, we only consider the sieve part of the general NFS method.

According to the result, CRYPTREC reported that future high-performance supercomputers will rise to the level of computing power required to solve RSA-1024 in one year at some future point between 2010 and 2020, as shown in Fig. 3. Improvement of factoring power poses a security threat to cryptographic algorithms based on integer factoring problem. These also force us to replace several cryptographic algorithms.

CRYPTREC also provides a projected estimate to find a collision for SHA-1. It is estimated that the complexity of the collision attack by Wang et al. is $2^{63} \sim 2^{69}$ work [23,24] and a clock cycle required for a SHA-1 operation is approximately 600 cycles/block [17]. It is also estimated that the complexity of generic second-preimage attacks is over 2^{105} work [12,4]. See Figure 4.

3.5 Migration Plan for Cryptography on PKI Systems

The necessity to replace vulnerable cryptographic techniques by more secure ones and adjust security parameters, such as the size of composite numbers in the RSA

[11] `http://www.cryptrec.go.jp/english/`

[12] `http://www.top500.org/`

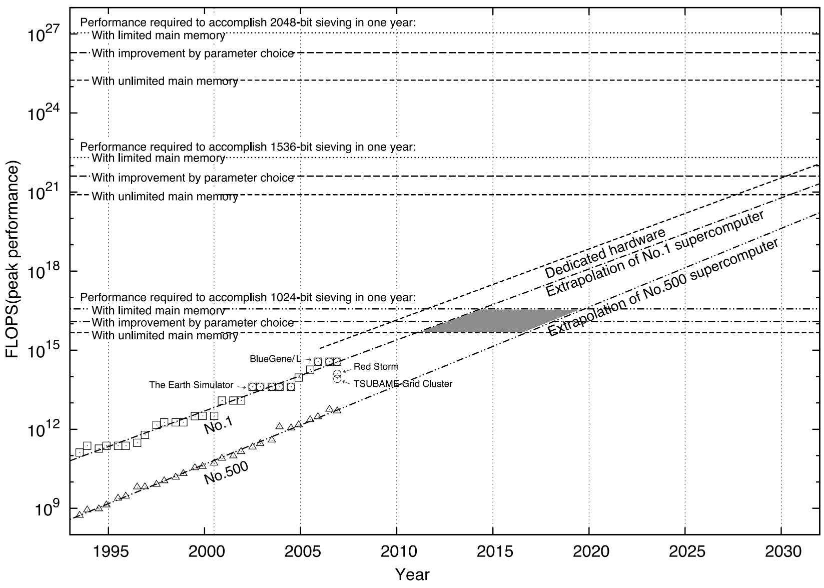


Fig. 3. Projected estimates of the GNFS sieving step

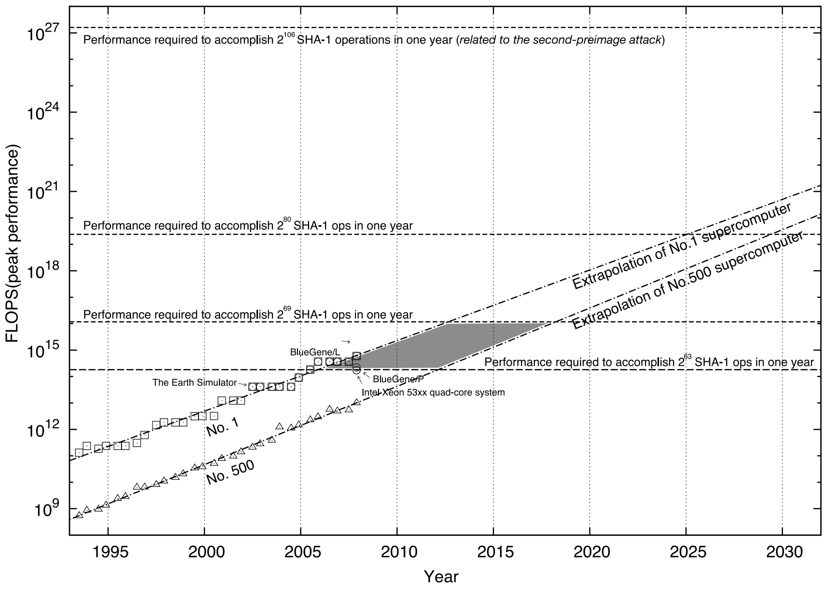


Fig. 4. Projected estimates of finding a collision of SHA-1

schemes, is not limited to the context of ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS. The computational complexity of RSA-1024 has been deteriorated steadily, and currently, the possibility of attacks for

SHA-1 is also considered simultaneously. For the case of replacement of SHA-1, It is discussed in [3]. The attack against hash functions by Wang et al. [23,24] attracted more and more attention to the security of existing hash functions. Some of them are no longer secure enough for signing. Thus, we have to replace vulnerable techniques by more secure ones.

In Japan, these algorithms, which are used in critical portion in government systems, are subject matter now. PKI systems are affected by this problem, because many functions are using cryptographic techniques. To solve this problem, long-term view is required, and we have to replace many functions and data. So, we call this plan, "Migration Plan". The PKI systems owner ministries have already recognized the problem, and they are expecting to have the migration plan. As each PKI system is operated under vertically divided administration, it will be difficult to migrate consistently. Also, to take an initiative for private sector is expected. It is planned to draw over all roadmap and schedule, and several guideline documents will be prepared.

For certificates in PKI systems, there is an operational life cycle based on the validity of certificates. (e.g.: five years) In principle, shortening the valid period of certificates is not our concern. The opportunity to change the algorithms should be the timing when the due date of certificates issued by each PKI system has come. It is expected that credentials which has more secure algorithms to be issued in order. Although there may be an opinion to insist rekeying even within the valid term, it may not be realistic. Procedures, which require excessive fiscal expenditure, will not be allowed.

The viewpoint of efficiency and cost are important in considering these migration plans. For newly issued smart card tokens and credentials in it, it should be considered to synchronize with the valid term of the cards and that of credentials to avoidable procedure and its cost. Of course, for newly issued credentials, secure algorithms should be used at the beginning. In this way, in developing migration plan for cryptography in PKI systems, efficient and economic procedure should be discussed.

3.6 Reference Cases in Europe

Government PKI Systems for Entity Authentication. The case for entity authentication function about which we have concerns is using database systems over the Internet. For example, we have a concern on Estonian X-road[13]. Although the size and flexibility of Estonian administrative system may be quite different from that of Japan, it is instructive in designing PKI systems as national infrastructure.

Citizen's Smart Card Data Format. From view point of data format and its interoperability, attentions are paid to OpenSC project[14]. Also, Belgian Personal Identity Card (BELPIC) [7] is getting attention, because BELPIC project has

[13] `http://www.riso.ee/en/information-policy/projects/x-road`

[14] `http://www.opensc-project.org/`

disclosed its middleware and applications as open source software. We believe that similar approach should be taken to confirm the interoperability in Japan.

Long-Term Archiving and Validation. According to the legislation history of ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS, we had been gazed upon movements in Germany. The concern about long-term archiving of digital documents is high in the same way in Germany.

ECOM[15], a private sector organization to promote electric commerce, had been working to develop some profiles for the long-term signature format based on RFC 3126 [20] and ETSI TS 101 733. They are still developing it in cooperation with ETSI, and domestic standard is also to be prepared.

4 Future Work: Domestic Issue and International Issue

4.1 Domestic Issue: Legislation Concerning Identification Requirement and Evidence

Although the later half of SP 800-63 in US is worth to consider adopting its scheme internationally, the former half, describing requirement for identification and its evidence, is a matter regulated by domestic law and ministerial ordinance. So, it might be impossible for a while to assume these issues as precondition.

In Japan, it is said that, there are multiple laws and regulations which define some requirements and evidences to identify a person. Actually, there is a provision in the Ministerial Ordinance of ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS, and there are other laws specific to banking industry and cellular phone industry. Here again, we can see instances of vertically divided administration. Current situation is complicated, and there may be even other laws. As social systems and rules for identification are fundamentals to PKI systems, some action will be needed to clarify the requirements and evidences.

4.2 International Issue: Migration Plan for Cryptography in PKI System

As explained above, In Japan, regard to RSA-1024 and SHA-1 in use, discussion on developing migration plan has started.

Potentially, it will become an internationally common issue. This issue requires long term view, and several guideline documents should be written to cope with this new issue. (e.g.: re-keying procedure and procedures about the insurance of roll-over certificates) And, to experiment if the interoperability of PKI systems can be kept in and after the migration, it may be necessary to set up some testing environment. It will worth sharing this kind of knowledge internationally.

Ideally, some technical scheme which can deal with the smooth change of criptographic algorithms is expected.

[15] `http://www.ecom.jp/en/index.html`

5 Conclusion

In this paper, we introduced the current situation of government PKI systems in Japan, and explained about new undertakings. Especially, we discussed on two major characteristics: Bridge Model and certificates for non-repudiation. Regarding the first characteristic, we believe that our experience of keeping interoperability is worth to be referenced by European countries. And for the second characteristic, it will be inevitable for Japan to introduce certificates for entity authentication. Then, we will refer to instructive European cases. We will continue to discuss on improving Japanese government PKI systems as a national identity management infrastructure.

References

1. Administrative Management Bureau: Government Public Key Infrastructure Interoperability Specification (in Japanese). Ministry of Internal Affairs and Communications (2001), `http://www.gpki.go.jp/session/`
2. ANS X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). Accredited Standards Committee X9, Inc. (2005)
3. Bellovin, S., Rescorla, E.: Deploying a New Hash Algorithm. In: First NIST Cryptographic Hash Workshop. National Institute of Standards and Technology (2005), `http://csrc.nist.gov/groups/ST/hash/first_workshop.html`
4. Bouillaguet, C., Fouque, P.A., Shamir, A., Zimmer, S.: Second Preimage Attacks on Dithered Hash Functions. Cryptology ePrint Archive: Report 2007/395 (2007); EUROCRYPT 2008 (to appear), `http://eprint.iacr.org/2007/395`
5. Burr, W.E., Dodson, D.F., Polk, W.T.: NIST Special Publication 800-63: Electronic Authentication Guideline. National Institute of Standards and Technology (2006), `http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf`
6. Challenge PKI Project: Project: Challenge PKI Test Suite 2.0. Japan Network Security Association, Information-technology Promotion Agency, Japan (2004), `http://www.jnsa.org/mpki/index.html`
7. Cock, D.D., Wolf, C., Preneel, B.: The Belgian Electronic Identity Card (Overview). In: Dittmann, J. (ed.) Sicherheit 2006. Lecture Notes in Informatics, P-77, pp. 298–301. Bonner Köllen Verlag (2006), `http://www.cosic.esat.kuleuven.be/publications/article-769.pdf`
8. Cryptography Research and Evaluation Committees: CRYPTREC Report 2006 (in Japanese). National Institute of Information and Communications Technology, Information-technology Promotion Agency, Japan (2007), `http://www2.nict.go.jp/y/y213/cryptrec_publicity/c06_wat_final.pdf`
9. The Japanese Act No. 102 of May 31 of 2000: The English Translation of the ACT ON ELECTRONIC SIGNATURES AND CERTIFICATION BUSINESS. Cabinet Secretatiat (2007), `http://www.cas.go.jp/jp/seisaku/hourei/data/aescb.pdf`
10. Juki-net: Smart card for Basic Resident Registration System (in Japanese). Ministry of Internal Affairs and Communications (2003), `http://www.soumu.go.jp/c-gyousei/daityo/juki_card.html`
11. Housley, R., Polk, W., Ford, W., Solo, D.: RFC 3280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. The Internet Society (2002), `http://www.ietf.org/rfc/rfc3280.txt`

12. Kelsey, J., Schneier, B.: Second Preimages on n-bit Hash Functions for Much Less than 2^n Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
13. Kleinjung, T.: Securing Cyberspace: Applications and Foundations of Cryptography and Computer Security. In: Workshop IV: Special purpose hardware for cryptography: Attacks and Applications, Institute for Pure and Applied Mathematics (2006), http://www.ipam.ucla.edu/schedule.aspx?pc=scws4
14. Kleinjung, T.: Evaluation of Complexity of Mathematical Algorithms. CRYPTREC technical report No.0601 in FY 2006. In: Cryptography Research and Evaluation Committees (2007), http://www.cryptrec.go.jp/estimation.html
15. Kleinjung, T.: Questions and answers regarding [14] (2007)
16. Lenstra, A.K., Lenstra Jr., H.W.: The development of the number field sieve. Lecture Notes in Mathematics, vol. 1554. Springer, Heidelberg (1993)
17. Nakajima, J., Matsui, M.: Performance Analysis and Parallel Implementation of Dedicated Hash Functions. In: Knudsen, L. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 165–180. Springer, Heidelberg (2002)
18. NIST FIPS PUB 140-2: Security Requirements for Cryptographic Modules. National Institute of Standards and Technology (2001), http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf
19. NIST FIPS PUB 186-2: DIGITAL SIGNATURE ALGORITHM (DSS). National Institute of Standards and Technology (2000), http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf
20. Pinkas, D., Ross, J., Pope, N.: RFC 3126, Electronic Signature Formats for long term electronic signatures. The Internet Society (2001), http://www.ietf.org/rfc/rfc3126.txt
21. RSA Laboratories: PKCS #1 v2.1: RSA Cryptography Standard. RSA Security, Inc. (2002), http://www.rsa.com/rsalabs/node.asp?id=2125
22. Shimaoka, M., Hastings, N., Nielsen, R.: Internet-Draft: Memorandum for multi-domain Public Key Infrastructure Interoperability. The IETF Trust (2008), http://www.ietf.org/internet-drafts/draft-shimaoka-multidomain-pki-12.txt
23. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
24. Wang, X., Yao, A., Yao, F.: Cryptanalysis on SHA-1. In: First NIST Cryptographic Hash Workshop. National Institute of Standards and Technology (2005), http://csrc.nist.gov/groups/ST/hash/first_workshop.html
25. Zhang, N., et al.: E-infrastructure Security: authentication Levels of Assurance (ES-LoA). The ES-LoA project (2007), http://www.es-loa.org/

A Privacy-Preserving eHealth Protocol Compliant with the Belgian Healthcare System

Bart De Decker[1], Mohamed Layouni[2], Hans Vangheluwe[2], and Kristof Verslype[1]

[1] Department of Computer Science, K.U.Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium
[2] School of Computer Science, McGill University, Montreal, Quebec, Canada

Abstract. Real world healthcare systems are generally large and overly complex systems. Designing privacy-friendly protocols for such systems is a challenging task. In this paper we present a privacy-preserving protocol for the Belgian healthcare system. The proposed protocol protects the patients' privacy throughout the prescription handling process, while complying with most aspects of the current Belgian healthcare practise. The presented protocol relies on standard privacy-preserving credential systems, and verifiable public key cryptography, which makes it readily fit for implementation.

Keywords: anonymous credentials, electronic healthcare, privacy.

1 Introduction

Healthcare represents one of the main pillars reflecting the quality of public service in our society. Over the years, countries around the world have experimented with a multitude of technical choices and policies to improve the quality of their health service. One technical choice that seems to be turning into a trend is the migration from traditional paper-based healthcare to electronic healthcare. The latter has a number of advantages. Among them we note the greater convenience and speed to access health data, which translates into shorter treatment delays, less medical errors, better statistics, higher cost-efficiency, better fraud detection mechanisms, and shorter refund delays for patients covered by health insurance plans.

Despite all the above benefits, patients around the world have shown a certain reluctance and skepticism towards new electronic healthcare systems. The reason for this skepticism is mainly attributed to the lack of assurances about the way patient data is handled, and the implications that may result from it on patients' privacy.

To help reduce this lack of trust one should design ehealth protocols with both security and privacy in mind. Due to the sensitive nature of health data, such protocols should be based on well established cryptographic primitives, and should provide defences against possible user inadvertencies such as ID card losses.

Designing protocols however without consideration for the current procedures, practices, and existing infrastructures, represents a great obstacle to the adoption of these protocols regardless of their ingenuity. This is due in part to the high costs required to change the existing infrastructure before the new system can be used. In some cases the proposed protocols require the elimination of entire parties. Sometimes these parties represent on the ground a government agency or a ministry, and removing them is simply unrealistic.

In this work, we design a protocol that protects the privacy of patients throughout the prescription handling process, while complying with most aspects of the current Belgian healthcare practice [1]. The Belgian healthcare system is a large and complex system with many players who do not necessarily share the same interests. The ehealth protocol we propose protects (1) the privacy of patients by eliminating any information leak that may harm the interests of the patient, (2) the privacy of doctors, their prescription habits, and their interactions with patients, and (3) the interests of the government by avoiding any provable evidence of a doctor's prescription behaviour, which could be sold to pharmaceutical companies for example. Moreover, our protocol has mechanisms to handle disputes and retrace fraudsters, all without changing the structure of the current Belgian healthcare practice.

Furthermore, healthcare systems with a structure similar to that of the Belgian system, can benefit from the protocol proposed in this paper modulo a few minor adaptations.

Paper Organization. First we start with related work in section 2. Then in section 3 we introduce the Belgian healthcare system. In section 4 we describe the security and privacy requirements achieved by our protocol. In sections 5 and 6 we describe the building blocks as well as the protocol we propose to achieve the previous requirements. In section 7, we evaluate the the proposed protocol. We conclude in section 8, and discuss a few ideas to extend our work.

2 Related Work

A significant amount of work related to ehealth can be found in the literature. One of the major focus points so far has been on the issue of migrating services from the paper-based setting to the electronic one. A great deal of work for instance has been dedicated to features such as semantic web and interoperability between various healthcare organizations [12,13,14,19]. Other issues have been addressed as well, such as reliability, accessibility, availability, storage integrity, and fault-tolerance [16,18].

Privacy in healthcare has also been addressed. Ateniese et al. [1] propose an ehealth protocol compatible with the healthcare system in the US. The proposed protocol provides *pseudonymous* privacy to the patients, and protects the identity as well as the prescription patterns of doctors. The patient's privacy relies on a tamper-resistant smartcard solution based on conventional public key certificates. The doctors' privacy however is based on a group signature scheme, allowing them to issue prescriptions to patients on behalf of an accredited group

[1] There are auxiliary procedures in the Belgian healthcare system that are not covered in this paper. The proposed protocol can be slightly modified to include them.

of doctors. The doctors' anonymity can be revoked by an escrow party, and all the prescriptions issued by a given doctor are linkable to each other by the insurance company. Prescription linkability is an added feature in [1], and is intended to allow insurance companies to gather statistics. The protocol we propose uses privacy-preserving credentials equipped with a selective disclosure feature, and provides stronger privacy guarantees for the patient and doctors. Moreover our protocol is more efficient than that of [1] owing to the higher performance of credential systems in comparison with group signatures.

Yang et al. [21] propose a smartcard-enabled electronic prescription system compatible with the healthcare system in the US, similar to that of [1]. They also present a signature delegation feature that allows a patient to authorize a delegate (e.g., family member) to pick up prescribed medicines, and sign a reception pad on the patient's behalf, without the patient giving his signing key to the delegate. Unlike Ateniese et al.'s construction, the scheme in [21] advocates for storing all patient health data on the smartcard in order to facilitate patient mobility, and spare doctors the burden of querying remote medical databases through an unreliable network. The smartcard in [21] is also used to store patient signing keys and certificates, as well as to compute signatures. While the smartcard paradigm is interesting in many ways, the protocol as described in [21] makes the security and privacy the patients completely dependant on the tamper-resistance of the card. Moreover, the construction in [21] is such that the identity of the pharmacist is fixed by the doctor at the time of issuing the prescription. This is clearly too restrictive from the patient's point of view, since no alternative is given if the patient cannot obtain all prescribed medicine at the designated pharmacist, or if he decides to fill his prescription at a pharmacist of his choice. Moreover, allowing doctors to designate a particular pharmacist at prescription issuing time, may result in kickback schemes between doctors and pharmacists.

In [20], Yang et al. present a password-based authentication scheme for healthcare delivery systems. The rationale behind their scheme is to allow patients to authenticate to healthcare providers using long-term short passwords, as opposed to public-key certificates which assume the existence of a public key infrastructure. It is a well known fact [4,11] however that password-based authentication systems are vulnerable to dictionary attacks. To protect against dictionary attacks, the authors in [20] propose a special network architecture with a front-end *service server* known to users, and a back-end *control server* hidden from users. To authenticate to the system, the user interacts with the service server, who in turn cooperates with the control server in order to validate the authentication request. The system in [20] is purely for authentication purposes; it provides no privacy for the patient, and does not consider issues such as controlling access to health data.

In [9], a system for privacy-preserving electronic health records is presented, which allows a patient to control who has access to her health records. Furthermore, both patient and doctor will remain anonymous towards any central authority. Since this system is also based on anonymous credentials, our system could easily be augmented with these privacy-preserving health records.

3 Brief Overview on the Belgian Healthcare System

A typical workflow in the Belgian healthcare system involves a *doctor*, a *patient*, a *pharmacist*, a *Medical Prescription Administration (MPA)*, a *Health Insurance Institute (HII)*, a public safety organization denoted *IFEB*[2], and a social security organization denoted *RIZIV*[3]. Every patient is member of one of the existing HIIs. Every pharmacist is attached to one of the existing MPAs. The latter is called the pharmacist's local MPA. An MPA processes all the prescriptions filled by its client pharmacists, and plays the role of an intermediary between pharmacists and the patients' HIIs. Similar to a router, it sorts received prescriptions by HII, and then forwards them in batch to the right HIIs.

A basic healthcare scenario can be described as follows. The patient visits a doctor and receives a prescription. The patient then takes his prescription to a pharmacist. The pharmacist checks the validity of the prescription, and charges the patient only a portion[4] of the cost. The remaining cost of the prescription will be paid for by the patient's Health Insurance Institute (HII). The pharmacist delivers the prescribed medicine to the patient, and forwards a copy of the prescription as well as an invoice to his local MPA. The MPA in turn processes the received data and forwards it to the patient's HII. The patient's HII checks the validity of the data, updates the patient's records (e.g., total medical expenses so far this year) and sends a reimbursement back to the MPA, who in turn relays it to the pharmacist.

Concurrently with executions such as the one above, the IFEB gathers statistical data from MPAs and interprets it. The IFEB also watches for fraud instances involving restricted drugs such as methadone. The RIZIV also plays a major role in the Belgian healthcare system. It finances the healthcare system by compensating the HIIs. In addition, the RIZIV oversees the overall healthcare system by retrieving and auditing sample prescriptions from the MPAs. The RIZIV is assumed to have direct access to the IFEB database.

System Model. Each player in the system above possesses a number of identity attributes. We describe the most important ones in the following.

Doctor: has a credential *DrCred* asserting that he is allowed to practise as a doctor. The Doctor has a unique identifier *DrID*, and a pseudonym *DrNym*. The correspondence between *DrID* and *DrNym* is known only to a trusted oversight authority such as the "College of Physicians". The Doctor's credential *DrCred* contains *DrID* and *DrNym* in addition to other identity attributes.

Patient: has an identifier *PtID*, and a social security status *PtSSS*. In addition, the patient has a "health expense account" *PtAcc* maintained by his HII. The latter is denoted *PtHII*. The value of *PtAcc* indicates the amount the patient has spent sofar in the current year on

[2] "Instituut voor farmaco-epidemiologie van België" in Dutch.
[3] "Rijksinstituut voor Ziekte- en Invaliditeitsverzekering" in Dutch.
[4] The size of this portion is determined by the patient's social security status.

health expenses. Admissible health expenses charged to the patient beyond a predetermined maximum amount will be covered by the HII. Finally, the patient has a pseudonym *PtNym*. The correspondence between *PtID*and *PtNym* is known only to the patient's HII. In summary, the patient's credential contains the attributes {*PtID, PtNym, PtHII, PtSSS, PtAcc, ...*}

<u>Pharmacist:</u> has an identifier *PharmID*, and a corresponding MPA denoted *PharmID_MPA*. The pharmacist's credential contains a number of attributes including *PharmID* and *PharmID_MPA*.

<u>MPA:</u> has a publicly known identifier *MPA_ID*, and a credential certifying its identity. The MPA serves a set of pharmacists, and generates statistics on prescription data on request from authorized organizations such as IFEB.

<u>HII:</u> has a publicly known identifier *HII_ID*, and a credential certifying its identity. The HII maintains the health expense accounts *PtAcc* of affiliated patients, and covers their admissible medical expenses.

<u>IFEB:</u> has a publicly known identifier *IFEB_ID*, and a credential certifying its identity. It gathers statistics, and conducts studies on public safety.

<u>RIZIV:</u> has a publicly known identifier *RIZIV_ID*, and a credential certifying its identity. it performs various oversight activities, and controls organizations such as IFEB.

4 Requirements

In this section, we discuss the main security and privacy properties we want to achieve in the proposed ehealth protocol. The functional requirements can be easily derived from the workflow described the previous section.

4.1 Security Requirements

General Security Requirements

- **Entity authentication (S1).** All parties should be able to properly authenticate each other. No party should be able to succeed in claiming a false identity, or false information about his identity.
- **Item integrity (S2).** Transcripts generated during the prescription lifecycle cannot be tampered with, without being detected with an overwhelming probability.
- **Revocability (S3).** It should be possible to revoke the credentials as well as the anonymity/pseudonimity of abusing parties.

Security Requirements Specific to the Belgian Healthcare System

- **Multiple prescription issuance detection capability (D1).** Oversight authorities such as the RIZIV should be able to detect malicious patients

who visit multiple doctors for the same illness in order to get multiple prescriptions of a particular drug.
- **Single prescription spending (D2).** A patient must not be able to fill the same prescription multiple times.
- **Prescription non-transferability (D3).** It should not be possible for a party to fill a prescription, if he is not the patient to whom the prescription was originally issued.
- **Inappropriate prescribing patterns detection capability (D4).** It should be possible to detect doctors who systematically prescribe expensive drugs (instead of generic, and hence, cheaper ones), or doctors who prescribe significantly more drugs of a certain type (e.g. antibiotics) despite known counter-indications etc. In such cases, the doctors involved might be served a warning, or an investigation might be initiated.
- **Correct pharmacist reimbursement (D5).** A pharmacist who is not correctly refunded by the MPA, should be able to prove it in order to be compensated.
- **Payment fraud detection capability (D6).** The pharmacist should be refunded only if he has indeed delivered the medicine to the patient. It should be possible to detect pharmacists who claim expenses for non delivered medicine.
- **Correct statistics (D7).** The IFEB must be ensured that the received statistics are correct.

4.2 Privacy Requirements

- **Minimum disclosure (P1).** During a medical consultation, the patient and doctor should be able to selectively (and provably) reveal to each other any property or predicate about their respective identities. In addition, parties involved in the prescription processing workflow should not be able to learn any information about the patient and doctor except what the latter willfully disclose to them. Data exchanged during the ehealth protocol execution should satisfy the access control requirements defined in table 1.
- **Patient unlinkability (P2).** Prescriptions issued to the same patient should not be linkable to each other, except by the patient's HII, or by the doctor (if the patient accepts to reveal such information to the doctor.) On the other hand, two patient prescriptions that cross the same MPA should be linkable to each other, but not to the patient's identity.
- **Patient untraceability (P3).** No party involved in the prescription workflow, except the HII and RIZIV, should be able to determine the identity of the patient. The RIZIV identifies patients only in case of abuse.
- **Absence of provable doctors' prescription behaviour (P4).** To prevent elicit kickbacks and bribery between doctors and pharmaceutical companies, pharmacists should not be able to provide evidence to pharmaceutical companies about doctors' prescription behaviour.

Table 1. Access control matrix

Party\Data	**Patient**	**Presc.**	**Doctor**	**Pharm.**	**MPA**	**HII**
Patient	ID (trivial)	all content	ID	ID	ID	ID
Doctor	nym	PrescID, data (trivial)	ID (trivial)	—	—	—
Pharm.	ss status	data	ID (if anomaly)	ID (trivial)	ID	—
MPA	nym, ss status	PrescID, data	nym	ID	ID (trivial)	ID
HII	ID	PrescID, cost	—	—	ID	ID (trivial)
IFEB	nym, ss status etc.	anon. stat. data	nym	geog. location	—	—

5 Building Blocks: Brief Overview

5.1 Commitments

A commitment scheme [17,10] allows a committer to hide a set of attributes inside a token, also called commitment. Later the committer can open the commitment by revealing the underlying attributes. The former phase is called the commitment phase, while the latter is called the opening phase. The commitment scheme is such that the committer cannot open the commitment to a set of attributes that is different from the one embedded in the commitment phase.

Notation. For a commitment *comm* with attributes $(x_1, \cdots, x_p)$, the expression $comm.x_j$ denotes the j^{th} attribute embedded in *comm*. To further conceal the values of the attributes underlying a commitment, one of the embedded attributes can be chosen at random and used as a blinding factor. A commitment can be opened by revealing the attributes in it. The latter is called opening information, and denoted *openInfo*.

5.2 Digital Credentials

A digital credential issued to user U is typically a set of assertions made by an certification authority about the identity attributes of U. To be viable, a credential system should satisfy a number of security properties such as unforgeability, and integrity. These properties are further discussed below. The X.509 public key certificate standard [15] is a well known example of digital credentials.

Privacy-preserving digital credentials (e.g., [5,6,7]) represent a more elaborate type of credentials, also referred to as *anonymous* credentials. In addition to the usual security properties necessary for traditional digital credentials, privacy-preserving credential systems possess a number of properties intended specifically to protect the identity of honest credential holders. Among these we note *selective disclosure*, *token untraceability*, *tokens unlinkability*, *multi-show unlinkability*, *limited-show untraceability*, and *signed audit trails* [5,6,7]. Privacy-preserving credentials are used as a major building block in this paper.

We distinguish three types of participants in a privacy-preserving credential system:

(1) An *issuer*, generally a recognized certification authority, who issues credentials to users in an *issuing protocol.*
(2) A *user*, to whom credentials are issued. The user, also referred to as the credential holder, shows his credentials, in a *showing protocol*, to third parties in exchange for goods and services. The user can selectively reveal any information about any subset of the attributes underlying his credential. The credential showing can be turned into a non-interactive signed proof. The resulting transcript can be used then as a *signed audit trail.* One desirable feature of this type of credentials is *token untraceability.* This feature ensures that no party, including the issuer can link a showing transcript to the identity of the credential holder. When different credentials owned by the same user are unlinkable to each other, we say that the credential system satisfies *token unlinkability.* When multiple showings of the same credential are not linkable to each other we say that we have *multi-show unlinkability.* The *limited-show untraceability* property is achieved when the identity of the credential remains hidden as long as the credential is not shown more than a predefined maximum number of times.
(3) A *verifier* to whom the user shows his credential. The verifier may later *deposit* the showing transcript at the credential issuer, for instance to redeem e-coins in the context of ecash. The latter protocol is called a *depositing protocol.*

Notation. For a credential *Cred* with attributes $(a_1, \cdots, a_n)$, the expression *Cred.*a_ℓ denotes the ℓ^{th} attribute of *Cred.* For example if we assume that the Doctor has an anonymous credential denoted *DrCred*, then *DrCred.ID* and *DrCred.exp* denote the identifier and expiry date of *DrCred* respectively.

Let A be a party holding an anonymous credential *Cred* and commitment *comm* encoding attributes $(a_1, \cdots, a_n)$ and $(x_1, \cdots, x_p)$ respectively. Party A can selectively disclose any information about the attributes underlying *Cred* and *comm.* Given (1) a predicate $\mathcal{P}$ on attributes $(a_1, \cdots, a_n)$ and $(x_1, \cdots, x_p)$, and (2) a message m, the expression $\text{SPK}\{\mathcal{P}(a_1, \cdots, a_n, x_1, \cdots, x_p)\}(m)$ denotes a signed proof of knowledge on message m, of attributes $a_1, \cdots, a_n$, $x_1, \cdots, x_p$ underlying *Cred* and *comm* respectively, and satisfying predicate $\mathcal{P}$. The expression $\text{SPK}\{\textit{comm.DrID} == \textit{DrCred.ID} \wedge \textit{DrCred.exp} \geq \text{today}\}(m)$ for example, denotes a signed proof of knowledge on message m, where the prover convinces a verifier that (1) he knows all the attributes underlying *comm* and *DrCred*, (2) that the *ID* embedded in *DrCred* is the same as the one embedded in *comm*, and (3) that credential *DrCred* has not expired yet.

5.3 Verifiable Encryption

A verifiable encryption scheme (e.g., [8]) for a relation R is a protocol that allows a prover to convince a verifier that a ciphertext is an encryption of a value w under a given public key such that w satisfies R, and no other information about w is disclosed. In a verifiable encryption scheme, the ciphertext is checked with respect to a public key associated with a known "decryptor".

Notation. The expressions $\mathit{VEnc}_A(\cdot)$ and $\mathit{Enc}_B(\cdot)$ denote the verifiable encryption under party A's public key, and the conventional public-key encryption under party B's public key respectively.

Let $\mathcal{M}$ be the message space of $\mathit{VEnc}(\cdot)$ the verifiable encryption scheme, and let $\mathcal{P}$ a boolean predicate on $\mathcal{M}$. The expression

$$vc = \mathit{VEnc}_{\text{RecID}}(m)\{\mathcal{P}(m)\}$$

denotes the verifiable encryption of m under the public key of RecID, the intended recipient. Given the public key of RecID, any verifier can be convinced that vc is an encryption under RecID' public key of a non-disclosed message that satisfies predicate $\mathcal{P}$.

6 The Proposed Protocol

6.1 Setting

Based on the system model and requirements described in Sections 3 and 4, we made a number choices regarding the type of credentials needed by each participant involved in the ehealth protocol. The patients and doctors are widely considered as private entities with high expectations of privacy; we therefore equip them with anonymous credentials. The other parties however are all public entities; it is sufficient to simply identify them with conventional X.509 public key certificates. These choices are summarized in Table 2.

The credentials of the MPAs, HIIs, RIZIV, IFEB, and pharmacists are issued by trusted government-approved certification organizations. The doctors' credentials are issued by a medical certification authority such as the college of physicians. The patients' credentials are issued by a central government-approved certification authority CA. The patient's pseudonym *PtNym* embedded in the patient's credential is not known to CA. The correspondence between *PtNym* and *PtID* is known only to the patient's HII. Issuing anonymous credentials on secret but committed attributes is easily done by standard techniques such as those in [5,6].

Table 2. Credential material per participant

Party / Cred. type	Patient	Dr.	Pharm.	MPA	HII	IFEB	RIZIV
Anon. Cred.	✓	✓					
X.509 Cert.			✓	✓	✓	✓	✓

6.2 Protocol Description

I. Doctor (Dr.) $\leftrightarrow$ Patient (Pt.)

(a) Dr. anonymously authenticates to Patient using his *DrCred*.

(b) Patient computes commitment $com_{Pt} := comm(PtID)$,

(c) Patient anonymously authenticates to Doctor using his credential *PtCred*. Moreover, Patient sends com_{Pt} to Doctor, and proves that $com_{Pt}.PtID == PtCred.PtID$

(d) Dr. computes commitment $com_{Dr} := comm(DrNym)$

(e) Dr. sets $Presc\_text :=$ {plain prescription text}

(f) Dr. computes the prescription's serial number *PrescID*, e.g., as a hash of *Presc_text*, com_{Pt}, and com_{Dr}.

(g) Dr. computes

$$\text{Presc} := \text{SPK}\{DrCred.DrNym == com_{Dr}.DrNym\}(Presc\_text, PrescID, com_{Dr}, com_{Pt}),$$

and sends it to the patient, along with the opening information of com_{Dr}.

II. Patient $\leftrightarrow$ Pharmacist

(a) Pharmacist authenticates to Patient using his X.509 pharmacist certificate *PharmCred*.

(b) Pt. recovers *PharmCred.MPA_ID*, the identity of the MPA serving the pharmacist.

(c) Pt. anonymously authenticates to Pharmacist using *PtCred*, and provably discloses his social security status.

(d) Pt. computes:

i. $vc_1 = VEnc_{\text{MPA}}(PtHII)\{PtHII = PtCred.PtHII\}$

ii. $vc_2 = VEnc_{\text{MPA}}(DrNym)\{DrNym = Presc.com_{Dr}.DrNym\}$

iii. $vc_3 = VEnc_{\text{RIZIV}}(PtNym)\{PtNym = PtCred.PtNym\}$

iv. $vc_3' = VEnc_{\text{RIZIV}}(PtHII)\{PtHII = PtCred.PtHII\}$

v. $vc_4 = VEnc_{\text{MPA}}(PtNym)\{PtNym = PtCred.PtNym\}$

vi. $vc_5 = VEnc_{\text{PtHII}}(PtNym)\{PtNym = PtCred.PtNym\}$

vii. $c_5 = Enc_{\text{MPA}}(vc_5)$

(e) Pt. sends to pharmacist:

i. Presc. and SPK$\{PtCred.PtID == Presc.com_{Pt}.PtID\}$(nonce)[5]

ii. $vc_1, vc_2, vc_3, vc_3', vc_4, c_5$ [6]

[5] The nonce can be chosen jointly by the patient and pharmacist, and may include information such as the date, *PharmID* etc.

[6] The patient Pt. sends c_5 to the pharmacist instead of vc_5, because Pt. wants to hide the identity of his HII from the pharmacist. In Belgium, health insurance institutes (HIIs) are managed by socio-political groups, and revealing the identity of a patient's HII, may disclose personal information about the patient's political inclination for example. That is why in the protocol above, the patient hides the identity of his HII from the pharmacist. Only the MPA (downstream) needs to know the identity of the patient's HII. The correctness of $vc_5 = Dec_{\text{MPA}}(c_5)$ will be checked by the MPA, prior to forwarding it to the right HII. Additional data that may be useful for statistics, such as PtAge, can be handed to the MPA inside vc_4.

(f) Pharmacist checks if Presc., SPK, and $vc_1, vc_2, vc_3, vc'_3, vc_4$ are correct. If all is correct then continue, else abort. If Presc. contains an anomaly (e.g. unusual or possibly lethal dosage), the pharmacists asks Pt. to name the doctor. The pharmacist will contact the doctor to correct the problem.

(g) Pharmacist charges patient, gets payed, and delivers drug.

(h) Pharmacist issues an invoice to Patient with the prescription's serial number *PrescID* embedded in it.

(i) Patient computes:
reception_ack := SPK$\{PtCred\}(PrescID,$
$PharmID, vc_1, vc_2, vc_3, vc'_3, vc_4, c_5)$,
and sends it to Pharmacist. This proves that the patient has indeed received the medicine from the pharmacist.

(j) Pharmacist checks if reception_ack is correct. If correct continue, else abort.

III. Pharmacist $\leftrightarrow$ MPA (*PharmCred.MPA_ID*)

(a) Pharmacist and MPA mutually authenticate

(b) Pharmacist forwards to MPA Presc., $vc_1, vc_2, vc_3, vc'_3, vc_4, c_5$, and reception_ack.

(c) If all is correct, the MPA continues. Else if $Dec_{\mathrm{MPA}}(c_5)$ is incorrect, then forward vc_3, vc'_3, and rest of transcript to RIZIV and request patient deanonymization.[7]

(d) MPA computes:
  i. $PtNym = Dec_{\mathrm{MPA}}(vc_4)$,
  ii. $PtHII = Dec_{\mathrm{MPA}}(vc_1)$,
  iii. $DrNym = Dec_{\mathrm{MPA}}(vc_2)$,
  iv. $vc_5 = Dec_{\mathrm{MPA}}(c_5)$

(e) MPA adds a DB entry indexed by *PrescID*, *PtNym*, *DrNym*, and stores any information relevant to the prescription.

IV. MPA $\leftrightarrow$ HII (*PtHII*)

(a) MPA and HII mutually authenticate

(b) MPA forwards reception_ack and vc_5 to the patient's HII

(c) HII checks the integrity of reception_ack and vc_5

(d) If correct, HII recovers $PtNym = Dec_{\mathrm{HII}}(vc_5)$, else abort and forward transcript to RIZIV for patient deanonymization.

(e) HII recovers *PtID* corresponding to *PtNym*

(f) HII updates patient *PtID*'s account *PtAcc* with proper amount

(g) HII sends reimbursement amount due to the MPA, along with the corresponding invoice containing *PrescID*.

(h) HII creates a database entry for the processed invoice with information such as *PtID*, *PrescID*, prescription cost, date etc.

(i) After receiving the refund from the HII, the MPA compensates the pharmacist.

[7] The RIZIV first recovers *PtNym* and *PtHII*) from vc_3 and vc'_3), then files a complaint with the judicial authorities who can subpoena the HII to provide the real identity of the fraudulent patient.

V. IFEB $\leftrightarrow$ MPA
   (a) MPA and IFEB mutually authenticate
   (b) IFEB requests statistics
   (c) MPA provides statistics on prescription data anonymized according to the privacy laws in place.

   The data available to the MPA is identified only by Doctor and Patient pseudonyms. This data is sufficient to generate meaningful statistics, including measurements requiring the aggregation of prescription data per patient or per doctor. The data available to the MPA, and the subsequently released statistics do not compromise the real identities of patients or doctors.

   Alternatively, the IFEB can obtain statistics from the HIIs. This can be done without weakening the privacy of the patient or inducing additional disclosures, since the HIIs already know the prescription data of their affiliated patients. The IFEB first queries the different HIIs for a specific statistical measurement, and then aggregates the separate *anonymized* results to derive the global measurement for the whole population. Data from the HIIs can also be used to double-check the accuracy of statistics collected from the MPAs.

Remarks

– In step I-(g) the Doctor computes the prescription as a signed proof of knowledge on the tuple ($Presc\_text$, $PrescID$, com_{Dr}, com_{Pt}). The predicate being asserted in the proof is that com_{Dr} contains the same attribute $DrNym$ embedded in $DrCred$. This results in the following observations:
  - Because the prescription is a signed proof, any one can check its validity non-interactively.
  - The prescription is tied via (com_{Dr}, com_{Pt}) to the identity of both the Doctor and the Patient. Recall that the Doctor issues the prescription only if the value of $PtID$ underlying com_{Pt} is consistent with $PtCred$ (the consistency proof was performed by the Patient in step I-(c).)
  - The Doctor discloses the opening information of com_{Dr} to the patient, to allow him to verifiably encrypt $DrNym$ under the public key of the pharmacist's MPA (in step II-(d-ii).) Note that the Doctor cannot encrypt $DrNym$ in advance since the identity of the pharmacist where the patient will buy his drugs is usually not known at the time of the prescription issuing.

7 Protocol Evaluation

In the following we provide proof sketches and arguments supporting the security of our protocol. we assume all underlying building blocks secure. A more formal and complete analysis will be given in the full version of the paper.

7.1 General Security Requirements

- **Entity authentication (S1).** This property follows immediately from the soundness and unforgeability of the underlying anonymous and public key certificates.
- **Item integrity (S2).** All binding data (e.g., prescription, acknowledgement, verifiable encryptions) exchanged during the protocol of Section 6, are signed either by a conventional public key signature or signed proof of knowledge, and are therefore resistant to any tampering.
- **Revocability (S3).** In case of abuse (which can be detected either by the MPA, the patient's HII, or the RIZIV), the user's identity is unveiled by opening one of the verifiable encryptions vc_3, vc_4, or vc_5. It is then possible to revoke the patient's credentials and prescriptions through blacklisting.

7.2 Security Requirements Specific to the Belgian Healthcare System

- **Multiple prescription issuance detection capability (D1).** When filling a prescription, the patient reveals information that will allow the MPA to recover his pseudonym (cf. step II-(d-iv)). Because multiple prescriptions issued to the same patient are linked to each other through the patient's pseudonym, oversight organizations such as the RIZIV or IFEB are able to detect abusive behaviour and stop malicious patients.
- **Single prescription spending (D2).** Follows from the fact that prescriptions are uniquely identified by a *PrescID*, and resistant to tampering.
- **Prescription non-transferability (D3).** This Follows from the soundness of the signed proofs of knowledge in step II-e of the protocol. The patient proves to the pharmacist that the nym in the prescription corresponds to the nym in its *PtCred*.
- **Prescription fraud detection capability (D4).** This can only be detected by the RIZIV by searching for abnormal behaviour in the IFEB database. The IFEB database contains only doctor pseudonyms, which can be linked to doctors' real identities with the help of an authority such as the "college of physicians".
- **Correct pharmacist reimbursement (D5).** For each prescription, the patient generates a *reception_ack*, which is a patient confirmation of provided services by the pharmacist. This proof is verified and stored by the pharmacist, MPA and HII. If something goes wrong, this proof can be used as evidence.
- **Payment fraud detection capability (D6).** The reception acknowledgement *reception_ack* issued by the patient guarantees to the HII that the patient has indeed received the medicine.
- **Correct statistics (D7).** The IFEB needs to rely on the trustworthiness of the MPAs to make sure it receives correct statistics. The latter property cannot be enforced by cryptographic means alone, since a malicious MPA could just ignore half of the transactions it has recorded. For better assurances, oversight organizations (e.g., RIZIV) in practice request random

sample data from health insurance institutes (HIIs) and cross-check them against data returned by the MPAs. A malicious MPA who fails to return consistent data, or returns incomplete data, will be further investigated and may have its licence revoked.

7.3 Privacy

- **Minimum disclosure (P1).** Owing to the selective disclosure feature offered by the zero knowledge proofs of knowledge, the security of the commitment and verifiable encryption schemes, the protocol of section 6 satisfies the access control requirements of table 1. This can be easily verified by simple examination of the protocol. Due to space limitation we leave the more formal proof of this property to the full version of the paper.
- **Patient unlinkability (P2).** Prescriptions are tied to the patient's pseudonym *PtNym* which can be recovered only by the MPA processing the prescription and the patient's health insurer (PtHII). All other parties have no access to the patient's identity or pseudonym, and thus cannot link any two prescriptions of the same patient. In the case of a treating doctor, the patient may freely decide to disclose his pseudonym to allow the linkability.
- **Patient untraceability (P3).** An examination of the protocol of section 6 shows that the identity of the patient is accessible only to the patient's HII who knows the correspondence between *PtNym* and *PtID*. In case of apparent abuse, the RIZIV may also have access to the patient's identity by filing a complaint with the judicial authorities who can subpoena the HII to deanonymize the fraudulent patient.
- **Absence of provable doctor's prescription behaviour (P4).** The protocol is designed in such a way that the real identity of a doctor is never associated with the prescriptions' content. The only exception occurs when the pharmacist sees a prescription anomaly (e.g. lethal dosage), in which case he asks the patient to name the doctor. This information however is not a reproducible proof, and thus cannot be used to convince a bribe-giver.

8 Concluding Remarks

The paper presents a privacy-preserving protocol for the Belgian healthcare system. The proposed protocol protects patients' privacy throughout the prescription handling process, while complying with the current Belgian practise. Despite the large number of parties involved, and the complexity of the application, the protocol we present minimizes information disclosure and satisfies the access control requirements of table 1. Furthermore, our protocol is equipped with a set of abuse detection and evidence gathering mechanisms that allow oversight authorities to solve instances of fraud and ensure accountability. In addition to protecting patients' privacy, our protocol provides a mechanism to prevent the intrusive monitoring of doctors' prescription patterns. The ability of third party players to determine the prescription patterns of a given doctor is often considered an undesirable aspect in healthcare, since it can be used by malicious pharmaceutical companies for example, (1) as a coercive tool against doctors who do not prescribe their products, or (2) as an instrument to facilitate bribery and kick-backs with doctors who do the opposite (cf., [3,2].) In our protocol, doctors are only pseudonymously identified to allow the legitimate gathering of statistical data about medicine consumption and its effect on the population. The real identity of the doctors is unveiled only in case of apparent abuse via a judicial procedure.

The design we propose in this paper is highly modular and can be adapted to other healthcare systems comparable to the Belgian one. For example, if we consider a jurisdiction where the real identity of the doctor (as opposed to his pseudonym) has to be indicated in plaintext in all transcripts generated during the prescription lifecycle, then one can easily adapt our protocol to the new setting by replacing the *DrNym* attribute in the authentication step of phase I, with the *DrID* attribute already embedded in the doctor's credential. The rest of the protocol can be easily modified accordingly.

Further improvements can be made to our protocol. For example one could strengthen access control to health records stored on remote databases by enforcing privacy policies defined by the patients. Another worthy avenue for future work would be to simplify the prescription workflow and reduce interactions (to the extent acceptable by the healthcare procedures and practices in place).

Acknowledgments. This research is a contribution to the European PRIME project and is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, the Research Fund K.U.Leuven and the IWT-SBO project (ADAPID) "Advanced Applications for Electronic Identity Cards in Flanders". The second author was previously funded by the Universitary Mission of Tunisia in North America.

References

1. Ateniese, G., de Medeiros, B.: Anonymous e-prescriptions. In: Jajodia, S., Samarati, P. (eds.) WPES, pp. 19–31. ACM, New York (2002)
2. Biovail faces heart drug kickback inquiry. Pharma Marketletter (September 1, 2003), `http://goliath.ecnext.com/coms2/summary_0199-3378487_ITM`.
3. Grand jury probes biovail over sales practices. The Toronto Star (Feburary 1,2008), `http://www.thestar.com/Business/article/299682`
4. Bellovin, S.M., Merritt, M.: Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In: ACM Conference on Computer and Communications Security, pp. 244–250 (1993)
5. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. The MIT Press, Cambridge (2000)
6. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
7. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)

8. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
9. Demuynck, L., De Decker, B.: Privacy-preserving electronic health records. In: Dittmann, J., Katzenbeisser, S., Uhl, A. (eds.) CMS 2005. LNCS, vol. 3677, pp. 150–159. Springer, Heidelberg (2005)
10. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
11. Gong, L., Lomas, T.M.A., Needham, R.M., Saltzer, J.H.: Protecting poorly chosen secrets from guessing attacks. IEEE Journal on Selected Areas in Communications 11(5), 648–656 (1993)
12. Health level 7 (hl7), `http://www.hl7.org/`
13. Hl7 reference information model, `http://www.hl7.org/library/data-model/RIM/modelpage_non.htm`
14. Integrating the healthcare enterprise, `http://www.ihe.net/`
15. ITU-T. Public-key and attribute certificate frameworks – X.509 Recommendation (2005), `http://www.itu.int/rec/T-REC-X.509/en`
16. Krummenacher, R., Simperl, E.P.B., Nixon, L.J.B., Cerizza, D., Della Valle, E.: Enabling the european patient summary through triplespaces. In: CBMS, pp. 319–324. IEEE Computer Society, Los Alamitos (2007)
17. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
18. Tavena, S., Palanque, P., Basnyat, S., Winckler, M.A., Law, E.: Clinical application design: Task modeling with failure in mind. In: World Congress on Internet in Medicine (MedNet), Toronto, Canada (2006)
19. Della Valle, E., Gadda, L., Perdoni, V.: COCOON: Building knowledge driven and dynamically networked communities within european healthcare systems (April 06, 2005)
20. Yang, Y., Deng, R.H., Bao, F.: Fortifying password authentication in integrated healthcare delivery systems. In: ASIACCS 2006: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, pp. 255–265. ACM, New York (2006)
21. Yang, Y., Han, X., Bao, F., Deng, R.H.: A smart-card-enabled privacy preserving e-prescription system. IEEE Transactions on Information Technology in Biomedicine 8(1), 47–58 (2004)

Fast Point Decompression for Standard Elliptic Curves

Billy Bob Brumley[1] and Kimmo U. Järvinen[2]

[1] Department of Information and Computer Science,
Helsinki University of Technology, P.O.Box 5400, FIN-02015 TKK, Finland
`billy.brumley@tkk.fi`
[2] Department of Signal Processing and Acoustics,
Helsinki University of Technology, P.O.Box 3000, FIN-02015 TKK, Finland
`kimmo.jarvinen@tkk.fi`

Abstract. Many standard elliptic curves (e.g. NIST, SECG, ANSI X9.62, WTLS, ...) over the finite field $\mathbb{F}_p$ have p a prime of Mersenne-like form—this yields faster field arithmetic. Point compression cuts the storage requirement for points (public keys) in half and is hence desirable. Point decompression in turn involves a square root computation. Given the special Mersenne-like form of a prime, in this paper we examine the problem of efficiently computing square roots in the base field. Although the motivation comes from standard curves, our analysis is for fast square roots in any arbitrary Mersenne-like prime field satisfying $p \equiv 3 \pmod 4$. Using well-known methods from number theory, we present a general strategy for fast square root computation in these base fields. Significant speedup in the exponentiation is achieved compared to general methods for exponentiation. Both software and hardware implementation results are given, with a focus on standard elliptic curves.

Keywords: elliptic curve cryptography, square roots modulo p, exponentiation, addition chains.

1 Introduction

Point compression is a frequently used operation in elliptic curve cryptography (ECC). Public keys (or any point on the curve) consisting of an x and y-coordinate can be compressed to only the x coordinate and some sign or compression bit, requiring only half the space. Given an x-coordinate, point decompression calculates the y-coordinate by using the curve equation; in the case of elliptic curves over a prime field $\mathbb{F}_p$, a square root must be computed. ECC is an attractive choice for small devices due to the reduced size of keys and signatures. There are many existing standards for ECC, for example [1,2,3,4]. Standard curves [5,4,3,6] have been constructed for use with elliptic curve cryptosystems, and such curves are often used in industry standards. For example, the curve P-192 [5] is used in the new Bluetooth Simple Pairing Protocol [7]. Popular software packages often also include these standard curves, for example

OpenSSL [8]. As there are many advantages for protocols making use of ECC to use point compression and decompression, and such operations are often done online, the speed is important.

This work is motivated by point decompression for those standard curves over $\mathbb{F}_p$ which satisfy $p \equiv 3 \pmod 4$ of Mersenne-like form. Hence much of the analysis focuses on square roots in the specific fields listed in these standards, although a more general analysis of Mersenne-like primes is provided as well. We present a fast method for calculating square roots by exponentiation in these fields, which is significantly faster than general exponentiation algorithms. The method is based on well-known equations from number theory that are often used in other areas of cryptography.

We begin in Sec. 2 with an overview of elliptic curves and square roots in prime fields. Addition chains are covered in Sec. 3 and some notation is introduced. An overview of fast exponentiation methods is provided as well. We present our strategy for computing square roots in Mersenne-like prime fields in Sec. 4, which yields significant speedup when compared to classic exponentiation methods. Software and hardware implementation results are provided in Sec. 5. We conclude in Sec. 6.

2 Elliptic Curves over Prime Fields

Over $\mathbb{F}_p$ for some prime $p > 3$, an elliptic curve $E(\mathbb{F}_p)$ is defined by the equation

$$y^2 = x^3 + ax + b. \tag{1}$$

An abelian group is formed using all of the points on the curve, and hence elliptic curve cryptosystems can be implemented. For example, Diffie-Hellman key agreement [9]: given a generator point G of large prime order r, Alice generates a random $a \in [1, r)$ and sends the point aG to Bob. Bob generates a random $b \in [1, r)$ and sends the point bG to Alice. The shared secret is then $a(bG) = b(aG)$, and commonly the result is then mapped (a projection) to the x-coordinate to arrive at a shared key.

For the sake of storage efficiency, Miller [10] noted that both the x and y-coordinate of a point need not be transmitted; given an x-coordinate, there are either two or zero solutions for y in (1) computed using a square root operation. The negative of a point $P = (x, y) \in E(\mathbb{F}_p)$ is the reflection on the x-axis; that is, $-P = (x, -y)$. Denoting P_x as the x-coordinate of the point P, clearly $P_x = (-P)_x$ and $(aP)_x = a(-P)_x$. If it is necessary for the point to be uniquely determined, a simple compression or sign bit can be used to identify which solution for y to use. As point decompression is often done online, the computational efficiency of this square root operation is important.

Clearly the main advantage is that storing x requires half the space of (x, y). There are also some security advantages. Public keys should be validated before use to prevent small subgroup attacks and false curve attacks. Validation involves verifying that a point does indeed lie on the curve and have large order. Unlike the multiplicative group of integers modulo p which always yield trivial subgroups, elliptic curve groups can be of prime order, and indeed most standard curves over prime fields are. Given an x-coordinate for point decompression, if a valid square root for the righthand side of (1) is found, (x, y) is guaranteed to be on $E(\mathbb{F}_p)$, preventing false curve attacks. If the order r of E is prime, then (x, y) is also guaranteed to be of order r, preventing small subgroup attacks. Such validation can, of course, be obtained given an uncompressed point by verifying that (1) holds, but when using point decompression, public key validation can be obtained naturally.

2.1 Square Roots Modulo p

Computing a y-coordinate from an x-coordinate using (1) involves computing a square root modulo p. To state the problem formally, denoting QR_p as the set of all quadratic residues modulo p, solve

$$\alpha^2 = \beta \pmod p, \text{ where } \beta \in QR_p \tag{2}$$

for α. The simplest case is $p \equiv 3 \pmod 4$. Euler's Criterion ensures that

$$\begin{aligned} \beta^{(p-1)/2} &\equiv 1 \pmod p, \text{ so} \\ \alpha^2 &\equiv \beta\beta^{(p-1)/2} \equiv \beta^{(p+1)/2} \pmod p \text{ and} \\ \alpha &\equiv \pm\beta^{(p+1)/4} \pmod p \end{aligned} \tag{3}$$

We focus on this case and disregard the remaining case of $p \equiv 1 \pmod 4$. For standard curves, $p \equiv 3 \pmod 4$ is most often the case. For example, OpenSSL supports 27 standard curves over $\mathbb{F}_p$ and 24 of them satisfy $p \equiv 3 \pmod 4$.

2.2 Mersenne-Like Prime Fields

By definition, the concept of Mersenne-like primes is by no means precise. A Mersenne number is of the form $2^d - 1 = \sum_{i=0}^{d-1} 2^i$ and a Mersenne-like prime generally contains a few more terms and/or a small constant. For a more general analysis, we will often refer to a Mersenne-like prime having the form of (4) where $a \geq 1$, $b > a + 1$, $c \geq b$, ...

$$p = \sum_{i=0}^{a} 2^i + \sum_{i=b}^{c} 2^i + \cdots \equiv 3 \pmod 4. \tag{4}$$

Although the binary representation of any arbitrary p can be described by such a form (a series of ones followed by a series of zeros followed by ...), we can identify Mersenne-like primes as those having a small number of summation terms, where "small" is left to interpretation. We will also consider slightly less general Mersenne-like primes having the form of (5).

$$p = 2^d - 2^j - 1 \equiv 3 \pmod 4 \text{ where } 1 < j < d - 1 \tag{5}$$

Mersenne-like form is desirable as it makes finite field arithmetic, such as modular reduction, more efficient; instead of expensive divisions, modular reduction can be performed using a small number of shifts, additions, and/or small multiplications. Table 1 contains some of the Mersenne-like prime fields for standard elliptic curves from various standards including NIST [5], SECG [4], ANSI X9.62 [3], and WTLS[6]. Clearly all of these primes can be described by using (5) and/or (4) as well. There are overlaps in these standards, and some of the standard curves use the same p but different curve coefficients. OpenSSL includes all of these standard curves by default.

Table 1. Mersenne-like prime fields from various standards

Curve name	p	
wtls8	$2^{112} - 537$	(6)
secp128r1/r2	$2^{128} - 2^{97} - 1$	(7)
secp160r1	$2^{160} - 2^{31} - 1$	(8)
secp160k1/r2 / wtls7	$2^{160} - 2^{32} - 2^{14} - 2^{12} - 2^{9} - 2^{8} - 2^{7} - 2^{3} - 2^{2} - 1$	(9)
wtls9	$2^{160} - 229233$	(10)
secp192r1 / NIST P-192 / X9.62 p192v1/v2/v3	$2^{192} - 2^{64} - 1$	(11)
secp192k1	$2^{192} - 2^{32} - 2^{12} - 2^{8} - 2^{7} - 2^{6} - 2^{3} - 1$	(12)
X9.62 p239v1/v2/v3	$2^{239} - 2^{143} - 2^{95} + 2^{47} - 1$	(13)
secp256r1 / NIST P-256 / X9.62 p256v1	$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$	(14)
secp256k1	$2^{160} - 2^{32} - 2^{9} - 2^{8} - 2^{7} - 2^{6} - 2^{4} - 1$	(15)
secp384r1 / NIST P-384	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$	(16)
secp521r1 / NIST P-521	$2^{521} - 1$	(17)

3 Addition Chains and Exponentiation

Clearly for computing square roots modulo $p \equiv 3 \pmod 4$, a fixed exponent, variable base exponentiation (3) is used. A survey of some common methods for general fast exponentiation can be found in [11]. Fast exponentiation is normally carried out using an *addition chain* ([12] is the de facto reference on the topic; also see [13]). Addition sequences are also of interest for exponentiation. We use the definition and notation from [14].

Definition 1. *An addition chain for an integer a_l is a sequence γ of l pairs $((j(1), k(1)), \ldots, (j(l), k(l)))$ of integers with $0 \leq k(i) \leq j(i) < i$ for all $1 \leq i \leq l$, and the semantics $\mathcal{S}(\gamma) = \{a_0, \ldots, a_l\}$ is the set of integers such that $a_0 = 1$ and $a_i = a_{j(i)} + a_{k(i)}$ for $1 \leq i \leq l$. An addition chain can be viewed as a directed graph consisting of nodes $\mathcal{S}(\gamma)$ and edges from $a_{j(i)}$ and $a_{k(i)}$ to a_i.*

Definition 2. *An addition sequence for a set of integers $\mathcal{W}$ is an addition chain γ such that $\mathcal{W} \subset \mathcal{S}(\gamma)$.*

The binary expansion of an integer clearly yields an addition chain (we will refer to this as the *binary* addition chain), although the length l is not necessarily optimal; an *optimal* addition chain is one for which l is minimal.

Example 1. The smallest positive integer for which the binary addition chain is not optimal is 15. The binary addition chain is of length 6:

$$\begin{aligned} \gamma &= ((0,0),(1,0),(2,2),(3,0),(4,4),(5,0)) \\ \mathcal{S}(\gamma) &= \{1,2,3,6,7,14,15\} \end{aligned}$$

$$\begin{tikzcd} 1 \arrow[r, Rightarrow] \arrow[rr, bend right] \arrow[rrrr, bend right] \arrow[rrrrrr, bend right] & 2 \arrow[r] & 3 \arrow[r, Rightarrow] & 6 \arrow[r] & 7 \arrow[r, Rightarrow] & 14 \arrow[r] & 15 \end{tikzcd}$$

and $a^{15} = ((a^2a)^2a)^2a$, 3 squarings and 3 multiplications. But the optimal addition chain is of length 5:

$$\begin{aligned} \gamma &= ((0,0),(1,0),(2,2),(3,3),(4,2)) \\ \mathcal{S}(\gamma) &= \{1,2,3,6,12,15\} \end{aligned}$$

$$\begin{tikzcd} 1 \arrow[r, Rightarrow] \arrow[rr, bend right] & 2 \arrow[r] & 3 \arrow[r, Rightarrow] \arrow[rrr, bend left] & 6 \arrow[r, Rightarrow] & 12 \arrow[r] & 15 \end{tikzcd}$$

and $a^{15} = ((a^2a)^2)^2a^2a$, 3 squarings and 2 multiplications (using the previously computed value a^2a in the last step).

3.1 Square-and-Multiply Exponentiation

The *square-and-multiply (binary) method* for fast exponentiation is shown in Algorithm 1. The number of multiplications needed is one less than the Hamming weight of the exponent e denoted $\omega(e)$ and the number of squarings one less than the binary length $\ell = \lfloor \log_2 e \rfloor + 1$ of e; we denote this cost as $(\omega(e)-1)\mathsf{M}+(\ell-1)\mathsf{S}$, where M and S denote field multiplications and field squarings, respectively. Algorithm 1 scans the bits of e in a left-to-right fashion; bit i of e is denoted as e_i. The bits of e can also be scanned from right-to-left, but one additional accumulator is needed. This simple method is in fact the only fast exponentiation method included in some popular cryptography standards ([2] for example).

Algorithm 1. Square-and-multiply exponentiation (binary method)

```
Input: Modulus $p$, exponent $e$ of length $\ell$, $\beta \in \mathbb{F}_p$
Output: $\beta^e \mod p$
$\eta \leftarrow 1$
for $i \leftarrow \ell - 1$ to $0$ do
    $\eta \leftarrow \eta^2 \mod p$                              // 1 squaring
    if $e_i = 1$ then $\eta \leftarrow \eta\beta \mod p$          // 1 multiplication
end
return $\eta$
```

Algorithm 2. Sliding window exponentiation

```
Input: Modulus $p$, exponent $e$ of length $\ell$, $\beta \in \mathbb{F}_p$, window size $w$
Output: $\beta^e \mod p$
Precompute $\beta^i \mod p$ for all odd $i < 2^w$.
$\eta \leftarrow 1$, $i \leftarrow \ell - 1$
while $i \geq 0$ do
    if $e_i = 0$ then $t \leftarrow 1$, $u \leftarrow 0$
    else Find the largest $t \leq w$ such that $u \leftarrow (e_i, \ldots, e_{i-t+1})$ is odd.
    $\eta \leftarrow \eta^{2^t} \mod p$                                  // $t$ squarings
    if $u > 0$ then $\eta \leftarrow \eta\beta^u \mod p$     // 1 multiplication from precomputed
                                                                          values
    $i \leftarrow i - t$
end
return $\eta$
```

Table 2. OpenSSL sliding window widths

Bit-length ℓ of e	Window size w
$\ell > 671$	6
$671 \geq \ell > 239$	5
$239 \geq \ell > 79$	4
$79 \geq \ell > 23$	3
$23 \geq \ell$	2

3.2 Sliding Window Exponentiation

Instead of scanning the bits of e one at a time, it is possible to scan w-bits at a time, or scan e in w-bit *windows*. This can significantly reduce the number of multiplications needed when coupled with some precomputation (offline for fixed base exponentiation, online for variable base exponentiation). The *sliding window method* takes this approach, with placement of the width w window such that value in the window is odd (zeros are "skipped"). Up to w squarings are then applied in each iteration, and then one multiplication is used to put the window in place. The strategy for precomputation and choice of window width w vary, depending on the intended application. Algorithm 2 outlines the general case, where all of the possible values of an odd width w windows are precomputed. An alternate strategy is to use a larger window width and only precompute needed values; some good heuristics are given in [15]. Note that Algorithm 1 is the specific case of Algorithm 2 with $w = 1$.

In general, the appropriate window width choice is clearly related to the binary length of the exponent, which can vary depending on the cryptosystem. For example, using RSA [16] it can be common to choose a small value for the public key (e.g. `0x10001`, short and low-weight)—this makes verifying digital signatures faster. It would then be better to not use any sliding window at all when verifying signatures.

Many wide-spread publicly available cryptographic software libraries implement something similar to Algorithm 2 for exponentiation, including OpenSSL. The different window width sizes that OpenSSL chooses for different exponent lengths are listed in Table 2. The amount of precomputation is $2^{w-1} - 1$.

3.3 Analysis, Square Root Costs

In practical implementations, the general disadvantage of using more efficient addition chains is that a large number of intermediate values may have to be stored. However, clearly for fixed exponent, variable base exponentiation, any computation of an efficient yet practical addition chain can be done offline.

Unfortunately, finding an optimal addition chain is NP-hard (see [15] for a discussion). In practice, efficient but not necessarily optimal addition chains are used. The length l of an optimal addition chain is bounded by

$$\log e + \log \omega(e) - 2.13 \leq l \leq \lfloor \log e \rfloor + \omega(e) - 1. \tag{18}$$

Note that the upper bound is the binary method.

Now that the theoretical bounds have been established, the fixed cost for square root computation for the fields in Table 1 using the binary method are listed in Table 3, where $e = (p + 1)/4$ is the exponent. Note that the binary method for P-521 (or any true Mersenne prime) is already optimal, and hence we omit any further analysis.

Remark 1. For the basic comparison of methods in this paper, we have chosen to compare versus the square-and-multiply method. The reasoning behind this choice is that it is the basic, standard method of performing an exponentiation—and as previously mentioned, the only method actually listed in some standards [2,3]. We note that in some cases for Mersenne-like primes it would be possible to improve performance by using a signed representation, trading a series of field multiplications for a single field inversion (which would only need to be computed and stored online once per exponentiation). Due to the high inversion to multiplication ratio (I/M) in $\mathbb{F}_p$, this approach would outperform the binary method in some cases, but not the sliding window method when using higher window widths as I/M is so high; the particularly efficient finite field arithmetic presented in [17] lists I/M=80 for NIST curves over $\mathbb{F}_p$.

3.4 Repunit Exponentiation

A number of the form $2^n - 1 = \sum_{i=0}^{n-1} 2^i$, all ones (weight n) in the expansion, is in general known as a *repunit* [18,14]. An optimal chain for a repunit is simple to find—the following equation is well-known and has indeed been used in cryptography before. Using the notation from [14], denoting $w_j = 2^j - 1$ it follows [13] that

$$w_{a+b} = \sum_{0 \leq i < (a+b)} 2^i = \left(\sum_{0 \leq i < a} 2^i \right) 2^b + \sum_{0 \leq i < b} 2^i = w_a 2^b + w_b \tag{19}$$

Table 3. Square root costs, binary method

p^*	e	$\omega(e)$	Cost	Bound on $\mathsf{M}+\mathsf{S}$
(4)	$\sum_{i=b-2}^{c-2} 2^i + \cdots$	$\omega(p)-(a+1)$	$(\lfloor\log_2 e\rfloor)\mathsf{M}+(\omega(e)-1)\mathsf{S}$	—
(5)	$\sum_{i=j-2}^{d-3} 2^i$	$d-j$	$(d-j-1)\mathsf{M}+(d-3)\mathsf{S}$	—
(6)	$\sum_{i=8}^{109} 2^i + \sum_{i=3}^{6} 2^i + 2$	107	$106\mathsf{M}+109\mathsf{S}$	113
(7)	$\sum_{i=95}^{125} 2^i$	31	$30\mathsf{M}+125\mathsf{S}$	127
(8)	$\sum_{i=29}^{157} 2^i$	129	$128\mathsf{M}+157\mathsf{S}$	162
(9)	$\sum_{i=31}^{157} 2^i + \sum_{i=13}^{29} 2^i + 2^{11} + \sum_{i=8}^{9} 2^i + \sum_{i=2}^{4} 2^i + 1$	151	$150\mathsf{M}+157\mathsf{S}$	162
(10)	$\sum_{i=16}^{157} 2^i + 2^{13} + 2^5 + 2^2$	145	$144\mathsf{M}+157\mathsf{S}$	162
(11)	$\sum_{i=62}^{189} 2^i$	128	$127\mathsf{M}+189\mathsf{S}$	194
(12)	$\sum_{i=31}^{189} 2^i + \sum_{i=11}^{29} 2^i + \sum_{i=7}^{9} 2^i + \sum_{i=1}^{3} 2^i$	184	$183\mathsf{M}+189\mathsf{S}$	194
(13)	$\sum_{i=142}^{236} 2^i + \sum_{i=93}^{140} 2^i + 2^{45}$	144	$143\mathsf{M}+236\mathsf{S}$	241
(14)	$\sum_{i=222}^{253} 2^i + 2^{190} + 2^{94}$	34	$33\mathsf{M}+253\mathsf{S}$	256
(15)	$\sum_{i=31}^{253} 2^i + \sum_{i=8}^{29} 2^i + \sum_{i=2}^{3} 2^i$	247	$246\mathsf{M}+253\mathsf{S}$	259
(16)	$\sum_{i=127}^{381} 2^i + \sum_{i=94}^{125} 2^i + 2^{30}$	288	$287\mathsf{M}+381\mathsf{S}$	387
(17)	2^{519}	1	$0\mathsf{M}+519\mathsf{S}$	519

* See Table 1 for the full form.

and an addition chain for w_{a+b} can be constructed using b squarings and two smaller addition chains for w_a and w_b. The problem is now simplified to finding an addition chain for n, and indeed in most practical applications a brute force search for an optimal addition chain for n can easily be carried out.

Example 2. Consider the previous binary method addition chain for 15; an addition chain for $w_{15} = 2^{15} - 1$ is given by

$$\begin{tikzcd}
w_1 \arrow[r, Rightarrow, "\cdot 2"] \arrow[rr, bend right] \arrow[rrrr, bend right] \arrow[rrrrrr, bend right] & w_2 \arrow[r, "\cdot 2"] & w_3 \arrow[r, Rightarrow, "\cdot 2^3"] & w_6 \arrow[r, "\cdot 2"] & w_7 \arrow[r, Rightarrow, "\cdot 2^7"] & w_{14} \arrow[r, "\cdot 2"] & w_{15}
\end{tikzcd}$$

where $\cdot 2^b$ are the b intermediate squarings; $a^{2^{15}-1}$ is computed using 6 multiplications and 14 squarings. The optimal chain of length 5 could be used to reduce the number of multiplications to 5.

Algorithm 3 is a straightforward implementation of (19) which uses the binary addition chain for n; this choice is often not optimal with regards to the number of field multiplications, but has a larger scope of applications. As written, Algorithm 3 is executed at a cost of

$$n+\log_2 n+\omega(n)-2, \text{ or in field operations } (n-1)\mathsf{S}+(\log_2 n+\omega(n)-1)\mathsf{M} \quad (20)$$

Algorithm 3. Repunit exponentiation

```
Input: Field element $\beta \in \mathbb{F}_p$, modulus $p$, exponent $n$.
Output: $\beta^{2^n-1} \mod p$.
Let $b_j$ be the MSB of $n$.
$\eta \leftarrow \beta$ , $k \leftarrow 1$
for $i \leftarrow j-1$ to 0 do
    $\mu \leftarrow \eta^{2^k} \mod p$                          // $k$ squarings
    $\eta \leftarrow \mu\eta \mod p$ , $k \leftarrow 2k$                // 1 multiplication
    if $b_i = 1$ then $\eta \leftarrow \eta^2\beta \mod p$ , $k \leftarrow k+1$  // binary method, $b_i$ are the
                                                                  bits of $n$
end
return $\eta$
```

Instead of using the binary chain for n, it is easy to substitute a shorter chain. Some efficient implementations of cryptographic operations make use of (19). For example, Itoh and Tsujii [19] used the idea for inversion in binary fields.

4 Fast Point Decompression for Standard Curves

For the square root computation needed for point decompression, we take a similar approach to sliding window exponentiation as in Algorithm 2 coupled with precomputation of repunits using Algorithm 3. As Table 3 illustrates, we can conceptually view e as consisting of k *repunit windows* of the form $2^{n_i} - 1$ (that is,"splitting" e on zeros). Let the distinct n_i appearing in e make up the set $\mathcal{W}$. Let $2^{n_s} - 1$ be the window in the most significant bits of e and $2^{n_b} - 1$ the largest window. To describe the strategy, we present the following theorem.

Theorem 1. *Let $p \equiv 3 \pmod 4$ be a prime and $\beta \in \mathbb{F}_p$ be a quadratic residue modulo p. Let γ of length l be the shortest addition sequence for $\mathcal{W}$. The cost of computing the square root of β in $\mathbb{F}_p$ is at most $\lfloor \log_2 e \rfloor - n_s + n_b$ squarings and $k - 1 + l$ multiplications in $\mathbb{F}_p$.*

Proof. As $\beta \in QR_p$, β^e modulo p clearly yields the square root. For this exponentiation, we use Algorithm 2 as a base. In the precomputation phase, we use γ combined with one execution of Algorithm 3 to compute all the values of $\beta^{2^{n_i}-1} \mod p$. The length of γ is l so this costs l multiplications and $n_b - 1$ squarings. We start the accumulator with a value of $\beta^{2^{n_s}-1} \mod p$. From the bit position to the right of $2^{n_s} - 1$, we proceed to the least significant bit of e, performing squarings and using multiplications to put the remaining $k - 1$ windows into place. The number of squarings is thus $\lfloor \log_2 e \rfloor - n_s + 1$ and the number of multiplications is $k - 1$. This yields a maximum of $\lfloor \log_2 e \rfloor - n_s + n_b$ squarings and $k - 1 + l$ multiplications in $\mathbb{F}_p$. □

The strategy therefore is to find the shortest addition sequence γ for $\mathcal{W}$, easily done through exhaustive search for seemingly all practical purposes (l is small).

Algorithm 4. Fast square roots, sliding window exponentiation with repunit precomputation

```
Input: Modulus $p \equiv 3 \pmod 4$, $\beta \in QR_p$, addition sequence $\gamma$
Output: Square root of $\beta$ in $\mathbb{F}_p$
// Using notation from Thm. 1
Using one execution of Algorithm 3 combined with $\gamma$, precompute $\beta^{2^{n_i}-1} \bmod p$ for all $n_i \in \mathcal{W}$.
$\eta \leftarrow \beta^{2^{n_s}-1} \bmod p$, $i \leftarrow \lfloor \log_2 e \rfloor - n_s$
while $i \geq 0$ do
    if $e_i = 0$ then $t \leftarrow 1$
    else Find the largest $t$ such that $(e_i, \ldots, e_{i-t+1})$ is a repunit window.  // no length restriction on $t$
    $\eta \leftarrow \eta^{2^t} \bmod p$  // $t$ squarings
    if $e_i \neq 0$ then $\eta \leftarrow \eta\beta^{2^t-1} \bmod p$  // 1 multiplication from precomputed values
    $i \leftarrow i - t$
end
return $\eta$
```

The binary chain in Algorithm 3 is then replaced with γ (this is easily done, see (19)) and only *one* execution is used to precompute all of the elements $\beta^{2^{n_i}-1}$ mod p, and the final exponentiation is done similarly as in Algorithm 2. This is illustrated in Algorithm 4.

4.1 Analysis and Application

Clearly for an arbitrary p, this strategy is not particularly efficient and in the worst case roughly regresses to the square-and-multiply method as the number of windows k approaches the weight $\omega(e)$. However, Given p of Mersenne-like form—in particular (4), (5) and Table 1—we can identify where this method will be generally advantageous:

– If k is very small (as it frequently is with Mersenne-like p), the number of multiplications needed to put these windows into place is significantly reduced.
– If $n_s = n_b$ (the largest window appears in the most significant bits of e), significantly fewer squarings are needed.

We now provide a few examples to illustrate how Algorithm 4 can be used efficiently.

Example 3. Consider (5): $k = 1$ and γ is the shortest addition sequence for $\mathcal{W} = n_s = d - j$ found easily through exhaustive search. We calculate $\beta^{2^{d-j}-1}$ mod p using Algorithm 3 at a cost of $(d - j - 1)\mathsf{S} + l\mathsf{M}$, where l is at most $\lfloor \log_2(d - j) \rfloor + \omega(d - j) - 1$ when using the binary chain (20). We then put this window into place at a cost of $k-1=0$ multiplications and $j-2$ squarings. This is the form of P-192; the cost of computing $\beta^{2^{128}-1} \bmod p$ is $7\mathsf{M}+127\mathsf{S}$.

$$\begin{tikzcd}
w_1 \arrow[r, Rightarrow, "\cdot 2"] & w_2 \arrow[r, Rightarrow, "\cdot 2^2"] & w_4 \arrow[r, Rightarrow, "\cdot 2^4"] & w_8 \arrow[r, Rightarrow, "\cdot 2^8"] & w_{16} \arrow[r, Rightarrow, "\cdot 2^{16}"] & w_{32} \arrow[r, Rightarrow, "\cdot 2^{32}"] & w_{64} \arrow[r, Rightarrow, "\cdot 2^{64}"] & w_{128}
\end{tikzcd}$$

An additional $62\mathsf{S}$ is then needed once this value is obtained. As the only window length is a power of 2 ($\mathcal{W}=\{128\}$), the binary chain is clearly optimal and in fact far less logic is needed in Algorithm 3 (the inner if condition can be omitted) and Algorithm 4.

Example 4. Consider P-384, with $k=3$ windows and $\mathcal{W}=\{1,32,255\}$. To calculate γ, we do exhaustive search for the shortest addition sequence γ for $\mathcal{W}$. The binary addition chain for 255 is of length 14 (but does not contain 32), an optimal addition chain for 255 is of length 10, but the shortest addition sequence for $\mathcal{W}$ is of length 11 (the value $\beta^{2^{32}-1} \bmod p$ can be computed intermediately at the cost of only one additional multiplication when compared to an optimal addition chain).

$$\begin{tikzcd}
w_1 \arrow[r, Rightarrow, "\cdot 2"] \arrow[rrrrrrrrr, bend right] & w_2 \arrow[r, Rightarrow, "\cdot 2^2"] & w_4 \arrow[r, Rightarrow, "\cdot 2^4"] \arrow[rrrrrr, bend left] & w_8 \arrow[r, Rightarrow, "\cdot 2^8"] & w_{16} \arrow[r, Rightarrow, "\cdot 2^{16}"] \arrow[rrr, bend left] & w_{32} \arrow[r, Rightarrow, "\cdot 2^{32}"] & w_{64} \arrow[r, "\cdot 2^{16}"] & w_{80} \arrow[r, "\cdot 2^{4}"] & w_{84} \arrow[r, "\cdot 2"] & w_{85} \arrow[r, Rightarrow, "\cdot 2^{85}"] \arrow[rr, bend left] & w_{170} \arrow[r, "\cdot 2^{85}"] & w_{255}
\end{tikzcd}$$

The precomputation cost is thus $254\mathsf{S}+11\mathsf{M}$; $k-1=2\mathsf{M}$ is needed to put the smaller windows into place and an additional $127\mathsf{S}$.

4.2 Analysis, New Square Root Costs

Table 4 provides the cost of square root computation for the applicable fields from Table 3 when using Algorithm 4. The "Savings" column is the saving achieved when compared to the square-and-multiply (binary) method (Algorithm 1). In savings calculation, the reasonable [17] estimate of $\mathsf{S}=0.85\mathsf{M}$ is used. Given the bounds of (18), in all cases for the standard curves the costs are optimal or near-optimal.

Remark 2. Although the savings listed in Table 4 seem extremely significant, we note that point decompression is not normally the time consuming operation in elliptic curve cryptography—scalar multiplication is. Therefore if the benchmark is on an entire cryptographic operation (for example, point decompression and then signature verification) the speedup obtained in practice from the use of such fast point decompression will not be as significant. However, this is all dependent on the intended application.

5 Implementation

Implementations of point decompression were done to investigate the feasibility of reaching the theoretical speedups listed in Table 4. Both software and hardware implementations are provided for comparison. The results are presented below.

Table 4. Square root costs, Algorithm 4

p	$\mathcal{W}$	$\mathcal{S}(\gamma)$	Cost	**Savings** (%)
(6)	$\{1, 4, 102\}$	$\{1, 2, 4, 8, 16, 17, 34, 68, 102\}$	$10\mathsf{M} + 109\mathsf{S}$	48.3
(7)	$\{31\}$	$\{1, 2, 3, 6, 12, 15, 30, 31\}$	$7\mathsf{M} + 125\mathsf{S}$	15.0
(8)	$\{129\}$	$\{1, 2, 4, 8, 16, 32, 64, 128, 129\}$	$8\mathsf{M} + 157\mathsf{S}$	45.9
(9)	$\{1, 2, 3, 17, 127\}$	$\{1, 2, 3, 6, 7, 14, 17, 31, 62, 124, 127\}$	$15\mathsf{M} + 157\mathsf{S}$	48.2
(10)	$\{1, 142\}$	$\{1, 2, 4, 8, 16, 17, 34, 35, 70, 71, 142\}$	$13\mathsf{M} + 157\mathsf{S}$	47.2
(11)	$\{128\}$	$\{1, 2, 4, 8, 16, 32, 64, 128\}$	$7\mathsf{M} + 189\mathsf{S}$	41.7
(12)	$\{3, 19, 159\}$	$\{1, 2, 3, 4, 8, 16, 19, 35, 70, 140, 159\}$	$13\mathsf{M} + 189\mathsf{S}$	49.0
(13)	$\{1, 48, 95\}$	$\{1, 2, 4, 8, 9, 18, 19, 38, 47, 48, 95\}$	$12\mathsf{M} + 236\mathsf{S}$	38.1
(14)	$\{1, 32\}$	$\{1, 2, 4, 8, 16, 32\}$	$7\mathsf{M} + 253\mathsf{S}$	10.5
(15)	$\{2, 22, 223\}$	$\{1, 2, 3, 4, 8, 11, 22, 44, 88, 110, 220, 223\}$	$13\mathsf{M} + 253\mathsf{S}$	51.8
(16)	$\{1, 32, 255\}$	$\{1, 2, 4, 8, 16, 32, 64, 80, 84, 85, 170, 255\}$	$13\mathsf{M} + 381\mathsf{S}$	44.9

5.1 Software

OpenSSL was used as a basis for the software implementation. OpenSSL performs point decompression via the function `ec_GFp_simple_set_compressed_coordinates`, which calculates the righthand side of (1) and in turn calls the function `BN_mod_sqrt` to compute the square root, which in this case calls the function `BN_mod_exp` which uses a sliding window (Algorithm 2). For curve operations such as point addition, OpenSSL defines specific field addition and squaring functions, allowing for fast reduction for the NIST primes listed in Table 1. In the case of a non-NIST prime, the fallback method is Montgomery multiplication [20]. However, the `BN_mod_sqrt` function is not provided by the elliptic curve portion of OpenSSL and thus it always uses Montgomery multiplication (no fast reduction).

For this implementation, we provide the wrapper function `ec_GFp_field_sqrt` inside the ECC portion of OpenSSL. This allows fast reduction to be used when available, and Montgomery multiplication otherwise. Three timings are given: one using the unmodified OpenSSL, one using the binary method (Algorithm 1), and one using the new method in Algorithm 4. The resulting timings for the binary method are sometimes better than those of the sliding window method and can be explained by the fact that `BN_mod_exp` is located outside of the ECC portion of OpenSSL and hence does not have access to fast reduction.

The implementation was done on an AMD Athlon Thunderbird 1.0GHz 32-bit processor with 1GB of RAM running Debian Linux. The compiler used was GCC v4.1.2 using compiling switches -march=athlon-tbird -O3 -pipe -fforce-addr -fomit-frame-pointer and OpenSSL v0.9.8g. The results are given in Table 5. The "Savings" column indicates the achieved savings of Algorithm 4 over the unmodified OpenSSL and Algorithm 1, respectively.

These practical results are inline with the theoretical speedups listed in Table 4, with any variation due to different squaring to multiplication ratios and some limited extra logic needed to implement Algorithm 4. These are actual

Table 5. Software timings, OpenSSL point decompression (μs)

p	OpenSSL (Alg. 2)	Binary (Alg. 1)	Alg. 4	**Savings** (%)	
(6)	159.9	214.3	123.1	23.0	42.6
(7)	159.3	152.6	129.8	18.5	14.9
(8)	278.1	365.3	224.4	19.3	38.6
(9)	300.7	393.9	234.4	22.0	40.5
(10)	288.8	385.2	231.3	19.9	40.0
(11)	358.6	299.6	190.8	46.8	36.3
(12)	386.7	547.0	305.4	21.0	44.2
(13)	507.9	616.6	412.8	18.7	33.1
(14)	516.6	310.8	280.5	45.7	9.7
(15)	597.5	804.3	431.4	27.8	46.4
(16)	1574.7	1386.3	789.9	49.8	43.0

timings of the OpenSSL point decompression function (unmodified and modified), so these are the results that will be seen in practice.

5.2 Hardware

Field specific processors were implemented in an Altera Stratix II EP2S180C3 FPGA [21] in order to study feasibility of the fast square root method in hardware. The primes included in this study are (7), (8), (9), (10), (11), and (16), as defined in Table 1. The processors are straightforward implementations of commonly-known algorithms and, hence, we omit a detailed description. However, a short presentation is given in the following.

Processors compute field multiplications with integer multiplications whose results are then reduced modulo p by using fast reduction methods. Integer multipliers were implemented using DSP blocks embedded into Stratix II FPGAs. Each DSP block can implement one up to 36x36-bit multiplication [21]. Reduction circuitries were implemented with reconfigurable logic and primes can be switched by reconfiguring the FPGA. Two different fast reduction methods were used. If the number of 2^i terms was small, we used the method developed for Mersenne-like primes in [22] which computes reductions with shifts and additions. Such equations are given for all NIST primes in [1] ((11) and (16)), [23] includes formulae for (7), and equations for (8) were derived by hand. Another type of primes that we considered have a relatively large number of terms with small powers, i.e. the primes have the form $2^d - c$ where c is relatively small. The primes defined by (9) and (10) have such forms. For them we used the method presented by Crandall [24] which computes reductions with additions and multiplications by c. The processors include memory for storing field elements and it was implemented using embedded memory blocks, called M4K memory blocks [21]. Table 6 summaries design decisions and methods that were used for different primes. Latencies for multiplication and squaring are also provided, as well as their ratio.

When the processors were described in VHDL and synthesized with Altera Quartus II 6.0 SP1 design software, resource requirements and maximum clock

Table 6. Summary of implementation methods

p	Multiplier size	Reduction method	Memory size (elements)	M	S	S/M
(7)	64x64-bit	Mersenne-like [23]	256x128-bit (256)	9	8	0.89
(8)	80x80-bit	Mersenne-like	256x160-bit (256)	9	8	0.89
(9)	80x80-bit	Crandall	256x160-bit (256)	9	8	0.89
(10)	80x80-bit	Crandall	256x160-bit (256)	10	9	0.90
(11)	96x96-bit	NIST [1]	256x192-bit (256)	9	8	0.89
(16)	96x96-bit	NIST [1]	256x192-bit (128)	29	23	0.79

Table 7. Implementation results on Altera Stratix II EP2S180C3 FPGA

p	ALUTs	Regs.	ALMs	M4Ks	DSPs	Max. clock
(7)	2,539	951	1,428	8	4	80.18 MHz
(8)	2,250	1,174	1,289	9	9	83.04 MHz
(9)	4,141	1,213	2,206	9	9	45.29 MHz
(10)	3,206	1,356	1,733	9	9	66.56 MHz
(11)	2,855	1,397	1,609	11	9	73.82 MHz
(16)	8,001	2,982	4,551	11	9	39.55 MHz

Table 8. Hardware timings, square root computation

	Binary (Alg. 1)		Alg. 4			
p	Latency	Time (μs)	Latency	Time (μs)	Speedup	**Savings** (%)
(7)	1,426	17.78	1,205	15.03	8.63	15.4
(8)	2,408	29.00	1,328	15.99	14.03	44.9
(9)	2,913	64.32	1,562	34.49	6.80	46.4
(10)	3,154	47.39	1,713	25.74	8.98	45.7
(11)	2,971	40.25	1,771	23.99	7.95	40.4
(16)	17,755	448.93	9,535	241.09	3.28	46.3

frequencies were as shown in Table 7. Costs of square root computations with Algorithm 1 (binary) and Algorithm 4 (the new method) are presented in Table 8. The "Speedup" column shows the speedup factors over the software timings from Table 5. The savings are clearly close to the ones presented in Table 4 in hardware as well.

As can be seen from Tables 7 and 8, the Mersenne-like primes give better results because reductions are faster and they require less area. Especially, (11) results in a very efficient implementation although the field size is relatively large. Crandall reduction is slower and requires more area, as expected. Nonetheless, Algorithm 4 gives major enhancements compared to Algorithm 1 for all implemented primes and the results are inline with the theoretical calculations given in Table 4. FPGAs also provide major speedups ranging from 3.28 to 14.03 times over software-based implementations.

6 Conclusion

This paper has explored the problem of computing fast point decompressions for curves defined over the finite field $\mathbb{F}_p$ with $p \equiv 3 \pmod 4$ of Mersenne-like form. A significantly more efficient method has been presented, based on a known equation from number theory which yields an estimated theoretical speed savings of up to 51.8% at a cost of little to no storage. In addition to a general analysis of Mersenne-like p, an extensive analysis of the prime fields for standard curves was given, so these results have a large scope of practical application as well. Both software and hardware implementation results have been provided, which not only are inline with the theoretical savings, but also show that there is significant speedup in hardware over software.

References

1. FIPS: Digital signature standard (DSS). FIPS PUB 186-2 (+ Change Notice). Technical report, U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology (2000)
2. IEEE: Standard specifications for public-key cryptography. Technical Report IEEE P1363 / D13, Institute of Electrical and Electronics Engineers, Inc. (1999)
3. ANSI: The elliptic curve digital signature algorithm. American National Standards Institute, ANSI X9.62-1998 (1998)
4. SECG: Standards for efficient cryptography. Standards for Efficient Cryptography Group, Version 1.0 (2000)
5. NIST: Recommended elliptic curves for federal government use. Technical report, National Institute of Standards and Technology (NIST) (1999)
6. WTLS: Wireless application protocol, wireless transport layer security specification. Wireless Application Forum (1999)
7. BSIG: Simple pairing whitepaper. Technical report, Bluetooth Special Interest Group (2006), `http://www.bluetooth.com/Bluetooth/Apply/Technology/Research/.Simple_Pairing.htm`
8. Cox, M., Engelschall, R., Henson, S., Laurie, B.: The OpenSSL Project. v0.9.8g (2007), `http://www.openssl.org/`
9. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Information Theory IT-22(6), 644–654 (1976)
10. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
11. Gordon, D.M.: A survey of fast exponentiation methods. J. Algorithms 27(1), 129–146 (1998)
12. Knuth, D.E.: Seminumerical Algorithms. 3rd edn. The Art of Computer Programming, vol. 2. Addison-Wesley, Reading (1998)
13. Brauer, A.: On addition chains. Bulletin of the American Mathematical Society 45, 736–739 (1939)
14. von zur Gathen, J., Nöcker, M.: Computing special powers in finite fields. Math. Comp. 73(247), 1499–1523 (2004) (electronic)
15. Bos, J.N., Coster, M.J.: Addition chain heuristics. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 400–407. Springer, Heidelberg (1990)
16. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM 21(2), 120–126 (1978)

17. Brown, M., Hankerson, D., López, J., Menezes, A.: Software implementation of the NIST elliptic curves over prime fields. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 250–265. Springer, Heidelberg (2001)
18. Beiler, A.H.: Recreations in the Theory of Numbers. Dover, NY (1964)
19. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. Inform. and Comput. 78(3), 171–177 (1988)
20. Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. 44(170), 519–521 (1985)
21. Altera: Stratix II device handbook, vol. 1–2, ver. 4.1 (2006)
22. Solinas, J.A.: Generalized Mersenne numbers. Technical report CORR 99-39, Centre for Applied Cryptographic Research, University of Waterloo (1999)
23. Guajardo, J., Blümel, R., Krieger, U., Paar, C.: Efficient implementation of elliptic curve cryptosystems on the TI MSP430x33x family of microcontrollers. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 365–382. Springer, Heidelberg (2001)
24. Crandall, R.: Method and apparatus for public key exchange in a cryptographic system. United States Patent 5,159,632 (1992)

An Efficient Strong Key-Insulated Signature Scheme and Its Application

Go Ohtake[1], Goichiro Hanaoka[2], and Kazuto Ogawa[1]

[1] Science & Technical Research Laboratories,
Japan Broadcasting Corporation
1-10-11 Kinuta, Setagaya-ku, Tokyo 157-8510, Japan
{ohtake.g-fw,ogawa.k-cm}@nhk.or.jp
[2] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology
1102 Akihabara Daibiru, 1-18-13 Sotokanda, Chiyoda-ku, Tokyo 101-0021, Japan
hanaoka-goichiro@aist.go.jp

Abstract. The security of a system is often compromised by exposure of secret keys even if its underlying cryptographic tools are perfectly secure, assuming that their secret keys will be never exposed to adversaries. A key-insulated signature scheme is a useful cryptographic primitive for reducing the damage caused by such leakage. In this paper, we propose an *efficient* strong key-insulated signature (KIS) scheme and prove its security. This scheme is significantly more efficient than conventional strong KIS schemes especially in terms of signature size, and it is provably secure under the discrete logarithm (DL) assumption in the random oracle model. It is constructed by extending the Abe-Okamoto signature scheme [1]; we give a formal proof of adaptive key-exposure security as it is not addressed in [1]. A typical application of our scheme is to an authentication system in which one (or a small number of) sender communicates with many receivers since multiple copies of the sender's signature are transmitted to individual receivers in such a system. We discuss a bidirectional broadcasting service as an example.

Keywords: strong key-insulated signature, key leakage, DL assumption, random oracle model, adaptive security.

1 Introduction

1.1 Background

Digital signature is now an important technology, but signing key leakage has become a critical problem. If a signing key is leaked to an adversary, she can easily impersonate the signer. Recently, several signature schemes have been proposed as countermeasures against key leakage, and the key-insulated signature (KIS) scheme [4] is a useful cryptographic primitive for reducing the damage caused by such leakage.

A typical application of the KIS scheme is a large-scale multi-receiver authentication system in which a signer communicates with a huge number of receivers.

In such a system, to renew his verification key, the signer has to send his new verification key to *all* receivers in an *authentic* manner. Therefore, we must deal with the key exposure problem without renewing a verification key. We noticed that a KIS scheme is an appropriate tool for this requirement. A bidirectional broadcasting service [11] is one such application.

The efficiency of an authentication system for a bidirectional broadcasting service heavily depends on the signature size since multiple copies (proportional to the total number of receivers) are individually transmitted to receivers. For example, when using a trivial KIS with simple double signing [4] in the Schnorr signature scheme [13], the number of signatures over the whole network becomes $1120 \cdot n$ bits, where n is the total number of users.

Strong key-insulated security [4], rather than standard key-insulated security, is needed to deal with various ways of key exposure. The difference between strong KIS scheme and standard KIS scheme is discussed in Section 1.2.

Our main motivation is to design an efficient strong KIS scheme with a significantly shorter signature size.

1.2 Related Work

In [3], Dodis, Katz, Xu, and Yung advocated the notion of *key-insulated security.* Their goal was to minimize the damage caused by secret-key exposures, and they proposed *key-insulated signature (KIS) schemes.* A KIS scheme enables the signing key to be updated without updating the verification key. Key updating establishes a lifetime for a signing key, and this means that damage due to a leaked signing key cannot occur beyond its lifetime. A KIS scheme has not only forward security, such that no one can generate signing keys before key leakage, but also backward security, such that no one can generate signing keys after key leakage. KIS schemes have been actively studied as a countermeasure against key leakage [5,8,14,15].

A KIS scheme is secure if the master key is stored in a secure device, but it is not secure if the master key is leaked to an adversary. Therefore, a signer must securely manage the master key when using a KIS scheme. In [4], Dodis, Katz, Xu, and Yung proposed a construction method for a *strong key-insulated signature scheme.* A standard KIS scheme is secure against only signing key leakage, whereas a strong KIS scheme is secure against either signing key leakage or master key leakage. Therefore, a strong KIS scheme is more secure than a KIS scheme. The strong KIS scheme in [4] is provably secure under the assumption of the infeasibility of a discrete logarithm problem (DLP). An upper bound is set on the number of signing key leakages in this scheme, however, and the size of the verification key is very large. In [4], the authors also gave a method for constructing a strong KIS scheme based on the *Guillou-Quisquater signature scheme* [6]. This scheme is more efficient than the former scheme, but it is provably secure under the RSA assumption. (This scheme is described in Appendix B.)

Malkin, Obana, and Yung gave a generic conversion of a *proxy signature scheme* [10], in which a signer can delegate his signing right to a third party, to a KIS scheme, which is provably secure in the standard model [9]. However, the

KIS scheme obtained from this conversion is not a *strong* KIS scheme. In [4], Dodis, Katz, Xu, and Yung showed that a KIS scheme can be converted into a *strong* KIS scheme. Their method of constructing a strong KIS scheme, however, uses double signing. That is, the scheme requires two signatures: a signature with a master key and a signature with the signer's secret key. Therefore, the scheme is not efficient. (This scheme is described in Appendix A.) In other words, it is difficult to construct an *efficient* and *strong* KIS scheme, even if there is an efficient proxy signature scheme.

There are other approaches for reducing the damage caused by signing key leakage. Bellare and Miner proposed a *forward-secure signature scheme* [2]. This scheme has forward security but not backward security. Itkis and Reyzin proposed an *intrusion-resilient signature scheme* [7] in which both the master and signing keys are updated. This scheme has forward security even if both master and signing keys are leaked. In contrast, a KIS scheme is not secure if both keys are leaked.

1.3 Our Contributions

We propose an *efficient* strong KIS scheme and prove its security. Our scheme is more efficient than conventional strong KIS schemes, especially in terms of signature size, and is provably secure under the DL assumption in the random oracle model.

The signature size in our scheme is about two fifths that in previous schemes. More precisely, the signature sizes of the schemes described in Appendices A and B and our scheme are 1120, 1184, and 480 bits, respectively.

As pointed out in [9], a KIS scheme can be generically constructed from any proxy signature scheme, and therefore, to design an efficient KIS scheme, we start with the Abe-Okamoto proxy signature scheme [1] as the basic primitive. However, the conversion from the Abe-Okamoto scheme into the KIS scheme is not straightforward since the security definition of [1] does not consider an adaptive adversary who can get signing keys at any time and in any order. Therefore, we have to address such adaptive security with a formal security proof from scratch, and extend it to strong key-insulated security without increasing the signature size.

As an example application for our scheme, we discuss a bidirectional broadcasting service. By applying our scheme to a bidirectional content distribution system proposed in [11], we can reduce the damage caused by master key leakage, while enabling efficient signing and verification. Moreover, the network cost for transmitting signed messages can be reduced. Currently, there are more than 35,000,000 users of Japanese broadcasting services. The total amount of signature data would be 39.2 (= 1120 $\times$ 35,000,000) Gbits if all users used the scheme in Appendix A. In contrast, our scheme would only require 16.8 (= 480 $\times$ 35,000,000) Gbits of signature data for all users, a reduction of 22.4 Gbits.

2 Definitions

We address the model of the KIS scheme and its security definition.

2.1 Model

Definition 1. The KIS scheme consists of five polynomial time algorithms (Gen, Upd^*, Upd, Sign, Vrfy) as follows:

– Gen: *key generation algorithm*
  This is a probabilistic algorithm that takes as input a security parameter 1^k and the total number of time periods N. It returns a master key SK^*, a verification key PK, and an initial signing key SK_0.
– Upd^*: *partial key generation algorithm*
  This is a probabilistic algorithm that takes as input indices i and j for time periods and a master key SK^*. It returns a partial key $SK'_{i,j}$.
– Upd: *key update algorithm*
  This is a deterministic algorithm that takes as input indices i, j, a signing key SK_i, and a partial key $SK'_{i,j}$. It returns a signing key SK_j for time period j.
– Sign: *signing algorithm*
  This is a probabilistic algorithm that takes as input an index j for a time period, a message M, and a signing key SK_j. It returns a pair $\langle j, s\rangle$ consisting of j and a signature s.
– Vrfy: *verification algorithm*
  This is a deterministic algorithm that takes as input a verification key PK, a message M, and a pair $\langle j, s\rangle$. It returns a bit b, where $b = 1$ means that the signature is accepted.

A time period is used when verifying signatures, and the verification key does not have to be updated even if the signing key is updated.

2.2 Security

We next address the security definition of a strong KIS scheme. First, we define the security of a KIS scheme; then we define the security of a *strong* KIS scheme. Intuitively, a KIS scheme is secure against a signing key leakage, whereas a strong KIS scheme is secure against either signing key or master key leakage.

Security Definition of Key-Insulated Signature Scheme. An attack on a KIS scheme means that a signing key was leaked by a signer and used by a third party. To model this attack formally, we give an adversary access to a key exposure oracle $\mathsf{Exp}_{SK^*,SK_0}(\cdot)$ and a signing oracle $\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot)$. $\mathsf{Exp}_{SK^*,SK_0}(\cdot)$ takes as input a time period i and returns a signing key SK_i. $\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot)$ takes as input a time period i and a message m, and it returns a signature $\sigma_{i,m}$. The KIS scheme is considered secure if verification of a signature that the adversary forges for any message has negligible probability of success. We formally define the security of a KIS scheme as follows:

Definition 2. Let $\Pi = (\mathsf{Gen}, \mathsf{Upd}^*, \mathsf{Upd}, \mathsf{Sign}, \mathsf{Vrfy})$ be a KIS scheme. Let $\mathcal{A}$ be an adversary. Define the success probability of signature forgery by $\mathcal{A}$ as follows:

$$\mathsf{Succ}_{\mathcal{A},\Pi}(k) := Pr\left[\mathsf{Vrfy}_{PK}(m^*, i^*, \sigma_{i^*,m^*}) = 1 \right|$$

$$(SK^*, PK, SK_0) \leftarrow \mathsf{Gen}(1^k, N),$$
$$(m^*, i^*, \sigma_{i^*,m^*}) \leftarrow \mathcal{A}^{\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot),\mathsf{Exp}_{SK^*,SK_0}(\cdot)}(PK)\Big],$$

where $\mathcal{A}$ cannot submit the query (i^*, m^*) to $\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot)$ and cannot submit the query i^* to $\mathsf{Exp}_{SK^*,SK_0}(\cdot)$. $\mathcal{A}$ is allowed to submit a query to the key exposure oracle up to t times, and if $\mathsf{Succ}_{\mathcal{A},\Pi}(k)$ is negligible, Π is (t, N)-*key-insulated*. If Π is $(N-1, N)$-*key-insulated*, Π is *perfectly key-insulated*.

Security Definition of Strong Key-Insulated Signature Scheme. Regarding a strong KIS scheme, a key leakage attack means that either a signing key or a master key was leaked by a signer and used by a third party. To model this attack formally, we give an adversary the master key and access to a signing oracle $\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot)$. $\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot)$ takes as input a time period i and a message m, and it returns a signature $\sigma_{i,m}$. The strong KIS scheme is considered secure if verification of the signature that the adversary forges for any message has negligible probability of success. We formally define the security of a strong KIS scheme as follows:

Definition 3. Let $\Pi = (\mathsf{Gen}, \mathsf{Upd}^*, \mathsf{Upd}, \mathsf{Sign}, \mathsf{Vrfy})$ be a KIS scheme which is (t, N)-*key-insulated*. Let $\mathcal{B}$ be an adversary. Define the success probability of signature forgery by $\mathcal{B}$ as follows:

$$\mathsf{Succ}_{\mathcal{B},\Pi}(k) := Pr\Big[\mathsf{Vrfy}_{PK}(m^*, i^*, \sigma_{i^*,m^*}) = 1 \Big|$$
$$(SK^*, PK, SK_0) \leftarrow \mathsf{Gen}(1^k, N),$$
$$(m^*, i^*, \sigma_{i^*,m^*}) \leftarrow \mathcal{B}^{\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot)}(PK, SK^*)\Big],$$

where $\mathcal{B}$ cannot submit the query (i^*, m^*) to $\mathsf{Sign}_{SK^*,SK_0}(\cdot,\cdot)$. Then, if $\mathsf{Succ}_{B,\Pi}(k)$ is negligible, Π is *strong* (t, N)-*key-insulated*. If Π is *strong* $(N-1, N)$-*key-insulated*, Π is *perfectly strong key-insulated*.

3 Our Construction

We propose an efficient strong KIS scheme that is secure against leakage of either signing keys or a master key.

3.1 Basic Concept

Our goal is to design an *efficient* strong KIS scheme with a significantly shorter signature size.

We construct the KIS scheme by extending the Abe-Okamoto proxy signature scheme [1]. The Abe-Okamoto scheme is based on the Schnorr signature scheme, and it is the most efficient proxy signature scheme in terms of verification cost and communication cost. However, constructing an efficient and strong KIS scheme from the Abe-Okamoto scheme is not a trivial exercise.

First, intending more security, we extend the Abe-Okamoto scheme to a KIS scheme that provides *adaptive security*. Adaptive security is security against an adversary who can get signing keys at any time and in any order, and it is not taken into consideration in the security definition of [1]. Consequently, we must address adaptive security with a formal security proof from scratch.

We then extend this scheme into one that provides *strong key-insulated* security without increasing the signature size. Malkin, Obana, and Yung [9] devised a generic method for converting any proxy signature scheme into a KIS scheme, but the resulting KIS scheme does not have *strong key-insulated* security. In [4], Dodis, Katz, Xu, and Yung showed that a KIS scheme can be converted into a *strong* KIS scheme. Their construction method, however, employs double signing. That is, the scheme needs two signatures: a signature with the master key and a signature with the signer's secret key. Therefore, the scheme is not efficient. (This construction is shown in Appendix A.) Thus, for the sake of efficiency, we have to construct a scheme without the above conversions.

3.2 Proposed Strong Key-Insulated Signature Scheme

The algorithms for our strong KIS scheme are constructed as follows:

- Gen: *key generation algorithm*
  Let $\mathbb{G}_q$ be a cyclic group of prime order q. Randomly select $g \in \mathbb{G}_q$, where g is a generator of $\mathbb{G}_q$. Then, randomly select x and $x' \in Z_q$. The master key $x_0 = x - x'$ is stored in a secure device, and x' is managed by a signer. Calculate $y_0 = g^{x_0}$ and $y' = g^{x'}$, and publish the verification key $PK = \langle q, g, y_0, y', G(\cdot,\cdot), H(\cdot,\cdot,\cdot,\cdot)\rangle$. Here, G and H are hash functions, where $G : \mathbb{G}_q \times \{0,1\}^* \to Z_q$, and $H : \mathbb{G}_q \times \mathbb{G}_q \times \{0,1\}^* \times \{0,1\}^* \to Z_q$.
- Upd*: *partial key generation algorithm*
  Randomly select $r_1 \in Z_q$ from a secure device, and calculate $v_1 = g^{r_1}$. Then, calculate $c_1 = G(v_1, T)$ using the inputted time period T, and obtain a partial key $x_1 = c_1 r_1 + x_0 \bmod q$ using the inputted master key x_0. x_1, v_1, and T are transmitted to the signer.
- Upd: *key-update algorithm*
  The signer calculates $c_1 = G(v_1, T)$ and verifies whether $g^{x_1} = v_1^{c_1} y_0$ is correct by using the inputted v_1, T, x_1, and y_0. If and only if the partial key verification is successful, the signer obtains the signing key $SK_T = x_1 + x' \bmod q$ for a time period T by using x_1 and the inputted x'.
- Sign: *signing algorithm*
  The signer randomly selects $r_s \in Z_q$ and calculates $v_s = g^{r_s}$. Then, the signer calculates $c_s = H(v_1, v_s, T, m)$ and $\sigma_s = c_s r_s + SK_T \bmod q$ by using the inputted message m, T, the signing key SK_T, and v_1. Finally, the signer transmits m, (σ_s, c_s, v_1), and T to a verifier.
- Vrfy: *verification algorithm*
  Using the inputted y_0, y', m, (σ_s, c_s, v_1), and T, the verifier calculates $c_1 = G(v_1, T)$. If the following equation holds, it returns $b = 1$; otherwise, it returns $b = 0$:
  $$c_s = H(v_1, (g^{\sigma_s}(v_1^{c_1} y_0 y')^{-1})^{1/c_s}, T, m).$$

$\boxed{\text{Secure device}}$		$\boxed{\text{Signer}}$
(x_0)		(x', y_0)
$r_1 \in_U Z_q$		
$v_1 = g^{r_1}$		
$c_1 = G(v_1, T)$		
$x_1 = c_1 r_1 + x_0 \bmod q$		
	$\xrightarrow{x_1, v_1, T}$	
		$c_1 = G(v_1, T)$
		$g^{x_1} \stackrel{?}{=} v_1^{c_1} y_0$
		$SK_T = x_1 + x' \bmod q$

Fig. 1. Partial key generation and key-update algorithms

$\boxed{\text{Signer}}$		$\boxed{\text{Verifier}}$
(SK_T, v_1, T)		(y_0, y')
$r_s \in_U Z_q$		
$v_s = g^{r_s}$		
$c_s = H(v_1, v_s, T, m)$		
$\sigma_s = c_s r_s + SK_T \bmod q$		
	$\xrightarrow{m, (\sigma_s, c_s, v_1), T}$	
		$c_1 = G(v_1, T)$
		$c_s \stackrel{?}{=} H(v_1, (g^{\sigma_s}(v_1^{c_1} y_0 y')^{-1})^{1/c_s}, T, m)$

Fig. 2. Signing and verification algorithms

Figure 1 illustrates Upd^* and Upd, and Fig. 2 illustrates Sign and Vrfy.

Remark 1. In the partial key generation and key update algorithms, the signer can verify whether the partial key transmitted from the secure device is valid. This is the feature that conventional strong KIS schemes lack.

Remark 2. If the secure device storing the master key x_0 is completely reliable, partial key verification is unnecessary during the signing key update. That is, one of the verification keys can be $y := y_0 y'$, instead of y_0 and y'. This implies that the verification key size can be reduced by half.

4 Security Analysis

We prove the security of our scheme from scratch in accordance with the security definition of a KIS scheme in [4]. As described in Section 3.1, we can not straightforwardly apply the security proof in [1] to our scheme, since the security definition of [1] does not consider an adaptive adversary, who can get signing keys at any time and in any order.

4.1 Overview of Security Proof

An adaptive adversary can get signing keys and signatures at any time and in any order. Hence, the security proof assumes that she can access a signing oracle and a key exposure oracle adaptively and all oracles have to answer her queries correctly. More precisely, the oracles have to simulate key exposures and signing perfectly. This simulation is not trivial.

In the following proof, we construct an adversary $\mathcal{B}_{ms}$ which can forge the signature of the *modified Schnorr signature scheme* by using an adversary $\mathcal{A}_p$ which can forge the signature of our scheme. $\mathcal{B}_{ms}$ expects that $\mathcal{A}_p$ forges a signature by using specific information that $\mathcal{B}_{ms}$ embeds into a certain oracle answer. The probability that $\mathcal{A}_p$ uses the embedded information is $1/\mathsf{poly}$. When $\mathcal{A}_p$ uses the information for the forgery, simulation of all the oracles is perfect and $\mathcal{B}_{ms}$ can forge a signature of the *modified Schnorr signature scheme* from $\mathcal{A}_p$'s output.

Furthermore, we consider two cases of key exposure: signing key leakage and master key leakage.

The proof can be sketched as follows: First, we show that the *modified Schnorr signature scheme* is EUF-ACMA secure if the discrete logarithm problem (DLP) is infeasible. We then show that our scheme is *perfectly key-insulated* if the modified Schnorr signature scheme is EUF-ACMA secure. Finally, we show that our scheme is *perfectly strong key-insulated* if it is *perfectly key-insulated.* This shows that our scheme is *perfectly strong key-insulated* assuming the infeasibility of the DLP.

4.2 Security Proof

Modified Schnorr Signature Scheme: The *modified Schnorr signature scheme* is constructed as follows:

– Gen: *key generation algorithm*
  Let $\mathbb{G}_q$ be a cyclic group of prime order q. Randomly select $g \in \mathbb{G}_q$, where g is a generator of $\mathbb{G}_q$. Then, randomly select $x \in Z_q$. Calculate $y = g^x$, and publish the verification key $PK = \langle q, g, y, G(\cdot,\cdot)\rangle$. Here, G is a hash function, where $G : \mathbb{G}_q \times \{0,1\}^* \to Z_q$.
– Sign: *signing algorithm*
  The signer selects $r \in Z_q$ randomly and calculates $v = g^r$. Then, the signer calculates $c = G(v, m)$ and $\sigma = cr + x \bmod q$, by using the inputted message m and its signing key x. Finally, the signer transmits m and (v, σ) to a verifier.
– Vrfy: *verification algorithm*
  Using the inputted m, (v, σ), and y, the verifier calculates $c = G(v, m)$. Then, if the following equation holds, it returns $b = 1$; otherwise, it returns $b = 0$:

$$g^\sigma = v^c y.$$

Definition 4. Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme. Let $\mathcal{F}$ be an adversary, which can access a signing oracle $\mathsf{Sign}(\cdot)$ that takes as input a message

m and returns a signature σ. Define the success probability of signature forgery by $\mathcal{F}$ as follows:

$$\begin{aligned}\mathsf{Succ}_{\mathcal{F},\Pi}(k) := Pr\Big[\mathsf{Vrfy}_{PK}(m^*, \sigma^*) = 1 \,\Big|\, \\ PK \leftarrow \mathsf{Gen}(1^k), \\ (m^*, \sigma^*) \leftarrow \mathcal{F}^{\mathsf{Sign}(\cdot)}(PK)\Big],\end{aligned}$$

where $\mathcal{F}$ cannot submit the query m^* to the signing oracle. If $\mathsf{Succ}_{\mathcal{F},\Pi}(k)$ is negligible, Π is EUF-ACMA secure.

Lemma 1. The *modified Schnorr signature scheme* is EUF-ACMA secure, assuming the infeasibility of the DLP.

Proof. We assume that there exists a probabilistic polynomial time adversary $\mathcal{A}$ that can forge a modified Schnorr signature. Then, according to the forking lemma [12], $\mathcal{A}$ outputs two valid tuples of message and signature, (m, v, c, σ) and (m, v, c', σ'), with non-negligible probability in polynomial time. Therefore, we can get the two equations $\sigma = cr + x$ and $\sigma' = c'r + x$. Solving these equations for r and x, we obtain

$$x = \frac{c'\sigma - c\sigma'}{c' - c} \bmod q.$$

This implies that the discrete logarithm x of $y = g^x$ can be obtained. That contradicts the assumption of DLP infeasibility. Therefore, there does not exist a probabilistic polynomial time adversary $\mathcal{A}$ that can forge a modified Schnorr signature. □

Lemma 2. The KIS scheme proposed in Section 3.2 is *perfectly key-insulated*, assuming that the *modified Schnorr signature scheme* is EUF-ACMA secure.

Proof. We prove Lemma 2 according to the Definition 2.

We assume that there exists a probabilistic polynomial time adversary $\mathcal{A}_p$ that can forge a signature in our scheme. Then, we show that there exists a probabilistic polynomial time adversary $\mathcal{B}_{ms}$ that can forge a modified Schnorr signature.

Let $\mathbb{G}_q$ be a cyclic group of prime order q. Randomly select $g \in \mathbb{G}_q$, where g is a generator of $\mathbb{G}_q$. Then, randomly select a master key $x_0 \in Z_q$, and calculate $y_0 = g^{x_0}$. Next, we assume that q, g, and y_0 are given to a simulator $\mathcal{B}_{ms}$. $\mathcal{B}_{ms}$ selects $x' \in Z_q$ randomly and calculates $y' = y_0 g^{x'}$. Then, $\mathcal{B}_{ms}$ transmits q, g, and y' to $\mathcal{A}_p$.

$\mathcal{A}_p$ can access key exposure oracles $\mathsf{Exp}(\cdot)$. $\mathsf{Exp}(\cdot)$ is a key exposure oracle that takes as input a time period and returns a signing key. $\mathcal{A}_p$ is not allowed to input the target time period into $\mathsf{Exp}(\cdot)$, but she can access a signing oracle $\mathsf{Sign}(\cdot,\cdot)$, which takes as input a message m and a time period T and returns a signature (σ_s, v_s, v_1). $\mathcal{A}_p$ also can access random oracles $\mathsf{G}(\cdot,\cdot)$ and $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$. $\mathcal{A}_p$ is not, however, allowed to submit the target tuple of a message and a time period to $\mathsf{Sign}(\cdot,\cdot)$. On the other hand, $\mathcal{B}_{ms}$ can access a signing oracle $\mathsf{Sign}'(\cdot)$

that takes as input a message and returns a signature. $\mathcal{B}_{ms}$ can also access a random oracle $\mathsf{G}'(\cdot,\cdot)$ but is not allowed to submit the target message to $\mathsf{Sign}'(\cdot)$. Next, we show that $\mathcal{B}_{ms}$ can simulate $\mathsf{Exp}(\cdot)$ and $\mathsf{Sign}(\cdot,\cdot)$.

We show how $\mathcal{B}_{ms}$ simulates $\mathsf{Sign}(\cdot,\cdot)$ when $\mathcal{A}_p$ transmits a message m and a time period T to $\mathsf{Sign}(\cdot,\cdot)$. There are two cases: $\mathcal{A}_p$ requests the signing key for a time period T from $\mathsf{Exp}(\cdot)$ before $\mathcal{A}_p$ accesses $\mathsf{Sign}(\cdot,\cdot)$, or $\mathcal{A}_p$ does not request the signing key beforehand.

Case1 : Request the signing key beforehand

First, $\mathcal{A}_p$ requests the signing key. $\mathcal{A}_p$ transmits T to $\mathsf{Exp}(\cdot)$. $\mathsf{Exp}(\cdot)$ then transmits T to $\mathsf{Sign}'(\cdot)$ directly. $\mathsf{Sign}'(\cdot)$ randomly selects $r_1 \in Z_q$ and calculates $v_1 = g^{r_1}$. Then, $\mathsf{Sign}'(\cdot)$ transmits v_1 and T to the random oracle $\mathsf{G}'(\cdot,\cdot)$, which returns c_1 to $\mathsf{Sign}'(\cdot)$. $\mathsf{Sign}'(\cdot)$ obtains $x_1 = c_1 r_1 + x_0$ by using the master key x_0 and sends (x_1, v_1) to $\mathsf{Exp}(\cdot)$. $\mathsf{Exp}(\cdot)$ adds (T, x_1, v_1) to the list (*List1*), which indicates the input-output relation of the signing oracle $\mathsf{Sign}'(\cdot)$. Finally, $\mathsf{Exp}(\cdot)$ calculates the signing key $SK_T = x_1 + x'$ by using x' and transmits (SK_T, v_1, x') to $\mathcal{A}_p$.

Next, $\mathcal{A}_p$ requests a signature. $\mathcal{A}_p$ transmits m and T to $\mathsf{Sign}(\cdot,\cdot)$. $\mathsf{Sign}(\cdot,\cdot)$ calculates the signature (σ_s, v_s, v_1) by using SK_T and v_1, which have already been obtained through the signing key request. Finally, $\mathsf{Sign}(\cdot,\cdot)$ transmits (σ_s, v_s, v_1) to $\mathcal{A}_p$.

Case2 : Never request the signing key beforehand

If $\mathcal{A}_p$ requests a signature from $\mathcal{B}_{ms}$, $\mathcal{B}_{ms}$ normally responds as follows:

$\mathcal{A}_p$ transmits m and T to $\mathsf{Sign}(\cdot,\cdot)$, which accesses $\mathsf{Exp}(\cdot)$ and obtains SK_T, v_1, and x'. $\mathsf{Sign}(\cdot,\cdot)$ calculates (σ_s, v_s, v_1) for (m, T) by using (SK_T, v_1) and transmits (σ_s, v_s, v_1) to $\mathcal{A}_p$.

$\mathcal{A}_p$ is not, however, allowed to transmit the target (m^*, T^*) to $\mathsf{Sign}(\cdot,\cdot)$, and $\mathcal{B}_{ms}$ is not allowed to transmit the target T^* to $\mathsf{Sign}'(\cdot)$. Therefore, $\mathcal{A}_p$ can get no information about (m^*, T^*) from those responses.

Here, $\mathcal{B}_{ms}$ takes the following strategy. We assume that $\mathcal{A}_p$ requests the hash value for the target T^* from the random oracle $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$ when $\mathcal{A}_p$ forges the signature. $\mathsf{Sign}(\cdot,\cdot)$ calculates the tuple of (v_1, v_s, T^*, m) and c_s, and it adds these to the list (*List2*), which indicates the input-output relation of $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$. Let q_H be the number of queries to $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$. $\mathcal{B}_{ms}$ expects that the jth query is the request for (m^*, T^*). If this expectation is true, the simulation of $\mathcal{B}_{ms}$ is successful. To be more precise, $\mathcal{B}_{ms}$ responds to the signing request of $\mathcal{A}_p$ as follows:

$\mathcal{A}_p$ transmits m and T^* to $\mathsf{Sign}(\cdot,\cdot)$, which randomly selects $r_1 \in Z_q$ and calculates $v_1 = g^{r_1}$. Then, $\mathsf{Sign}(\cdot,\cdot)$ transmits (v_1, T^*) to the random oracle $\mathsf{G}'(\cdot,\cdot)$, which returns c_1 to $\mathsf{Sign}(\cdot,\cdot)$. $\mathsf{Sign}(\cdot,\cdot)$ then randomly selects $c_s, \sigma_s \in Z_q$ and calculates

$$v_s = \left(\frac{g^{\sigma_s}}{v_1^{c_1} y'}\right)^{1/c_s}.$$

$\mathsf{Sign}(\cdot,\cdot)$ adds the tuple of (v_1, v_s, T^*, m) and c_s to *List2*. Finally, $\mathsf{Sign}(\cdot,\cdot)$ transmits (σ_s, v_s, v_1) to $\mathcal{A}_p$.

According to the forking lemma [12], $\mathcal{A}_p$ outputs two valid signatures (m^*, (σ_s, c_s, v_1), T^*) and (m^*, (σ'_s, c'_s, v_1), T^*) for the message m^* and the time period T^* with non-negligible probability in polynomial time. Therefore, $\mathcal{B}_{ms}$ can get the two equations $\sigma_s = c_s r_s + SK_{T^*}$ and $\sigma'_s = c'_s r_s + SK_{T^*}$. Solving these equations for r_s and SK_{T^*}, we get

$$SK_{T^*} = \frac{c'_s \sigma_s - c_s \sigma'_s}{c'_s - c_s} \bmod q.$$

$\mathcal{B}_{ms}$ calculates $x_1^* = SK_{T^*} - x'$ and outputs (x_1^*, v_1, T^*) as the modified Schnorr signature for T^*.

Let $\epsilon_{\mathcal{B}_{ms}}$ be the success probability of $\mathcal{B}_{ms}$ forging (x_1^*, v_1, T^*). Let $\epsilon_{\mathcal{A}_p}$ be the success probability of $\mathcal{A}_p$ forging (m^*, (σ_s, c_s, v_1), T^*) and (m^*, (σ'_s, c'_s, v_1), T^*). Then, $\epsilon_{\mathcal{B}_{ms}}$ is equal to $\epsilon_{\mathcal{A}_p}$ times the probability that the expectation of a request to the random oracle $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$ is true. That is,

$$\epsilon_{\mathcal{B}_{ms}} = \epsilon_{\mathcal{A}_p} \cdot \frac{1}{q_H}.$$

Let $\mathcal{Q}_1$, $\mathcal{Q}_2$, ..., $\mathcal{Q}_{q_H}$ be the inputs to the random oracle $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$. Let ρ_1, ρ_2, ..., ρ_{q_H} be the outputs from $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$. We assume that $\mathcal{A}_p$ outputs the signature $(m^*, (\sigma_s, c_s, v_1), T^*)$ for $\mathcal{Q}_j = (v_1, v_s, T^*, m^*)$ with a probability greater than or equal to $1/P(n)$. (Let $P(n)$ be the polynomial in n, the length of the verification key). Then, according to the forking lemma [12], there exists the set Ω_j of random tape ω and the set $R_{j,\omega}$ of outputs (ρ_1, ρ_2, ..., ρ_{j-1}) from the random oracle, such that $\mathcal{A}_p$ outputs two valid signatures $(m^*, (\sigma_s, c_s, v_1), T^*)$ and $(m^*, (\sigma'_s, c'_s, v_1), T^*)$ for $\mathcal{Q}_j = (v_1, v_s, T^*, m^*)$ with a probability greater than or equal to $1/4P(n)$. That is,

$$\epsilon_{\mathcal{A}_p} \geq \frac{1}{4P(n)}.$$

Therefore,

$$\epsilon_{\mathcal{B}_{ms}} \geq \frac{1}{4q_H P(n)}.$$

As a result, $\mathcal{B}_{ms}$ can forge the modified Schnorr signature by using $\mathcal{A}_p$ with non-negligible probability in polynomial time. □

Lemma 3. The KIS scheme proposed in Section 3.2 is *perfectly strong key-insulated*, assuming that the KIS scheme is *perfectly key-insulated*.

Proof. We prove Lemma 3 according to the Definition 3.

Let $\mathbb{G}_q$ be a cyclic group of prime order q. Randomly select $g \in \mathbb{G}_q$, where g is a generator of $\mathbb{G}_q$. Then, randomly select $x' \in Z_q$, and calculate $y' = g^{x'}$. Next, we assume that q, g, and y' are given to a simulator $\mathcal{B}_{ms}$. $\mathcal{B}_{ms}$ randomly selects a master key $x_0 \in Z_q$ and calculates $y_0 = y' g^{x_0}$. Then, $\mathcal{B}_{ms}$ transmits q, g, y_0, and x_0 to $\mathcal{A}_p$.

$\mathcal{A}_p$ can access a signing oracle $\mathsf{Sign}(\cdot,\cdot)$, which takes as input a message m and a time period T and returns a signature (σ_s, v_s, v_1). $\mathcal{A}_p$ also can access random oracles $\mathsf{G}(\cdot,\cdot)$ and $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$. $\mathcal{A}_p$ is not, however, allowed to submit the target tuple of a message and a time period to $\mathsf{Sign}(\cdot,\cdot)$. On the other hand, $\mathcal{B}_{ms}$ can access a signing oracle $\mathsf{Sign}'(\cdot)$ that takes as input a message and returns a signature. $\mathcal{B}_{ms}$ can also access the random oracle $\mathsf{G}'(\cdot,\cdot)$ but is not allowed to submit the target message to $\mathsf{Sign}'(\cdot)$. Next, we show that $\mathcal{B}_{ms}$ can simulate $\mathsf{Sign}(\cdot,\cdot)$.

$\mathcal{A}_p$ transmits m and T to $\mathsf{Sign}(\cdot,\cdot)$. $\mathsf{Sign}(\cdot,\cdot)$ then transmits T to $\mathsf{Sign}'(\cdot)$ directly. $\mathsf{Sign}'(\cdot)$ randomly selects $r_1 \in Z_q$ and calculates $v_1 = g^{r_1}$. Then, $\mathsf{Sign}'(\cdot)$ transmits v_1 and T to $\mathsf{G}'(\cdot,\cdot)$, which returns c_1 to $\mathsf{Sign}'(\cdot)$. $\mathsf{Sign}'(\cdot)$ obtains $x_1 = c_1r_1 + x'$ and sends (x_1, v_1) to $\mathsf{Sign}(\cdot,\cdot)$. $\mathsf{Sign}(\cdot,\cdot)$ then calculates $SK_T = x_1 + x_0 = c_1r_1 + x' + x_0$. $\mathsf{Sign}(\cdot,\cdot)$ also randomly selects $r_s \in Z_q$ and calculates $v_s = g^{r_s}$. $\mathsf{Sign}(\cdot,\cdot)$ transmits the tuple of (v_1, v_s, T, m) to $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$, which returns c_s to $\mathsf{Sign}(\cdot,\cdot)$. Finally, $\mathsf{Sign}(\cdot,\cdot)$ obtains $\sigma_s = c_sr_s + SK_T = c_sr_s + x_0 + c_1r_1 + x'$ and transmits (σ_s, v_s, v_1) to $\mathcal{A}_p$.

We assume that $\mathcal{A}_p$ outputs a valid signature $(m^*, (\sigma_s^*, c_s^*, v_1^*), T^*)$ for the message m^* and the time period T^* with non-negligible probability in polynomial time. Then, $\mathcal{B}_{ms}$ randomly selects $r_1^* \in Z_q$ and $r_s^* \in Z_q$ and calculates $v_1^* = g^{r_1^*}$ and $v_s^* = g^{r_s^*}$. $\mathcal{B}_{ms}$ transmits v_1^* and T^* to $\mathsf{G}'(\cdot,\cdot)$, which returns c_1^* to $\mathcal{B}_{ms}$. $\mathcal{B}_{ms}$ also transmits v_1^*, v_s^*, T^*, and m^* to $\mathsf{H}(\cdot,\cdot,\cdot,\cdot)$, which returns c_s^* to $\mathcal{B}_{ms}$. Finally, $\mathcal{B}_{ms}$ calculates $x_1^* = \sigma_s^* - (c_s^*r_s^* + x_0)$, and outputs (x_1^*, v_1^*, T^*) as the modified Schnorr signature for T^*.

Let $\epsilon_{\mathcal{B}_{ms}}$ be the success probability of $\mathcal{B}_{ms}$ forging (x_1^*, v_1^*, T^*). Let $\epsilon_{\mathcal{A}_p}$ be the success probability of $\mathcal{A}_p$ forging $(m^*, (\sigma_s^*, c_s^*, v_1^*), T^*)$. Then,

$$\epsilon_{\mathcal{B}_{ms}} = \epsilon_{\mathcal{A}_p}.$$

We assume that $\mathcal{A}_p$ outputs the signature $(m^*, (\sigma_s^*, c_s^*, v_1^*), T^*)$ with a probability greater than or equal to $1/P(n)$. (Let $P(n)$ be a polynomial in n, the length of the verification key). That is,

$$\epsilon_{\mathcal{A}_p} \geq \frac{1}{P(n)}.$$

Therefore,

$$\epsilon_{\mathcal{B}_{ms}} \geq \frac{1}{P(n)}.$$

As a result, $\mathcal{B}_{ms}$ can forge the modified Schnorr signature by using $\mathcal{A}_p$ with non-negligible probability in polynomial time. □

From Lemma 1–3, we obtain the following theorem:

Theorem 1. Our scheme is *perfectly strong key-insulated*, assuming the infeasibility of the DLP.

5 Performance

Table 1 lists results comparing the strong KIS scheme proposed in Section 3.2 and the conventional strong KIS schemes: the certificate-based scheme using Schnorr

Table 1. Comparison of our scheme and the conventional strong KIS schemes: a certificate-based scheme using Schnorr signatures (CB scheme) and the scheme based on Guillou-Quisquater signature [4] (GQ scheme). CB and our scheme are constructed over elliptic curves.

	CB scheme	GQ scheme	Our scheme
Verification key size (bits)	320	1024	160
Signature size (bits)	1120	1184	480
Computational cost (signing)	720	1776	240
Computational cost (verification)	1440	1776	720
Security assumption	DL	RSA	DL

signatures (we call this the *CB scheme*, as shown in Appendix A), and the scheme based on Guillou-Quisquater signature [4] (we call this the *GQ scheme*, as shown in Appendix B), in terms of verification key size, signature size, computational cost for signing and verification, and security assumption. The verification key does not include the system parameters: $q, g, G(\cdot,\cdot), H(\cdot,\cdot,\cdot,\cdot)$. The computational cost means the number of multiplications for modulo exponentiation[1]. We assume that the CB scheme and our scheme are constructed over elliptic curves. The size of both elements of $\mathbb{G}_q$ and Z_q are 160 bits. We also assume that the size of element of Z_n^* is 1024 bits and the output length of hash functions is 160 bits.

The signature size in our scheme is about two fifths that of the conventional schemes. More precisely, the signature sizes of CB, GQ, and our scheme are 1120, 1184, and 480 bits, respectively. It is recommended that the signature size be as short as possible, especially in the services where a signer communicates with a huge number of receivers. Our scheme is also efficient in terms of verification key size and computational cost for signing and verification.

6 Application

A typical application of our scheme is to an authentication system in which one (or a small number of) sender communicates with many receivers since in such a system, multiple copies of the sender's signature are transmitted to individual receivers. In particular, we discuss a bidirectional broadcasting service.

In this type of service, such as for TV shopping or watching quiz programs, users must transmit personal information, such as names, addresses, and phone numbers, to a broadcaster through a network. The broadcaster distributes content and manages the users' information. A user has a terminal (digital TV) to receive content, and a tamper-resistant module (TRM), such as a smart card, is inserted into the terminal. The broadcaster transmits personal information requests to the users. Each terminal then transmits the requested personal

[1] In this paper, we assume that the number of multiplications for g^a is $1.5|a|$, where $|a|$ means the bit length of a.

information back to the broadcaster, which receives the information and stores it in a database for several services.

When users employ such bidirectional services, broadcasters must be authenticated using digital signatures in order to protect themselves against impersonation. That is, when a broadcaster transmits a personal information request to a user, it signs for the request with its signing key. The user's terminal receives the request and verifies the signature with a verification key stored in the TRM. The request issuer is authenticated and confirmed through this process, and the user can feel safe in providing personal information to the broadcaster.

Ohtake, Hanaoka, and Ogawa proposed a bidirectional content distribution system that enables secure provider authentication against key leakage [11]. This system uses a KIS scheme for provider authentication, and the broadcaster periodically updates its signing key to reduce the damage caused by key leakage. The authors, however, do not concretely specify any KIS scheme. Our scheme could be applied to this system.

Since there are many users in a bidirectional broadcasting service, a great amount of signed data is transmitted through the network, and the memory size of each TRM for storing a verification key is limited, our scheme would be especially effective for this system. The effect of applying it would be as follows:

– Reduced damage due to master key leakage
  The master key x_0 and x' are managed by two different servers. Even if x_0 is leaked, the signing key cannot be updated without x'.
– Efficient signing and verification
  A verification key is stored in each TRM. The length of the verification key in our scheme is 160 bits, which is shorter than in the conventional strong KIS scheme and thus suitable for such a small-memory-capacity device.
  Moreover, the network cost for transmitting signed messages is reduced. As mentioned before, there are more than 35,000,000 users of Japanese broadcasting services. The signature size of the certificate-based strong KIS scheme is 1120 bits. If this scheme were applied to all users, the total amount of signature data would be 39.2 ($= 1120 \times 35{,}000{,}000$) Gbits. In contrast, the signature size of our scheme is 480 bits. The total amount of its signature data would be only 16.8 ($= 480 \times 35{,}000{,}000$) Gbits, a reduction of 22.4 Gbits.
– Application of sources and libraries for the Schnorr signature scheme
  In developing a security system using our signature scheme, we can directly use the existing sources and libraries for the Schnorr signature scheme. As a result, the development cost for a security system can be reduced.

References

1. Abe, M., Okamoto, T.: Delegation Chains Secure up to Constant Length. IEICE Trans. Fundamentals E85-A(1), 110–116 (2002)
2. Bellare, M., Miner, S.: A Forward-Secure Digital Signature Scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)

3. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
4. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong Key-Insulated Signature Schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Heidelberg (2002)
5. Gonzalez-Deleito, N., Markowitch, O., Dall'Olio, E.: A New Key-Insulated Signature Scheme. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 465–479. Springer, Heidelberg (2004)
6. Guillou, L.C., Quisquater, J.-J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing Both Transmission and Memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)
7. Itkis, G., Reyzin, L.: SiBIR: Signer-Base Intrusion-Resilient Signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 499–514. Springer, Heidelberg (2002)
8. Le, Z., Ouyang, Y., Ford, J., Makedon, F.: A Hierarchical Key-Insulated Signature Scheme in the CA Trust Model. In: Zhang, K., Zheng, Y. (eds.) ISC 2004. LNCS, vol. 3225, pp. 280–291. Springer, Heidelberg (2004)
9. Malkin, T., Obana, S., Yung, M.: The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 306–322. Springer, Heidelberg (2004)
10. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: Proc. of ACMCCS 1996, pp. 48–57 (1996)
11. Ohtake, G., Hanaoka, G., Ogawa, K.: Provider Authentication for Bidirectional Broadcasting Service with Fixed Verification Key. In: Proc. of ISITA 2006, pp. 155–160 (2006)
12. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
13. Schnorr, C.P.: Efficient signature generation for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
14. Weng, J., Liu, S., Chen, K., Li, X.: Identity-Based Key-Insulated Signature with Secure Key-Updates. In: Lipmaa, H., Yung, M., Lin, D. (eds.) Inscrypt 2006. LNCS, vol. 4318, pp. 13–26. Springer, Heidelberg (2006)
15. Zhou, Y., Cao, Z., Chai, Z.: Identity Based Key Insulated Signature. In: Chen, K., Deng, R., Lai, X., Zhou, J. (eds.) ISPEC 2006. LNCS, vol. 3903, pp. 226–234. Springer, Heidelberg (2006)

A Certificate-Based Strong Key-Insulated Signature Scheme

Here, we show the construction method for the certificate-based strong KIS scheme using Schnorr signatures.

First, the master key $x_0 \in Z_q$ is stored in a secure device, and $x' \in Z_q$ is managed by a signer. Then, publish the verification keys $y_0 = g^{x_0}$ and $y' = g^{x'}$. Second, generate the signing key SK_T and the verification key PK_T for a time period T in the secure device, and send those keys to the signer with a certificate $\text{Sign}_{x_0}(PK_T)$. Here, $\text{Sign}_x(y)$ denotes a Schnorr signature for a message y and using a key x. To a message m, the signer adds a signature $\mathrm{Sign}_{SK_T}(m)$ by using SK_T and a certificate $\mathrm{Sign}_{x'}(m)$ by using x'. Then, the signer sends these to a verifier with PK_T and $\mathrm{Sign}_{x_0}(PK_T)$. The verifier verifies $\mathrm{Sign}_{x_0}(PK_T)$ and $\mathrm{Sign}_{x'}(m)$ by using public keys y_0 and y' and verifies $\mathrm{Sign}_{SK_T}(m)$ by using PK_T.

B Strong Key-Insulated Signature Scheme Based on Guillou-Quisquater Signature

We show the construction method for the strong KIS scheme based on the Guillou-Quisquater signature.

First, randomly select primes p, q; then calculate and publish $n = pq$. Second, randomly select e, where e is a relatively prime to $(p-1)(q-1)$, then calculate $d = 1/e \bmod (p-1)(q-1)$. Publish e as a verification key. Set d_1 and d_2, such that $d_1 + d_2 = d \bmod (p-1)(q-1)$. The master key d_1 is stored in a secure device, and d_2 is managed by a signer. Generate the partial key $x_1 = (1/f(T))^{d_1}$ in the secure device by using d_1 and a time period T. Here, f is a function $f : \{0,1\}^* \rightarrow Z_n^*$. x_1 and T are transmitted to the signer. The signer calculates the signing key $SK_T = x_1 \cdot (1/f(T))^{d_2}$ for a time period T by using d_2. Then, the signer randomly selects k, and calculates $r = k^e \bmod n$. For a message m, the signer calculates $l = h(m, r)$ and $s = k \cdot SK_T^l \bmod n$ using the signing key SK_T and obtains the signature (s, l). Here, h is a hash function $h : \{0,1\}^* \times \{0,1\}^n \rightarrow \{0,1\}^c$, and c is a constant value. The signer sends m, (s, l), and T to a verifier. The verifier calculates $u = s^e \cdot f(T)^l \bmod n$ and $l' = h(m, u)$ using the verification key e and verifies whether $l = l'$ is satisfied or not.

Efficient Generic Forward-Secure Signatures and Proxy Signatures

Basel Alomair, Krishna Sampigethaya, and Radha Poovendran

Network Security Lab. (NSL)
Electrical Engineering Department
University of Washington
{alomair,rkrishna,rp3}@u.washington.edu

Abstract. We propose a generic method to construct forward-secure signature schemes from standard signature schemes. The proposed construction is more computationally efficient than previously proposed schemes. In particular, the key updating operation in the proposed scheme is orders of magnitude more computationally efficient than previous schemes, making it attractive for a variety of applications, such as electronic checkbooks. Another advantage of our proposed scheme is the ability to be easily extended to proxy signature schemes. We define two notions of forward-security in the proxy signature setup, namely, strong forward-secure proxy signatures and weak forward-secure proxy signatures. We then describe a construction of a scheme that satisfies the strong forward-secure proxy signature property.

Keywords: Digital signatures, forward-security, proxy, efficiency.

1 Introduction

1.1 Background

One of the biggest threats to the security of standard digital signatures is from the exposure of the secret (signing) keys. If the secret key of an authorized user is exposed, an adversary with access to the exposed key can forge signatures that are indistinguishable from the signatures of the authorized user. Furthermore, all the signatures of the authorized user become repudiable, even if they have been generated before the key exposure. In order to minimize the damage caused by key exposure, the notion of forward-security in digital signatures was put forth by Anderson in [2] and formalized by Bellare and Miner in [3]. In forward-secure signature schemes (FSS), although an adversary with access to exposed keys can generate valid signatures, the validity of signatures generated prior to the key exposure will remain intact. Consequently, forged signatures with past dates are distinguishable from valid signatures.

A forward-secure signature scheme is a key evolving scheme [3]. In previously proposed schemes [3,1,13,11,5,14,18], time is divided into disjoint intervals, say T periods $t_1, t_2, ..., t_T$; each period t_i has a secret key, while the public key remains the same. At the end of each interval, a new secret key is generated and the secret key corresponding to the previous interval is deleted [3]. Hence, FSS are time dependent; that is, FSS must be correlated in some way to the time when the signature is generated. On the other hand, a verifier of an FSS must also have a mechanism to verify that the signature generated during interval t_i is uniquely related to the secret key that is valid at t_i. To ensure forward-secrecy, however, it is required that old secret keys cannot be computed by unauthorized users based on the knowledge of present or future keys.

As pointed out in [3], it is trivial to design FSS if we allow the size of the registered[1] secret and/or public keys, or the size of the signatures to grow proportionally with the number of intervals T. It is impractical, however, to assume users' ability to register an arbitrary number of keys from the Certificate Authority (CA) [3]. Except for [2], all proposed schemes avoid such an impractical assumption [3,1,13,11,5,14,18].

In the existing literature, there are two major classes of approaches to design FSS. The first approach is to design schemes based on specific number theoretic assumptions [3,1,13,11,5]. The second approach is to apply a generic construction to standard signature schemes [2,3,14,18]. Generic schemes have the advantage of provable-security assuming secure standard signature schemes exist [7]. On the other hand, specific schemes can only be proven secure in the random-oracle model[7]. Another advantage of generic schemes is that they exhibit the added flexibility to be instantiated from different standard signature schemes. This gives generic schemes the room for trading off computational and storage efficiencies by using base schemes with different performance characteristics, rather than being bound to the properties of a specific base scheme [7].

An unsolved problem in previously proposed schemes [3,1,13,11,5,14,18] is the validity of signatures generated within the key exposure interval. Obviously, all signatures generated before the key exposure but within the same interval will be repudiable, since the same key is used throughout the entire interval. Therefore, the design of interval lengths can be a nontrivial task. The longer the interval, the more the signatures generated with the same key, hence, violating the whole idea behind forward-security in digital signatures. On the other hand, shorter intervals will result in a more frequent key updates, even if no signature has been generated during the intervals.

In this paper, we propose a generic construction, that can be applied to any standard signature scheme based on the discrete logarithm problem (DLP)[2], to compose an FSS scheme. Our treatment of forward-security is different; instead of corresponding keys with time intervals, our keys are tied with signatures. That is, each key is used for one and only one signature; after every signature generation, the key is updated independent of time. In our scheme, the key update algorithm is orders of magnitude more computationally efficient than previously proposed schemes. Furthermore, computational and storage efficiencies are very competitive in all parts of the scheme, compared to other generic schemes in [14,18] and achieves better performance overall (Tables 2 and 3).

Another advantage of our scheme is that its extension to proxy signatures is straightforward and intuitive. We define two levels of forward-security in proxy signatures, and propose a generic construction of *forward-secure proxy signature schemes* that achieves the stronger level of security. To the best of our knowledge, the construction of forward-secure proxy signature schemes has not been addressed in the literature.

[1] Registered keys are the ones authenticated by the Certificate Authority.

[2] In fact, the construction can be extended to non-DLP based signature by slight modifications.

1.2 Related Work

In [2], Anderson extended the idea of forward-security in session key exchange protocols introduced in [10,8] to the context of digital signatures. Since then, many schemes have been proposed to design FSS where the parameters sizes (registered private/public keys and signatures) do not depend on the total number of intervals T. These proposals can be divided into two categories: schemes based on specific number theoretic assumptions, and generic schemes constructed from a standard signature scheme as a building block.

Number theoretic schemes. Bellare and Miner [3] formalized the problem introduced by Anderson [2] and suggested the first solution. Their proposed scheme was based on the hardness of factoring the Blum-Williams integers. Although their scheme does not require the number of secret or public keys, nor the length of the signature to grow with the number of intervals T, it does, however, have rather long keys. Other number theoretic schemes appeared later including [1,13,11,5]. Abdalla and Reyzin [1] improved on the scheme of [3] by shortening the length of the keys. Kozlov and Reyzin [13] proposed an FSS with fast key update. Their proposed scheme requires only one modular exponentiation to update the keys; hence, allowing for shorter intervals. Itkis and Reyzin [11] proposed an FSS that requires only two modular exponentiations with short exponents for signing and verifying. Boyen *et al.* [5] proposed a scheme where the update can be performed on encrypted keys as a second layer of security.

Generic approach schemes. A different approach to designing FSS is to use existing standard digital signature schemes as building blocks. Anderson [2] proposed the first scheme, where the signer is required to have a registered secret key for each interval, thus having a secret key size linear in T. Bellare and Miner [3] introduced the binary tree scheme to reduce the required number of registered secret keys to $O(\log T)$. Their improvement came with the cost of longer signatures and longer verification time (both are increased by a factor $O(\log T)$). As argued by Bellare and Miner [3], however, signature schemes with one of the parameters (registered secret/public keys, or signature sizes) growing proportionally with T are considered impractical.

The first practical generic construction was proposed by Krawczyk [14]. In the initialization phase of [14], the signer, using a single pair of registered secret/public keys, generates a total of T certificates, one for each period. The *certificate chain* of length T need not be secret. However, it must be available for the signer to generate valid signatures. If the certificate corresponding to time interval t_i is lost, no signature can be generated during that interval. The other generic scheme was the one proposed by Malkin *et al.*, usually referred to as the MMM scheme [18]. The MMM scheme makes use of Merkle tree-type certification chains combined with ideas from the binary tree scheme of [3]. In the MMM scheme, instead of generating T certificates in the initialization phase, the scheme is divided into subtrees. At the end of each subtree, $\log t_i$ secrets are generated, where t_i denotes the time period. Hence, reducing the secret key storage from $O(T)$ to $O(\log T)$.[3] This reduction, however, is paid for in the key

[3] In the worst case scenario, $i = T$.

update algorithm as their update time is increased by a factor of log t_i compared to [14]. Moreover, unlike the certificate chain [14], their keys must remain secret.

In a different but related work, Ma and Tsudik [17] proposed the forward secure sequential aggregate (fssagg) authentication schemes.

1.3 Our Results

Compared to previously proposed generic FSS [2,14,18] and the tree based scheme of Bellare and Miner [3], our generic construction method exhibits desirable properties, such as:

Time independence. All previously proposed schemes, including number theoretic based schemes, assign keys to time intervals. Except for the MMM scheme [18], all other schemes require the total number of intervals T to be predefined. Thus, the design of short intervals will exhaust the set of private keys rather quickly. On the other hand, the design of longer intervals leads to signing more messages with the same key, violating the principle of FSS.

Although, the MMM scheme [18] does not require the total number of time intervals to be predefined, the efficiency of some of its operations[4] is dependent upon the interval when the signature is generated. That is, the scheme will become less efficient with time. Therefore, the choice of shorter intervals will drive the scheme to be less efficient relatively rapidly.

This problem is overcome in our scheme by demanding that key update is performed immediately after every signature generation, rather than the end of the time interval.

Flexibility. In addition to the flexibility granted by the ability to use different standard schemes as building blocks, our scheme provides a flexibility of its own. It provides options to trade-off between space and computational efficiencies. For example, some parameters evaluated in the key generation stage can be stored in the system to make key updating and signing more computationally efficient. Moreover, some parameters are unrestricted and can be generated in either one of two operations. For example, a modular exponentiation operation can be performed either in the key update operation or the signature generation operation, trading off computational efficiencies in the two operations, depending on application.

Computational efficiency. As a direct consequence of the requirement that key update is performed immediately after signature generation, the key update operation must be performed efficiently. For that, we propose a key update algorithm that is orders of magnitude more efficient than the previous generic schemes [14,18], as can be found in Tables 2 and 3. Furthermore, the proposed scheme achieves excellent overall performance. In [18], Malkin *et al.* provide a comparison of number theoretic approaches [3,1,13,11] with their MMM scheme, showing how the MMM scheme outperforms these number theoretic approach schemes. Since the MMM scheme is arguably the most efficient generic scheme [7,16], we quantify, in Tables 2 and 3, the performances

[4] See Table 2.

of our proposed scheme against other generic construction based schemes, including the MMM scheme, showing how our proposed scheme outperforms those earlier schemes.

Applicability to proxy signature schemes. Our proposed scheme can be easily applied to proxy signature schemes. In fact, as discussed in Section 4, the implementation of forward-secure proxy signature schemes based on our proposed scheme is straightforward. The extension of generic construction to proxy signature schemes has not been addressed in any of the previous proposals for FSS.

The rest of the paper is organized as follows. In Section 2 we review some preliminaries and basic concepts that will be used in the subsequent sections. We present our generic construction in Section 3. In Section 4 we describe the construction of forward-secure proxy signature scheme. We conclude our paper in Section 5.

2 Preliminaries

Throughout the rest of the paper we assume the existence of public key infrastructure, where users have registered private and public key pair (x, y), where x represents the private key and y represents the public key. Depending on context, the term signing key is used interchangeably with the terms private or secret key. Similarly, the terms public and verifying key are used synonymously.

2.1 Definitions

A DLP-based standard signature scheme is defined as follows:

Definition 1 (Standard Digital Signature Scheme). *A standard digital signature scheme* SS= $(\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$*, with* $\mathcal{P}, \mathcal{K}, \mathcal{S}$*, and* $\mathcal{V}$ *being polynomial-time algorithms with the following functionalities.*

1. $\mathcal{P}$ *is a randomized parameter-generating algorithm that, on input* 1^k*, where* k *is a security parameter, outputs a description of a multiplicative group* $\mathcal{G}$*, a generator* g*, and a description of a one-way hash function. These parameters are assumed to be publicly known.*
2. $\mathcal{K}$ *is a randomized key-generating algorithm that takes the output of* $\mathcal{P}$ *as input and outputs a pair of keys* (x, y)*, where* x *is a secret key and* y *is the corresponding public key.*
3. $\mathcal{S}$ *is a possibly randomized signing algorithm that takes as input a message* $M \in \{0, 1\}^*$*, a secret key* x*, and possibly a set of parameters* λ*. Depending on application, no set of parameters might be needed, in which case* λ *is taken to be the empty set. The algorithm outputs a signature* σ *on the message* M*.*
4. $\mathcal{V}$ *is a deterministic verification algorithm that takes as input* (M, y, σ)*, such that:*

$$\mathcal{V}(M, y, \sigma) = \begin{cases} 1, & if \;\; \sigma = \mathcal{S}(M, x, \lambda) \\ 0, & otherwise \end{cases} . \tag{1}$$

Equation (1) demands that the verification algorithm $\mathcal{V}$ *outputs* 1 *only if the signature* σ *on message* M *is generated using the secret key* x *corresponding to the public key* y*. Otherwise put, the verification algorithm* $\mathcal{V}$ *outputs* 1 *only if the signature is valid.*

2.2 Security Parameters

To be able to provide fair comparison for different schemes we need to quantify the cost of performing different operations in each scheme. Time and space complexity is used here to provide such a basis for comparison. For that, as in [18], we define two different security parameters.

> l : a security parameter such that performing an exhaustive search over l-bit sequences is infeasible. This is the security parameter of conventional symmetric cryptography. We assume the output of cryptographic hash function and the size of secret keys in signature schemes are of size l.
> k : a security parameter such that the k-bit composite integers are hard to factor and such that the discrete logarithm problem modulo a k-bit prime is believed to be hard. We assume that the size of public keys in signature schemes are of size k.

Distinguishing between the two parameters is important, because symmetric key cryptography is much more efficient than asymmetric. The factoring problem can be solved in sub-exponential time $\exp(k^{\frac{1}{3}} \log^{\frac{2}{3}} k)$, thus, asymptotically one might need $k \approx O(l^3)$ [18].

Although more efficient multiplication algorithms exist, for simplicity we will assume that multiplying two integers a and b is performed in $O(\text{size}(a)\ \text{size}(b))$ time and modular exponentiation is performed in $O(\text{size}(\mathit{exponent})\ \text{size}(\mathit{modulos})^2)$ time [6]. Throughout the rest of the paper, we will ignore the cost of integer addition and integer comparison.

3 Proposed Generic Construction of Forward Secure Signature Schemes

3.1 The Proposed Scheme

In this section we describe our generic construction method to design FSS. The key point for the forward-security of our scheme is allowing the signer to possess *two pairs of registered keys*. Allowing the signer to possess two pairs of registered keys results in a simple design of FSS that is computationally more efficient than previously proposed schemes. Instead of generating T certificates as in [14], or a secret for every tree leave as in [18], our scheme is based on the generation of a *forward-security chain* R. The forward-security chain is not required to generate signature, as opposed to [14], nor is it required to be kept secret, as opposed to [18]. In fact, it need not be stored in the system and signatures can still be generated and verified. Another advantage of our scheme and the scheme of [14], over the MMM scheme, is that the chain R and the T certificates are generated off-line in the key generation phase.

To describe our construction method, let SS=$(\mathcal{P}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ be a standard digital signature scheme as in Definition 1. Based on SS, the constructed forward-secure signature scheme is FSS=$(\mathcal{P}, \mathcal{K}, \mathcal{FKG}, \mathcal{FS}, \mathcal{FV}, \mathcal{KU})$, where $\mathcal{P}, \mathcal{K}, \mathcal{FGK}, \mathcal{FS}, \mathcal{FV}$, and $\mathcal{KU}$ are polynomial-time algorithms. The algorithms $\mathcal{P}$ and $\mathcal{K}$ are exactly the same as in the base scheme. We further require that the algorithm $\mathcal{K}$ is run twice so the signer will possess two pairs of registered private and public keys (x_1, y_1) and (x_2, y_2). The forward-secure key generation algorithm $\mathcal{FGK}$, the forward-secure signing algorithm $\mathcal{FS}$, the forward-secure verifying algorithm $\mathcal{FV}$, and the forward-secure key update algorithm $\mathcal{KU}$ are described in detail below.

Key Generation. On input of a security parameter l, the user generates a prime p and a prime q that divides $p-1$, such that $q \geq 2^l$. The user picks an element $g \in \mathbb{Z}_p^*$ of order q, and selects a hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The parameters p, q, g, and h are assumed to be publicly known.[5] With the above public parameters and the total number of signatures for the forward-secure scheme T in hand, the signer generates the forward-security chain $R = (r^{(1)}, r^{(2)}, ..., r^{(T)})$, where each $r^{(i)}$ corresponds to i^{th} signature. To start, the signer generates a integer $k^{(1)}$ picked randomly from the multiplicative group $\mathbb{Z}_q^*$. The value of $r^{(1)}$ is then computed from $k^{(1)}$ as:

$$r^{(1)} = g^{k^{(1)}} \; mod \; p. \tag{2}$$

Using the one-way hash function h, the signer continues to construct a chain of $k^{(i)}$'s:

$$k^{(i)} = h(k^{(i-1)}), \tag{3}$$

of length T. For each $k^{(i)}$ the corresponding $r^{(i)}$ is computed as in equation (2) and the forward-security chain R is constructed as an ordered tuple of the $r^{(i)}$'s. Figure 1 illustrates an implementation of the key generation phase.

The function h in equation (3) must be a one-way function so that evaluating $k^{(i-1)}$ from $k^{(i)}$ can be assumed infeasible. Moreover, by the discrete logarithm assumption, computing $k^{(i)}$ using the knowledge of $r^{(i)}$ is infeasible.

```
Algorithm $\mathcal{FKG}(T)$
    $k^{(1)} \xleftarrow{R} \mathbb{Z}_q^*$;
    $r^{(1)} \leftarrow g^{k^{(1)}} \; mod \; p$;
    For $i = 2, ..., T$ do
        $k^{(i)} \leftarrow h(k^{(i-1)})$;
        $r^{(i)} \leftarrow g^{k^{(i)}} \; (mod \; p)$;
        Delete $k^{(i-1)}$;
    EndFor
    $R \leftarrow (r^{(1)}, r^{(2)}, r^{(3)}, ..., r^{(T)})$;
    Return $R$
```

After the forward-security chain R has been generated, the signer uses its first secret key x_1 to sign the chain (using any secure standard signature scheme.) The secret key x_1 is only needed to sign the chain R in the key generation phase and should not be stored in the system to ensure forward-secrecy. Otherwise, an adversary breaking into the system can forge a signature for any R. Note that the only parameter that the signer

[5] This setup is the same as in the Schnorr signature scheme [23]. For different standard signature schemes, setup varies according to the used standard scheme.

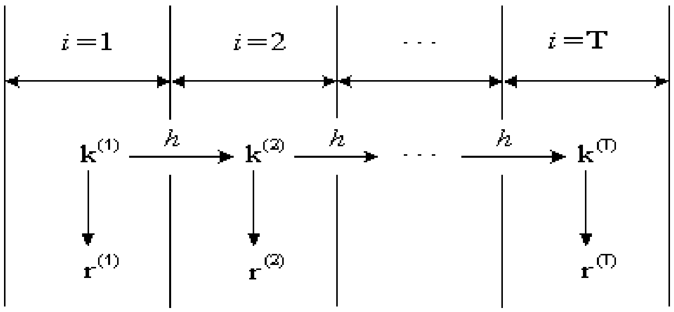


Fig. 1. The forward-security chain generation: Secret key for a given signature is a hash of the secret key for the previous signature

is required to store after the completion of the key generating phase is the value of the random number $k^{(1)}$ and the private key x_2. The chain R is used to provide forward-security and it is not required for signatures generation. Observe that the key generation is performed only once during the lifetime of the FSS, and it is performed off-line.

Signature Generation. To sign the i^{th} message, the signer uses its second secret key x_2 and the value of $k^{(i)}$ to run the signing algorithm $\mathcal{FS}$. The signer calls the base signing algorithm with $k^{(i)}$ passed as a parameter.

$$s = \mathcal{S}(M, x_2, \lambda = k^{(i)}), \tag{4}$$

where $\mathcal{S}$ represents the signing algorithm corresponding to the standard signature scheme used as a building block. The tuple $\sigma = (i, s, r^{(i)})$ comprises the signature on message M.

```
Algorithm $\mathcal{FS}(M, x_2, i, k^{(i)})$
    $s \leftarrow \mathcal{S}(M, x_2, \lambda = k^{(i)})$;
    Return $\sigma = (i, s, r^{(i)})$
```

Note that, instead of generating a random k for every signature as in DLP-based signatures, such as El-Gamal [9] or Schnorr [23], the value $k^{(i)}$ is passed as a parameter in our scheme. Thus, our forward-secure signature generation is more computationally efficient than its base scheme. No previous generic FSS can be more efficient then its base scheme. Further improvement can be made, depending on the resources available for the signer. If computational efficiency is more important than storage, the signer can store R in the system. Storing R in the system will save the signer one modular exponentiation by passing $r^{(i)}$ as a parameter; an operation that requires $O(lk^2)$ time to be performed. This efficiency improvement, however, comes with a cost of $O(Tk)$ bits in storage.

Signature Verification. The verification algorithm $\mathcal{FV}$ is shown below. The verifier uses the signer's public key y_2 to verify the validity of the signature, using the standard verification algorithm $\mathcal{V}$. Then, the verifier runs the standard signature verification to verify the validity of the forward-security chain R using the signer's public key y_1.

```
Algorithm $\mathcal{FV}(M, y_2, \sigma, R, y_1, \sigma_R)$
  If $\mathcal{V}(M, y_2, \sigma) = 1$
    If $\mathcal{V}(R, y_1, \sigma_R) = 1$   and   $r_\sigma^{(i)} = r_R^{(i)}$
      Return 1
    Else
      Return 0
    EndIf
  Else
    Return 0
  EndIf
```

Observe that, without the second check, which is to check the validity of the chain R and if the value $r^{(i)}$ in the signature σ is equal to the authenticated $r^{(i)}$ in R, the verification algorithm is just the verification algorithm of the base scheme. That is, the scheme can be used as a standard scheme and, if needed (e.g., in case of a dispute), R can be used to ensure forward-security.

Key Update. After the (i-1)th signature has been generated, the signer updates the secret key $k^{(i-1)}$ by applying the one-way hash function, to get $k^{(i)}$. As soon as the value of $k^{(i)}$ has been computed, the value of $k^{(i-1)}$ must be deleted to ensure forward-security, as can be seen in algorithm $\mathcal{KU}$ below.

```
Algorithm $\mathcal{KU}(k^{(i-1)})$
  $k^{(i)} \leftarrow h(k^{(i-1)})$;
  Delete $k^{(i-1)}$
  Return $k^{(i)}$
```

The key update operation uses one application of the hash function, an operation that requires $O(l^2)$ in time. No previous FSS [3,1,13,11,5,14,18] can achieve the same performance, except for Anderson's proposal [2] which has been argued to be impractical. Even better, by storing the chain of $k^{(i)}$'s in the system, the key update can be performed in $O(1)$ time with the expense of $O(Tl)$ bits in storage, giving the system designer a choice to trade-off storage and computational efficiencies. Note that if the chain of $k^{(i)}$'s is stored, $k^{(j)}$ must be deleted immediately after the generation of the jth signature to ensure forward security.

Table 1 illustrates our construction of an FSS based on the Schnorr signature [23].

Theorem 1. *If an adversary can break the forward-security of our scheme, she can solve the discrete logarithm problem or invert a one-way function.*

Proof. Compared to the underlying signature scheme, our scheme introduces the precomputation of the $k^{(i)}$'s and the forward-security chain R. Given the security of the underlying scheme, the adversary needs to know the $k^{(i)}$ corresponding to the ith signature to forge a valid signature. If $k^{(i)}$ has already been deleted from the system, the

Table 1. Applying the proposed construction method to design an FSS based on the Schnorr signature scheme [23]. The key generation $\mathcal{FKG}$, signature generation $\mathcal{FS}$, signature verification $\mathcal{FV}$, and the key updating $\mathcal{KU}$ algorithms are summarized.

```
Algorithm $\mathcal{FKG}(T)$
  $k^{(1)} \xleftarrow{R} \mathbb{Z}_q^*$
  $r^{(1)} \leftarrow g^{k^{(1)}}\ mod\ p$;
  For $i = 2, ..., T$ do
    $k^{(i)} \leftarrow h(k^{(i-1)})$;
    $r^{(i)} \leftarrow g^{k^{(i)}}\ (mod\ p)$;
    Delete $k^{(i-1)}$
  EndFor
  $R \leftarrow (r^{(1)}, r^{(2)}, r^{(3)}, ..., r^{(T)})$;
  Return $R$
```

```
Algorithm $\mathcal{FS}(M, x_2, i, k^{(i)})$
  $r^{(i)} \leftarrow g^{k^{(i)}}\ mod\ p$;
  $s \leftarrow h(M, i, r^{(i)})x_2 + k^{(i)}\ mod\ q$;
  Return $\sigma = (i, s, r^{(i)})$
```

```
Algorithm $\mathcal{FV}(M, y_2, \sigma, R, y_1, \sigma_R)$
  If $r^{(i)} \equiv y_2^{h(M,i,r^{(i)})} g^s\ mod\ p$
    If $\mathcal{V}(R, y_1, \sigma_R) = 1$ and $r_\sigma^{(i)} = r_R^{(i)}$
      Return 1
    Else
      Return 0
    EndIf
  Else
    Return 0
  EndIf
```

```
Algorithm $\mathcal{KU}(k^{(i-1)})$
  $k^{(i)} \leftarrow h(k^{(i-1)})$;
  Delete $k^{(i-1)}$
  Return $k^{(i)}$
```

adversary can compute $k^{(i)}$ either from $r^{(i)}$ by solving the DLP or from present $k^{(j)}$, $j > i$, by inverting the one-way hash function. □

3.2 Performance Analysis

In this section we analyze the performance of our scheme.

Key generation. The key generation requires the user to generate a total of T $k^{(i)}$'s, one for each signature. Each $k^{(i)}$ is obtained from $k^{(i-1)}$ by one application of the hash function. Each application of the hash function is performed in $O(l^2)$ time. For each $k^{(i)}$, the corresponding $r^{(i)}$ is computed using one modular exponentiation. Each computation of $r^{(i)}$ is performed in $O(lk^2)$ time. Therefore, the entire key generation operation is performed in $O(T(l^2 + lk^2))$ time.

Signature generation. Our forward-secure signature generation is the signature generation of the standard scheme with $k^{(i)}$ passed as a parameter. If the Schnorr signature scheme is used, the signature generation of our scheme is performed in $O(lk^2+2l^2)$ time.

Table 2. Comparing the asymptotic performances of the generic constructions of Krawczyk [14], MMM [18], and our scheme, when the Schnorr signature scheme [23] is used as a building block

	Krawczyk	MMM	**Our Scheme**
Key update time	$2l^2 + lk^2$	Tk^2l	l^2
Signature time	$3l^2 + lk^2$	$3l^2 + lk^2$	$2l^2 + lk^2$
Verifying time	$2(2lk^2 + l^2)$	$2(2lk^2 + l^2) + (\log l + \log T)l^2$	$2(2lk^2 + l^2)$
Signature size	$\log T + 2l + 4k$	$(\log l + \log T + 2)l + 4k$	$\log T + l + k$
Key gen time	$(5l^2 + 2lk^2)T$	k^2l^2	$(l^2 + lk^2)T + lk^2 + 3l^2$
Secret key size	l	$k + l(\log l + \log T)$	$2l$
Public key size	k	l	$2k$

Signature verification. The verification of our scheme is the same as in the standard scheme with the added verification of the forward-security chain R and the check that $r^{(i)}$ in the signature is equal to $r^{(i)}$ in R. If Schnorr signature scheme is used, the verification operation can be performed in $O(4lk^2 + 2l^2)$. Observe, however, the verification of more than one signature requires only a single verification of the forward-security chain R.

Key update. After the generation of every signature, the key $k^{(i-1)}$ is updated to get $k^{(i)}$ and the old key is deleted. The hashing operation can be performed in $O(l^2)$ time. Although this key update operation outperforms all previous schemes, it can be made even more efficient. If the user can store the chain of $k^{(i)}$'s generated during the key generation phase, the key update can be performed in $O(1)$ time.

Key sizes. Since the signer possesses two pairs of registered keys, the size of the secret key is $2l$ bits, while the size of the public key is $2k$ bits.

Signature size. The only extra parameter in the signature of our scheme, compared to the standard scheme, is the inclusion of the signature's number i. Resulting in an extra $\log i$ bits in the size of the signature. If Schnorr signature is used, the resulting signature size will be $O(l + k + \log i)$ bits.

3.3 Applicability

The properties and performances of all parts of our scheme make it attractive for a variety of applications. Our focus will be electronic checkbooks (e-checkbooks) which was our primary motivation. Just like a regular checkbook, an e-checkbook usually contains multiple checks. Each e-check is meant to be signed by the checkbook holder and verified by the bank issuing the e-checkbook.

The signature number in our scheme is mapped to the check serial number in the e-checkbook application. When a scheme with inefficient key update algorithm is used and the user's e-checkbook is stolen, an unauthorized user with access to the e-checkbook can forge a signature on the last signed e-check that is indistinguishable from the authorized signature, making the last check signed by the authorized user repudiable. Therefore, a scheme with very efficient key update algorithm, like our scheme,

Table 3. Comparing the performances of the generic constructions of Krawczyk [14], MMM [18], and our scheme, when the Schnorr signature scheme [23] is used as a building block and for $k = 1024$, $l = 160$, and $T = 1000$. $(*)$ this operation is done only once for the lifetime of the forward secure scheme and is performed offline.

	Krawczyk		MMM		**Our Scheme**
	Absolute	Relative	Absolute	Relative	Absolute
Key update time	1.7×10^8	6800	1.7×10^{11}	6800000	2.5×10^4
Signature time	1.7×10^8	1	1.7×10^8	1	1.7×10^8
Verifying time	6.7×10^8	1	6.7×10^8	1	6.7×10^8
Signature size	4.4×10^3	3.67	7.2×10^3	6	1.2×10^3
Key gen time	3.4×10^{11}	2	2.7×10^{10}	$-6.3(*)$	1.7×10^{11}

is of most importance to the application of e-checkbook. Furthermore, since signatures are meant to be verified by the bank, the forward-security chain R can be stored in the bank's database; thus, no need to worry about the exchange of the chain R. Moreover, the verification of the chain R needs to be performed only once.

3.4 Comparison to Other Schemes

In this section, we compare the performance of our scheme with the Krawczyk [14] and Malkin *et al.* [18] when the Schnorr signature scheme is used for construction. Table 2 shows the asymptotic performances of the three schemes in the worst case scenario (when $t_i = T$ for the schemes in [14,18], and $i = T$ in our scheme). If $k \approx O(l^3)$, as suggested in [18], the three schemes are asymptotically comparable, except for the key update operation where the performances are $O(l^7)$, $O(Tl^7)$, and $O(l^2)$ for the Krawczyk, MMM, and our scheme respectively. Thus, supporting our claim that our key update algorithm is orders of magnitude more efficient.

Table 3 compares the performances when $k = 1024$, $l = 160$, and $T = 1000$. The values of k and l are the typical values in Schnorr signature scheme. The columns corresponding to the Krawczyk and the MMM schemes are split into two halves. The first half shows the performance of the corresponding scheme where the second half shows the normalized performances when our scheme is used as a reference point. Positive values represent the relative advantage of our scheme, while negative values represent the advantage of their schemes over ours. Since storage is cheap in today's technology, the non-secret storage of the T certificates in the Krawczyk scheme and the chain R in our scheme are ignored.

In the next section, we show how to construct forwrad-secure proxy signature schemes.

4 Forward-Secure Proxy Signature Scheme

4.1 Background

In the previous section, we showed how to construct an FSS with the requirement that the signer possesses two pairs of registered keys. In this section, we show how the same idea can be extended to construct a forward-secure proxy signature scheme (FSPS) from any secure proxy signature scheme, without the requirement that the signer possesses two pairs of registered keys. This requirement is avoided because, in the proxy signature setup, there are two legitimate users, namely, the proxy designator and the proxy signer, and each one of them is assumed to possess a pair of registered private and public keys, (x_a, y_a) and (x_b, y_b), respectively. For historical reasons the proxy designator and signer are called Alice and Bob respectively. In proxy signature schemes, Alice delegates her signing capability to Bob. The idea of digital proxy signatures was first introduced by Mambo *et.al.* [20]. Many of the proposed proxy signature schemes appeared in the literature are based on the following concept: Alice has a pair of keys (x_a, y_a), where x_a and y_a represent the secret and public keys, respectively. To delegate her signing power to the Bob, Alice generates a warrant describing Bob's authorities to sign messages on her behalf. The warrant is then signed by Alice (using a standard signature scheme) and sent to Bob. After checking the validity of the signature, Bob combines Alice's signature with his secret key x_b to generate a *proxy* signing key x_p. Bob uses the *proxy* signing key x_p to sign messages on behalf of Alice using a standard signature scheme. To validate a proxy signature, the verifier computes the public key y_p corresponding to the proxy secret key x_p (usually a function of Alice's and Bob's public keys; that is, $y_p = f(y_a, y_b)$) and use the corresponding standard signature verification algorithm to verify the signature.

Many proxy signature schemes have appeared in the literature [21,22,12,15,4]. As noted by Wang [24], little effort has been made towards the design of FSPS. In most of the existing proxy signature schemes, Bob generates a proxy key x_p, and the same key will be used for proxy signing throughout the signature delegation period, without any modifications. Hence, exposure of the proxy key x_p will enable a forger to generate proxy signatures that are indistinguishable from the legitimate signatures. We propose, to the best of our knowledge, the first design of an FSPS.

4.2 Definitions

It is important to distinguish between two keys, namely, Bob's secret key x_b and the key generated by Bob to generate signatures on Alice's behalf x_p. Bob's key x_b is the one distributed by the CA and does not change. The other key x_p is generated by Bob and can be updated whenever Bob wishes.

Before describing the construction, we define two different notions of forward-security for proxy signatures; weak forward-secure proxy signature (WFSPS) schemes and strong forward-secure proxy signature (SFSPS) schemes.

Definition 2 (Weak Forward Secure Proxy Signatures). *A proxy signature scheme is said to be weak forward-secure if it is resilient to the exposure of the proxy key x_p.*

Definition 3 (Strong Forward Secure Proxy Signatures). *A proxy signature scheme is said to be strong forward-secure if it is resilient to the exposure of the proxy key x_p and the private key of the proxy signer x_b.*

The lack of appropriate definitions for forward security in proxy signatures led Malkin *et al.* to conclude that proxy signatures are by design forward secure [19]. Given our

Table 4. A forward-secure proxy signature scheme constructed by applying the proposed construction method to the proxy signature scheme in [12]. Assuming the forward-security chain has been generated successfully, the proxy key generation $\mathcal{PKG}$, proxy signature generation $\mathcal{PS}$, proxy signature verification $\mathcal{PV}$, and the proxy key updating $\mathcal{PKU}$ algorithms are summarized. The subscript a indicates parameters generated by the proxy designator and w represents the warrant describing the authority given to the proxy, as in the original scheme in [12]. x_b represents the registered secret key of the proxy signer, while x_p represents the key generated to sign messages.

```
Algorithm $\mathcal{PKG}(w, \sigma_a, k^{(i)}, i, x_b)$
  $r^{(i)} \leftarrow g^{k^{(i)}} \ mod\ p$
  $x_p^{(i)} \leftarrow h(i, r^{(i)})x_b + h(w, r_a)x_a + k_a \ mod\ q$
  Return $x_p^{(i)}$
```

```
Algorithm $\mathcal{PS}(M, x_p, k^{(i)})$
  $s_p \leftarrow h(M)x_p^{(i)} + k^{(i)} \ mod\ q$;
  Return $\sigma_M = (i, s_p, r^{(i)}, w, r_a)$;
```

```
Algorithm $\mathcal{PV}(y_a, y_b, M, \sigma_M, R, \sigma_R)$
  $y_p^{(i)} \leftarrow y_b^{h(i,r^{(i)})} y_a^{h(w,r_a)} r_a^{-1} \ mod\ p$;
  If $r^{(i)} \equiv (y_p^{(i)})^{h(M)} g^{s_p} \ mod\ p$
    If $\mathcal{V}(R, y_a, \sigma_R) = 1$ and $r_\sigma^{(i)} = r_R^{(i)}$
      Return 1
    Else
      Return 0
    EndIf
  Else
    Return 0
  EndIf
```

```
Algorithm $\mathcal{PKU}(k^{(i-1)})$
  $k^{(i)} \leftarrow h(k^{(i-1)})$;
  Delete $k^{(i-1)}$
  Return $k^{(i)}$
```

definitions above, under the condition that Bob generates a new x_p for every signature, proxy signatures are only weak forward-secure.

4.3 Our Proposed Strong Forward-Secure Proxy Signature Scheme

The idea here is the same idea for constructing regular FSS. For lack of space, we omit describing the details of the construction and outline the basic concept. The major difference between the standard proxy signature scheme and the forward-secure version is in the proxy and key initialization stage, which we describe below.

Proxy and key Initialization. The proxy key generation is an interactive protocol. To start the protocol, Alice decides the number of signatures T for the forward-secure proxy scheme. Upon receiving T, Bob runs the same algorithm $\mathcal{FKG}$ used in the construction of non-proxy forward-secure signature schemes to generate the forward-security chain R of length T. The forward-security chain R is then sent to Alice who signs R with her private key x_a and publishes the signed R.

For the i^{th} message M to be signed, the pair $(k^{(i)}, r^{(i)})$ is used by Bob to generate forward-secure proxy signatures with the private key x_p. The fact that R is signed by Alice ensures forward-security, even if Bob's system has been broken into.

Table 4 illustrates our construction of a forward-secure proxy signature scheme based on the provable secure scheme of Kim *et al.* [12]. The constructed scheme in Table 4 assumes that the proxy and key initialization has been performed successfully and the forward-security chain *R* is available for verifiers.

The security analysis of the forward-secure proxy scheme is similar to the non-proxy one. Assuming the standard proxy scheme is secure[6], provided that the forward-security chain *R* has been generated successfully, the forward-security is granted by the hardness of the discrete logarithm problem and the existence of one-way functions.

5 Conclusion

In this paper, we studied the two major design approaches of forward secure signature schemes, i.e. number-theoretic and generic construction based approaches. The generic construction schemes can be based on any underlying signature scheme, hence, offering more flexibility in terms of meeting a wide spectrum application requirements. We proposed a generic construction method to obtain a forward-secure signature scheme that is very efficient in parameter size and computation times. A quantitative comparison with related schemes showed that the proposed scheme is more computationally efficient in terms of signature verification, signature size, and key updating time.

Further, we identified that in the literature there was no explicit design of forward secure proxy signatures. Therefore, we first defined two notions of forward security in the context of proxy signatures, i.e. weak forward-secure and strong forward-secure proxy signatures. We then showed how the proposed generic construction method can be easily extended to any proxy signature scheme to obtain strong forward secure proxy signatures.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments.

References

1. Abdalla, M., Reyzin, L.: A new forward-secure digital signature scheme. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 116–129. Springer, Heidelberg (2000)
2. Anderson, R.: Two remarks on public key cryptology. In: Invited Lecture, ACM-CCS 1997 (1997)
3. Bellare, M., Miner, S.: A forward-secure digital signature scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
4. Boldyreva, A., Palacio, A., Warinschi, B.: Secure Proxy Signature Schemes for Delegation of Signing Rights (2003), `http://eprint.iacr.org/2003/096`
5. Boyen, X., Shacham, H., Shen, E., Waters, B.: Forward-secure signatures with untrusted update. In: Proceedings of the 13th ACM conference on Computer and communications security, pp. 191–200 (2006)

[6] Kim *et al.* [12] is an example of a provable secure proxy signature scheme.

6. Buchmann, J.: Introduction to Cryptography. Springer, Heidelberg (2004)
7. Cronin, E., Jamin, S., Malkin, T., McDaniel, P.: On the performance, feasibility, and use of forward-secure signatures. Proceedings of the 10th ACM conference on Computer and communications security, 131–144 (2003)
8. Diffie, W., Oorschot, P.C., Wiener, M.: Authentication and authenticated key exchanges. Designs, Codes and Cryptography 2(2), 107–125 (1992)
9. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
10. Günther, C.: An identity-based key-exchange protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
11. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 332–354. Springer, Heidelberg (2001)
12. Kim, S., Park, S., Won, D.: Proxy signatures, Revisited. In: Proceedings of the First International Conference on Information and Communication Security, pp. 223–232 (1997)
13. Kozlov, A., Reyzin, L.: Forward-Secure Signatures with Fast Key Update. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 241–256. Springer, Heidelberg (2003) Revised Papers
14. Krawczyk, H.: Simple forward-secure signatures from any signature scheme. In: Proceedings of the 7th ACM conference on Computer and communications security, pp. 108–115 (2000)
15. Lee, B., Kim, H., Kim, K.: Strong proxy signature and its applications. In: Proc. of SCIS - Symposium on Cryptology and Information Security, pp. 603–608 (2001)
16. Libert, B., Quisquater, J., Yung, M.: Forward-secure signatures in untrusted update environments: efficient and generic constructions. In: Proceedings of the 14th ACM conference on Computer and communications security, pp. 266–275 (2007)
17. Ma, D., Tsudik, G.: Extended Abstract: Forward-Secure Sequential Aggregate Authentication. In: IEEE Symposium on Security and Privacy, 2007. SP 2007, pp. 86–91 (2007)
18. Malkin, T., Micciancio, D., Miner, S.: Composition and Efficiency Tradeoffs for Forward-Secure Digital Signatures. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, Springer, Heidelberg (2002)
19. Malkin, T., Obana, S., Yung, M.: The Hierarchy of Key Evolving Signatures and a Characterization of Proxy Signatures. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 306–322. Springer, Heidelberg (2004)
20. Mambo, M., Usuda, K., Okamoto, E.: Proxy Signatures: Delegation of the Power to Sign Messages. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 79(9), 1338–1354 (1996)
21. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: Proceedings of the 3rd ACM conference on Computer and communications security, pp. 48–57 (1996)
22. Petersen, H., Horster, P.: Self-certified keys-concepts and applications. Proc. Communications and Multimedia Security 97, 102–116 (1997)
23. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
24. Wang, G.: Designated-verifier proxy signature schemes. In: Security And Privacy in the Age of Ubiquitous Computing: IFIP TC11 20th International Information Security Conference, Chiba, Japan, May 30-June 1, 2005 (2005)

Fault Attacks on Public Key Elements: Application to DLP-Based Schemes

Chong Hee Kim[⋆], Philippe Bulens[⋆⋆], Christophe Petit[⋆⋆⋆], and Jean-Jacques Quisquater

UCL Crypto Group, Université Catholique de Louvain,
3 Place du Levant, 1348 Louvain-la-Neuve, Belgium
Tel.: +32 (0) 10 47 81 41; Fax: +32 (0) 10 47 25 98
{first.lastname}@uclouvain.be

Abstract. Many cryptosystems suffer from fault attacks when implemented in physical devices such as smart cards. Fault attacks on secret key elements have successfully targeted many protocols relying on the Elliptic Curve Discrete Logarithm Problem (ECDLP), the Integer Factorization Problem (IFP) or the Discrete Logarithm Problem (DLP). More recently, faults attacks have also been designed against the *public* key elements of ECDLP and IFP-based schemes.

In this paper, we present the first fault attacks on the public key elements of DSA and ElGamal, two DLP-based signature schemes. Our attacks fully recover a 160-bit DSA secret key and a 1024-bit ElGamal secret key with $\sim 4\cdot 10^7$ and $\sim 3\cdot 10^6$ faulty signatures respectively. Such figures might suggest that DLP-based schemes are less prone to fault attacks than ECDLP- and IFP-based schemes. However, the integrity of public keys should always be checked in order to thwart such attacks since improvements may reduce the required amount of faulty signatures in the near future.

Keywords: Smart cards, side channel, fault injection, faults attacks, ElGamal, DSA.

1 Introduction

From a classical point of view, cryptanalysis is an abstract mathematical notion. However in practice, algorithms have to be implemented on real physical devices that are exposed to side-channel attacks like timing attacks [12,14,27,28,30], power attacks [11,20,22,26], electromagnetic attacks [1,16,25] as well as fault attacks [3,9,17].

After the discovery of fault attacks (1970's) [21], various mechanisms for fault production and propagation have been discovered and researched. Some of the most popular fault injection techniques include variations in supply voltage, clock frequency, temperature or the use of white light, X-ray or ion beams [3].

[⋆] Supported by Walloon Region, Belgium / E.USER project.
[⋆⋆] Supported by Walloon Region, Belgium / First Europe Program.
[⋆⋆⋆] Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

It is not sufficient to just induce a fault in the cryptographic device during a calculation involving a secret. Depending on the algorithm, it is also crucial to generate the *right* fault at the *right* time and on the *right* place within the chip. The generated fault must be exploitable, meaning that the attacker has to be able to extract some information about the secret from the erroneous result of the algorithm. The first attack [9] that used a fault to derive secret information from a cryptosystem targeted a Chinese Remainder Theorem (CRT) based RSA implementation. After that, DES [5], RSA and ElGamal [2], LUC and Demytko [19], ECC [6], AES [7], and DSA [24] have also been compromised by fault attacks.

All these attacks targeted secret keys and computations involving secret keys. Recently, several papers have considered fault attacks on public key elements, demonstrating the necessity to protect public keys against fault attacks at least for ECDLP (Elliptic Curve Discrete Logarithm Problem) [4] and IFP (Integer Factorization Problem) based algorithms [10,23,29]. Curiously and to the best of our knowledge, no fault attack on public key elements has been proposed so far for any DLP (Discrete Logarithm Problem)-based schemes.

In this paper, we present the first fault attacks on the public key elements of two DLP-based algorithms, namely the ElGamal and DSA signature schemes. We also estimate the complexity of our attacks (the number of faults required) both by a theoretical analysis and software simulations. Although it turns out that our attacks require a large number of faults, they highlight the necessity to check the integrity of both public and private keys elements for DLP-based schemes.

Fault Attack Model: In our model, an attacker tries to corrupt not the private but **the public keys** by faults. Whereas some attacks [2,24] require that only a single bit is flipped or a particular byte is set to zero, our model only assumes that the attacker is able to enforce random register faults. While such single or particular bit flips are rather hard to achieve in practice, changing a word or many words in an undetermined way is the most simple fault to induce. It can simply be obtained by inducing a fault on address decoders for example, when parameters stored in EEPROM or Flash are transferred to RAM. This results in a random transient change in the public key.

Outline of the Paper: The rest of this paper is organized as follows. Section 2 describes previous fault attacks on ECDLP and IFP schemes with a corruption of public keys. In Sections 3 and 4 we present our new attacks on DSA and ElGamal signature scheme and describe their complexity by a theoretical analysis and software simulations. Section 5 contains some open discussions to improve the attacks, and Section 6 concludes the paper.

2 Related Work

Previous attacks on public key elements of cryptographic schemes were performed on ECDLP [4,13] or IFP [29,23,10] cryptosystems.

2.1 Previous Attacks on ECDLP

Here, the attacker tries to shift the computation from a given secure elliptic curve E (in Weierstrass parameterization) to an insecure curve E^*. Consider a smart card that has to compute dP, for d a scalar and P a point on the curve

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

If a fault is induced on P, it is changed into a point P^* on a curve

$$E^* : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + {a_6}^*.$$

The attack exploits the fact that the parameter a_6 is not used in the usual point addition formula on Weierstrass elliptic curves. As a consequence, the whole computation is performed on the curve E^*, and dP^* is obtained. If now E^* is an insecure curve (typically because $\text{ord}(E^*)$ has a small factor), the discrete logarithm problem can be solved on it, which gives information about d. Faults occurring during the computation of dP are also exploitable [4]. The attack here assumes that only a few error bits are inserted in order to be successful. This hypothesis has been relaxed in [13] where a random and unknown fault either in the base point P, in the base field F_p or F_{2^q} underlying the curve, or in the curve's parameters is turned into breaking the scheme.

2.2 Previous Attacks on IFP

Seifert's RSA Attack [29]: The attacker tries to corrupt RSA public key N into N^* during RSA signature verification by faults, where N^* is prime. Then he can simply compute the private exponent d^*, as $e^{-1} \bmod (N^* - 1)$, assuming that e is relatively prime to $(N^* - 1)$. In the off-line part, he can choose bits of N to modify the creation of N^*. In the on-line part, he repeatedly queries the device with a specially constructed message-signature pair and causes data faults until this particular N^* is used as the modulus in the signature verification algorithm.

Seifert proved that there exists an algorithm that will be successful with probability at least $\mathcal{O}(1/k)$ in getting fraudulent programs S^* authenticated and therefore executed on a machine relying on RSA-authentication of programs. The running time of the algorithm is $log^{\mathcal{O}(1)}(N)$, where N and e are a k-bit RSA public keys.

Generalization of Seifert's RSA Attack [23]: Muir generalized Seifert's attack to moduli N^* not necessarily primes but still easy to factorize.

He simplified the analysis of Seifert's attack. He also showed an experimental result for the off-line stage of RSA-Attack. His analysis and computational trials show that if an adversary is able to cause random faults in *only 4 bits* of a 1024-bit RSA modulus stored in a device, then there is more than 50% chance that the modified modulus N^* has good property to attack. For Seifert's original attack, it required 8 bits.

Brier et al.'s RSA Attack [10]: The attacker tries to corrupt RSA public key N into N^* similar to Seifert's attack, but here he tries to attack signature generation process, $s = m^d \bmod N$. Moreover, at the end of the attack the attacker has the secret key d.

The basic idea is based on the fact that, for small moduli - for example, from 15 to 20 digits -, discrete logarithms are efficiently computed by square root methods such as *baby-step giant-step* or *Pollard's rho*. In particular if $p^a|N^*$ and r is a small number dividing the multiplicative order of m modulo p^a, then from $s^* = m^d \bmod N^*$, $d \bmod r$ can be recovered. Gathering some fault couples (m_i, s_i) corresponding to unknown moduli ${N_i}^* \neq N$, the attacker retrieves the private exponent d off-line by progressively determining $d \bmod r_j$ for some small prime powers r_j. Once the product $R = \prod_k r_j$ exceeds the modulus N (and so unknown $\varphi(N)$), d is recovered with the Chinese Remainder Theorem.

Even when the adversary does not have any information on ${N_i}^*$, he can recover a 1024-bit secret d with about 60000 faults, and he needs only 28 faults once he knows the faulty ${N_i}^*$ values.

3 A New Fault Attack on DSA

3.1 The Digital Signature Algorithm

The system parameters for DSA [18] are $\{p, q, h, g\}$, where p is prime (at least 512 bits), q is a 160-bit prime dividing $(p-1)$, h is a hash function and $g \in \mathbb{Z}_p^*$ has order q. The private key is an integer $x \in \mathbb{Z}_q^*$ and the public key is $y = g^x \bmod p$.

Signature: To sign a message m, the signer picks a random $k < q$ and computes:

$$u \leftarrow (g^k \bmod p) \bmod q \qquad \text{and} \qquad v \leftarrow \frac{h(m) + xu}{k} \bmod q.$$

The signature of m is the pair (u, v).

Verification: To check (u, v) the verifier ascertains that:

$$u = (g^{wh(m)} y^{wu} \bmod p) \bmod q, \qquad \text{where } w = v^{-1} \bmod q.$$

3.2 A New Attack by a Fault Induction on *p* and *q*

Let us suppose that an attacker succeeded in invoking transient faults before starting signature generation and changing the values of p, q into p^*, q^*, respectively. Assuming the generated k satisfies $\gcd(k, q^*) = 1$, he gets:

$$u^* \leftarrow \left(g^k \bmod p^*\right) \bmod q^* \qquad \text{and} \qquad v^* \leftarrow \frac{h(m) + xu^*}{k} \bmod q^*.$$

Consider any integer t dividing p^* and q^*, such that $\varphi(t)$ divides q^* and g is not a root of zero modulo t. Then, we have

$$(u^*)^{v^*} \equiv g^{h(m)+xu^*} \bmod t. \tag{1}$$

Consequently,

$$\frac{(u^*)^{v^*}}{g^{h(m)}} \equiv (g^{u^*})^x \bmod t. \tag{2}$$

Denote $DL(\alpha, \beta, t)$ the discrete logarithm of $\beta \bmod t$ with respect to α. For any r_j dividing the multiplicative order of (g^{u^*}) modulo t we have

$$x \equiv DL(\alpha, \beta, t) \bmod r_j, \tag{3}$$

where $\beta = \frac{(u^*)^{v^*}}{g^{h(m)}} \bmod t$ and $\alpha = g^{u^*} \bmod t$ can be computed from the faulty signature.

Assuming the attacker knows the value of the faulty (p^*, q^*) induced, this suggests that an algorithm can recover the secret key x, by successively recovering x modulo r_j with various faulty signatures until the lowest common multiple of those r_j is large enough to recover x using the Chinese Remainder Theorem.

Toy Example. Take $p = 124540019, q = 17389, g = 10083255$. Let the private key be $x = 12496$, so $y = g^x \bmod p$. The message to be signed verifies $h(m) = 5246$, and $k = 9557$ is chosen. The signature is the pair $(u = 34, v = 13049)$. Now suppose that the attacker succeeded in making faults and changing the value of p into p^* and q into q^* before generating the signature. Then, we find x with the values shown in Table 1 .

Table 1. Toy example of the attack on DSA

p^*	q^*	k	u^*	v^*	t	α	β	r_j	$x \bmod r_j$	$\mathrm{lcm}(\{r_j\})$
124539997	17030	41	13384	9140	131	33	28	65	16	65
124539973	17110	6171	12501	422	59	47	7	58	26	3770
124539983	16536	9961	11694	1214	53	20	36	26	16	3770
124539989	17296	491	6434	202	47	24	18	23	7	86710

3.3 Analysis of the Attack

We now provide an estimation of the number of faults needed to recover the whole secret x. Our analysis will be completed in four steps. We first estimate the probability P_t that for a given t, the condition

$$C_t := (t \mid q^*) \wedge (\varphi(t) \mid q^*) \wedge (t \mid p^*) \wedge (\gcd(k, q^*) = 1) \tag{4}$$

is satisfied. Then, we approximate by 1 the probability $P_{r|t}$ that when this condition is satisfied for t, the secret x is recovered modulo r, where r is a prime power dividing $\phi(t)$. From these two probabilities we derive a global estimate for the probabilities $P[r_j]$ to recover x modulo r_j for various r_j, and we finally estimate the number of faults required from these probabilities. Each step in the analysis induces an error by some small factor in the estimations, so our final estimations only give an upper bound on the number of faults required, precise up to a small factor.

We first estimate $P_t = \Pr[C_t]$ where the probability is on random independent choices of k, p^* and q^*. Let write l_t for $\operatorname{lcm}(t, \varphi(t))$ and decompose q^* as $q^* = q_1^* l_t + q_2^*$. Then

$$P_t = \Pr[q_2^* = 0] \Pr[t \mid p^*] \Pr[(\gcd(k, q_1^*) = 1) \wedge (\gcd(k, l_t) = 1)].$$

Clearly, $\Pr[q_2^* = 0] = \frac{1}{l_t}$ and $\Pr[t \mid p^*] = \frac{1}{t}$. Neglecting correlations between the conditions on the gcds, we have

$$\Pr[(\gcd(k, q_1^*) = 1) \wedge (\gcd(k, l_t) = 1)] \approx \Pr[\gcd(k, q_1^*) = 1] \Pr[\gcd(k, l_t) = 1].$$

As $\Pr[\gcd(k, l_t) = 1] = \frac{\varphi(l_t)}{l_t}$ and $\Pr[\gcd(k, q_1^*) = 1] \approx \lim_{Q \to \infty} \sum_{q=0}^{Q} \frac{\varphi(q)}{q} = \frac{6}{\pi^2} \approx 0.6$ we finally get

$$P_t \approx 0.6 \frac{\varphi(l_t)}{t l_t^2} = 0.6 \frac{\varphi(\operatorname{lcm}(t, \varphi(t)))}{t (\operatorname{lcm}(t, \varphi(t)))^2}. \tag{5}$$

The conditions on the gcds are actually dependent with a positive correlation: clearly if $\gcd(k, q_1^*) = 1$ then for example k is more likely to be a prime and then $\gcd(k, l_t) = 1$ will be satisfied also. However, as

$$\frac{\Pr[(\gcd(k, q_1^*) = 1) \wedge (\gcd(k, l_t) = 1)]}{\Pr[\gcd(k, q_1^*) = 1] \Pr[\gcd(k, l_t) = 1]} \leq \frac{\Pr[\gcd(k, q_1^*) = 1]}{\Pr[\gcd(k, q_1^*) = 1] \Pr[\gcd(k, l_t) = 1]} = \frac{l_t}{\varphi(l_t)},$$

for most t values the actual probability is only a small factor larger than our approximation.

Now suppose a fault is performed such that Condition (4) is satisfied. For each r dividing $\varphi(t)$, we evaluate the probability $P_{r|t}$ that the attacker recovers $x \bmod r$ from Equation (2), *i.e.* by computing a discrete logarithm modulo t.

As p^* is uniformly random, the remainder of $u^* = g^k \bmod p^*$ modulo $\varphi(t)$ is nearly uniformly distributed in $\mathbb{Z}_{\varphi(t)}$. We deduce that α is nearly uniformly distributed in the subgroup of $\mathbb{Z}_t^*$ generated by g. Write $r_{g,t} = \prod_{i=1}^{I} p_i^{e_i}$ and $r_{\alpha,t}$ for the multiplicative orders of g and α modulo t, where p_i are distinct primes. Then, for any divisor $r = \prod_{i=1}^{I} p_i^{e'_i}$ of $r_{g,t}$, the probability that $r_{\alpha,t} = r$ is $\frac{\varphi(r)}{r_{g,t}} = \prod p_i^{-e_i} \prod_{e'_i \neq 0} (p_i - 1) p_i^{e'_i - 1}$, and the probability that r divides $r_{\alpha,t}$ is $P_{r|t} = \prod_{e'_i \neq 0} p_i^{-(e_i - e'_i + 1)} (p_i^{e_i - e'_i + 1} - 1)$.

In particular if r is a power of a prime, *i.e.* $r = p_r^{e'_r}$, then $\frac{1}{2} \leq P_{r|t} = p_r^{-(e_r - e'_r + 1)} (p_r^{e_r - e'_r + 1} - 1) \leq 1$. We approximate this probability by 1 for all r, which is a good approximation for all but very small r. The approximation means the following: we suppose that once a fault that satisfies Condition (4) for some t is induced, the attacker recovers x modulo r for any r dividing $\varphi(t)$.

For any set of integers T, write P_T for the probability that all $t \in T$ verify Condition (4), and no other integer. We now evaluate $P[r]$, the probability to recover the secret modulo a prime power r if the attack is performed once.

According to our previous arguments, we have

$$P[r] \approx \sum_{\substack{T=\{t_i\} \\ \text{s.t. } \exists t \in T \text{ s.t. } r|\varphi(t)}} P_T .$$

We finally approximate

$$P[r] \approx \sum_{\substack{T=\{t_i\} \\ \text{s.t. } t_r \in T}} P_T = P_{t_r} \qquad \text{where } t_r = \arg \max_{t \text{ s.t. } r|\varphi(t)} P_t . \tag{6}$$

This approximation underestimates the actual probability because all the sets in the original sum that do not contain t_r are neglected. However, according to the estimation (5), P_t tend to decrease very fast with t, so for the sets T that do not contain t_r the probability P_T is much smaller than P_{t_r}, and the actual probability $P[r]$ is only a small factor larger than our estimation.

We now estimate the number of bits of the secret key that can be recovered for a given amount of faults. In order to recover x by means of the Chinese Remainders Theorem, an adversary should know a list of values $x \bmod r_j$ such that $\text{lcm}(\{r_j\}) > x$. On the other hand, if N faults are performed, it is likely that x will be recovered modulo r_j for all r_j such that $P[r_j] > N^{-1}$, so the quantity of information that can be recovered with N faults is

$$\log_2(\text{lcm}(\{r_j | P[r_j] \geq N^{-1}\})). \tag{7}$$

This conclude our analysis.

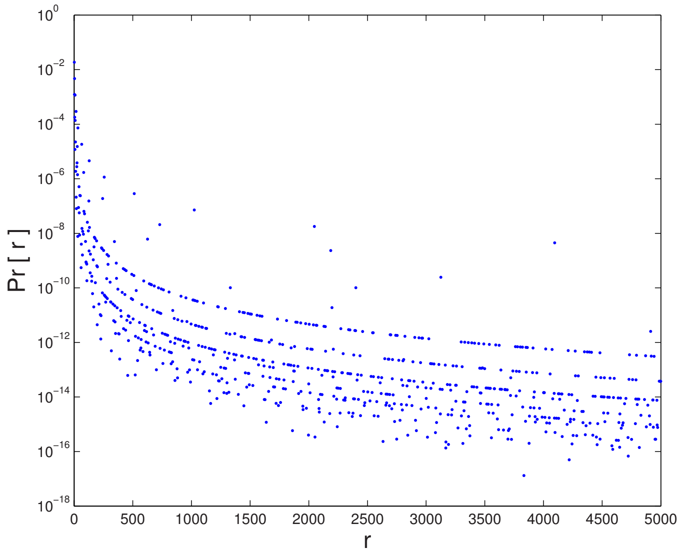


Fig. 1. Approximation (6) for all r power of prime smaller than 5000

In Figure 1, we display the probabilities $P[r_j]$ evaluated from Equations (5) and (6) for every r_j power of prime up to 5000. Using this computation, we further estimate for any $N_s = 10^s$ with $4 \leq s \leq 10$, the number of bits of the secret key that can be recovered. The results are presented in Figure 2. The last row in the table corresponds to 160-bits security and predicts that about $2.14 \cdot 10^8$ million faults are needed to recover the secret. The table and the figure additionally show that, at least for this range of parameters, the attack complexity is between cubic and biquadratic in the size of the secret.

3.4 Experimental Results

In order to check the soundness of our theoretical analysis, we simulated the attack in software (with parameters p a 1024-bit prime, q a 160-bit prime and g about the size of p). The results are shown in Table 2, where each column gives the amount of faulty couples (p^*, q^*) used to retrieve m bits of the private key x. As can be seen, the required amount of faults in our simulations is 3 to 10 times smaller than the $2.14 \cdot 10^8$ theoretical bound with a mean of $4.16 \cdot 10^7$ faults.

$N_s = 10^s$	$\lvert x\rvert \approx \log_2(\text{lcm}(\{r_j\}))$
10^4	9
10^5	19
10^6	38
10^7	61
10^8	122
10^9	253
10^{10}	472
$2.14 \cdot 10^8$	160

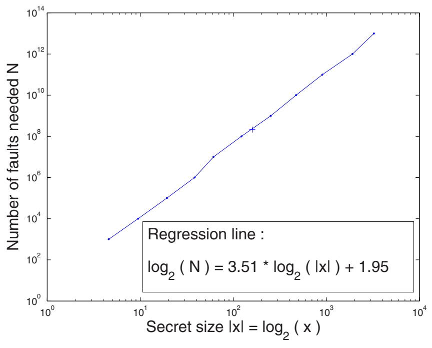


Fig. 2. Following the theoretical analysis of our attack on DSA, the number of faults needed N is between cubic and biquadratic with respect to the secret size $|x|$

Table 2. Seven experimental software results for our attack on DSA. In each column, we display the number of faults needed to recover m bits of the secret for each of the experiments.

Exp. n°	$m = 32$	$m = 64$	$m = 96$	$m = 128$	$m = 160$
1	463986	5676441	8549083	26476140	38621903
2	811215	8898520	14945495	22174790	34861119
3	284706	3336328	5577267	11579152	20960118
4	798230	6438425	21496166	31049856	61711260
5	282811	8363054	14303797	26594960	39066576
6	721742	8762969	20601681	38398403	73933924
7	453623	3174393	10220088	17145832	22623221

4 A New Fault Attack on ElGamal Signature Algorithm

4.1 ElGamal Signature Algorithm

In the ElGamal signature scheme [15], to generate a private and public key pair, we first choose a prime p, and two random numbers, g and x, such that both g and x are less than p. The private key is x and the public key is $(y = g^x \bmod p, g, p)$.

Signature. To generate a signature on a message m, the signer first picks a random k such that k is relatively prime to $(p-1)$. He then computes

$$u \equiv g^k \bmod p \qquad \text{and} \qquad v \equiv \frac{m - xu}{k} \bmod (p-1).$$

Verification. The signature is the pair (u, v) and is verified by checking that

$$y^u u^v \equiv g^m \bmod p.$$

4.2 A New Attack by Fault Injection on p^*

Suppose that there are faults before the signature generation and the modulus p is changed into p^*. Then, if $\gcd(k, p^* - 1) = 1$, we have

$$u^* \equiv g^k \bmod p^* \qquad \text{and} \qquad v^* \equiv \frac{m - xu^*}{k} \bmod (p^* - 1).$$

Let t divide p^* and $\varphi(t)$ divide $(p^* - 1)$. Then, we have

$$(u^*)^{v^*} \equiv g^{(m - xu^*)} \bmod t. \tag{8}$$

Consequently, if g is not a root of 0 modulo t,

$$\frac{(u^*)^{v^*}}{g^m} \equiv (g^{-u^*})^x \bmod t. \tag{9}$$

For any r_j dividing the multiplicative order of $(g^{-u^*}) \bmod t$, we have

$$x \equiv \mathrm{DL}(\alpha, \beta, t) \bmod r_j, \tag{10}$$

where $\beta = \frac{(u^*)^{v^*}}{g^m} \bmod t$ and $\alpha = g^{-u^*} \bmod t$ can be computed from the faulty signature.

Assuming that the attacker knows the value of the faulty p^* induced, equation (10) makes it possible to recover some information about x. By retrieving $x \bmod r_j$ for sufficiently many r_j satisfying $\mathrm{lcm}(\{r_j\}) > x$, the whole secret can be recovered by use of Chinese Remainder Theorem.

4.3 Analysis and Experimental Results

The analysis of ElGamal case is similar to (and actually much easier than) DSA case, so we omit its detail. In the table of Figure 3, we computed $\log_2(\mathrm{lcm}(\{r_j\}))$

$N_f \approx b_s^{-1}$	$\lvert x\rvert \approx \log_2(\mathrm{lcm}(\{r_j\}))$
10^4	63
10^5	218
10^6	641
10^7	1788
10^8	5319
10^9	16019
10^{10}	47468
$2.89 \cdot 10^6$	1024

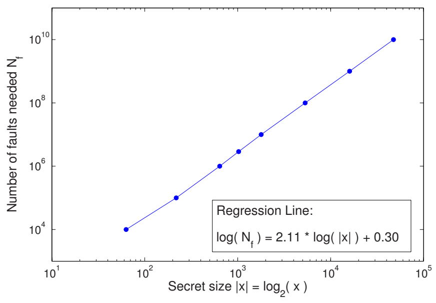


Fig. 3. Following the theoretical analysis of our attack on ElGamal, the number of faults needed N_f is a bit more than quadratic with respect to the secret size $|x|$

Table 3. Six experimental software results with increasing factorization bound for our attack on ElGamal. In each column, we display the number of faults needed to recover m bits of the secret for each of the experiments.

Exp n°	Bound on t	$m = 64$	$m = 128$	$m = 256$	$m = 512$	$m = 1024$
1	10^4	5794	39790	104258	786038	3366136
2	$2 \cdot 10^4$	4902	24054	126230	611494	2893428
3	$5 \cdot 10^4$	4902	24054	126230	571170	2688882
4	$5 \cdot 10^4$	8810	21742	102974	572170	2931512
5	$5 \cdot 10^4$	7420	23290	142810	696046	3011938
6	$10 \cdot 10^4$	3078	43274	135194	636864	2808468

for all r_j that have probability larger than $b_s = 10^{-s}$ for $4 \leq s \leq 10$. The last row in the table corresponds to 1024-bits security and predict that about three million faults are needed to recover the secret. The table and the figure additionally show that, at least for this range of parameters, the attack complexity is a bit more than quadratic in the size of the secret.

Table 3 exhibits the amount of faulty primes generated in order to recover m bits of the private key x in various simulations. In these experiments, each faulty p^* was factorized up to a bound that is indicated in the first column of the table (our predictions considered a bound of $5 \cdot 10^4$). As can be seen, the practical results match pretty well the theoretical expectations. Moreover, the table shows that considering more factors of p^* slightly decreases the numbers of faulty primes required to find the key.

5 Open Discussion and Future Directions

We have shown both theoretically and by software simulation that fault attacks on public key elements are successful against the DLP-based DSA and ElGamal signature schemes. Our attacks are however not yet practical because they require that the attacker guesses the faulty values he induced, and because the quantities of faulty signatures needed are too large for practical applications.

5.1 Finding the Faulty Value on the Public Key Elements

Blind Method [10]. As we have seen above, if Condition (4) is verified for (p^*, q^*), we recover x modulo r from the discrete logarithm of β in basis α. However, if the adversary does not know the faulty values, he cannot check whether Condition (4) holds. What he can do, is storing all computed values and distinguish the correct one by a statistical method. We point out that the discrete logarithm of β with respect to α may not exist. In this case, the attacker knows that Condition (4) is not verified and simply discards the fault. When the condition does not hold but the logarithm exists, we assume that all values are as likely for $x \bmod r$. Consequently, a bias-based attack may be built for DSA. We remark however, that due to the small probability for Condition (4) to occur, the bias will be much smaller. The same remark holds for our attack against ElGamal.

Collision Method [10]. This method requires a different model of faults, namely a *Dictionary* of possible faulty values. For our attack on DSA, this dictionary $\mathcal{D}$ would of course be two-dimensional. In [10], the authors use the notion of *marker* of an element of the dictionary, which would be defined as couples (t, r) for DSA, where $(t \mid q^*) \wedge (\varphi(t) \mid q^*) \wedge (t \mid p^*)$ and r is a not too small factor of $\varphi(t)$. For every fault $(p^*, q^*) \in \mathcal{D}$ and every marker (t, r), the attacker keeps the discrete logarithm $DL(\alpha, \beta, t, r)$. As soon as two identical values are found, this common value is believed to be the true value, that is $x \bmod r$. In order to avoid false positives, $\sqrt{r}$ should be much larger than $\mid \mathcal{D} \mid$ [10], so t must be much larger than $\mid \mathcal{D} \mid^2$.

We argue that this method cannot be applied in the context of our attack on DSA, simply because most likely, no element of the dictionary will have an appropriate marker. Indeed, the probability that a random element satisfies $(t \mid q^*) \wedge (\varphi(t) \mid q^*) \wedge (t \mid p^*)$ is $\frac{1}{t \cdot \operatorname{lcm}(t, \varphi(t))}$ so the probability that an element of the dictionary has a marker (t, r) for some $t > \mid \mathcal{D} \mid$ is smaller than

$$\sum_{t=\mid \mathcal{D} \mid^2}^{\infty} \frac{1}{t \cdot \operatorname{lcm}(t, \varphi(t))} \leq \sum_{t=\mid \mathcal{D} \mid^2}^{\infty} \frac{1}{t^2} \approx \frac{1}{\mid \mathcal{D} \mid^2}.$$

The same conclusion holds for ElGamal.

Optimal Method [10]. This method uses the whole information available from faults, in the sense that it keeps all possible values for the sequence of faults $((p_1^*, q_1^*), (p_2^*, q_2^*), (p_3^*, q_3^*), \ldots)$ that are coherent with the faulty signatures $((u_1, v_1), (u_2, v_2), (u_3, v_3), \ldots)$. Coherency conditions include for example that $\gcd(g, p^*, q^*)$ divides u^*. Moreover it is "optimal" in the sense that it associates a selectivity parameter related to a "maximum-likelihood selection" to each possible sequence of faults.

In theory, this method can be applied to our attacks on DSA and ElGamal. However, handling the lists of fault values will most likely become intractable in practice.

Further Methods. By relaxing the condition $\sqrt{r} \gg |\ \mathcal{D}\ |$ we can allow some false positives in the collision method (and discriminate between false and true positives via coherency conditions). In our attack on DSA, we could also think of having a dictionary for p^* only. Furthermore, we could use a fault model in which an induced error occurs only on a few bytes instead of on the whole value. This is reasonable because a byte or a word is a basic unit for communication and computation in modern embedded devices such as smart cards. Therefore we can reduce the possible candidates for faulty values.

5.2 Reducing the Required Number of Faulty Signatures

As $\mathrm{lcm}(\{r_i\})$ should be larger than the size of the secret key, the required number of faulty signatures dramatically increases with the secret key size. To partially solve this problem, we can think of a hybrid method recovering part of the secret key with our attack and the remaining bits with an exhaustive search. In this case, we should find the optimal number of bits by considering the trade-off between the required number of faulty signatures and the amount of computation for the exhaustive search.

6 Conclusion

For the first time in the literature, we have shown that fault attacks on two DLP-based schemes, DSA and ElGamal signature schemes, are possible by inducing faults on the public key. Both attacks were carefully analyzed in order to derive their probabilities of success. In each case, the analysis was validated by software simulation. The amount of faulty signatures needed to recover a 160-bit DSA secret key is about $\sim 4 \cdot 10^7$, and $\sim 3 \cdot 10^6$ for a 1024-bit ElGamal.

In order to render the attack more practical, some improvements are required to reduce the number of faulty signatures and to ease the finding of the faults induced. But fault attacks against DLP-based schemes are meant to improve over time, we therefore recommend checking the integrity of public parameters whatever the underlying number theoretic problem, that is in the case of DLP-based scheme as much as in the case of ECDLP- or IFP-based schemes.

References

1. Agrawal, D., Archambeault, B., Rao, J., Rohatgi, P.: The EM side-channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)
2. Bao, F., Deng, R., Han, Y., Jeng, A., Narasimhalu, A., Ngair, T.: Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 115–124. Springer, Heidelberg (1998)
3. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. In: Fault Diagnosis and Tolerance in Cryptography in association with DSN 2004 – The International Conference on Dependable Systems and Networks, pp. 330–342 (2004)

4. Biehl, I., Meyer, B., Muller, V.: Differential fault attacks on elliptic curve cryptosystems. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 131–146. Springer, Heidelberg (2000)
5. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
6. Blömer, J., Otto, M., Seifert, J.-P.: Sign change fault attacks on elliptic curve cryptosystems. In: Fault Diagnosis and Tolerance in Cryptography FDTC 2005, pp. 25–40 (2005)
7. Blömer, J., Seifert, J.-P.: Fault based cryptanalysis of the advanced encryption standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
8. Boneh, D., DeMillo, R., Lipton, R.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
9. Boneh, D., DeMillo, R., Lipton, R.: On the importance of eliminating errors in cryptographic computations. Journal of Cryptology 14(2), 101–119 (2001); An earlier version appears in [8]
10. Brier, E., Chevallier-Mames, B., Ciet, M., Clavier, C.: Why one should also secure RSA public key elements. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 324–338. Springer, Heidelberg (2006)
11. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
12. Brumley, D., Boneh, D.: Remote timing attacks are practical. In: Proceedings of the 12th Usenix Security Symposium, pp. 1–14 (2003)
13. Ciet, M., Joye, M.: Elliptic curve cryptosystems in the presence of permanent and transient faults. Design, Codes and Cryptography 36(1), 33–43 (2005)
14. Dhem, J.-F., Koeune, F., Leroux, P.-A., Mestré, P., Quisquater, J.-J., Willems, J.-L.: A practical implementation of the timing attack. In: Schneier, B., Quisquater, J.-J. (eds.) CARDIS 1998. LNCS, vol. 1820, pp. 167–182. Springer, Heidelberg (2000)
15. ElGamal, T.: A public key cryptosystems and a signature scheme based on Discrete-Logarithm. IEEE Trans. Information Theory 31(4), 469–472 (1985)
16. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
17. Giraud, C., Thiebeauld, H.: A survey on fault attacks. In: Smart Card Research and Advanced Applications VI - 18th IFIP World Computer Congress, pp. 159–176. Kluwer Academic Publishers, Dordrecht (2004)
18. National institute of standards and technology. Digital Signature Standard. NIST FIPS PUB 186-2 (2000)
19. Joye, M., Quisquater, J.-J.: A new and optimal chosen message attack on RSA-type cryptosystem. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 302–313. Springer, Heidelberg (1997)
20. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
21. May, T., Woods, M.: A new physical mechanism for soft errors in dynamic memories. In: Proceedings of the 16-th International Reliability Physics Symposium (1978)

22. Messerges, T., Dabbish, E., Sloan, R.: Examining smart-card security under the threat of power analysis attack. IEEE Transactions on Computers 51(5), 541–552 (2002)
23. Muir, J.: Seifert's RSA fault attack: simplified analysis and generalizations. IACR Eprint archive (2005)
24. Naccache, D., Nguyên, P.Q., Tunstall, M., Whelan, C.: Experimenting with faults, lattices and the DSA. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 16–28. Springer, Heidelberg (2005)
25. Seifert, J.-P., Römer, T.: Electromagnetic analysis EMA: Measures and countermeasures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
26. Quisquater, J.-J., Samyde, D.: Automatic code recognition for smartcards using a kohonen neural network. In: Proceedings of the Fifth Smart Card Research and Advanced Application Conference (CARDIS 2002) (2002)
27. Schindler, W.: A timing attack against RSA with the chinese remainder theorem. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 109–124. Springer, Heidelberg (2000)
28. Schindler, W., Quisquater, J.-J., Koeune, F.: Improving divide and conquer attacks against cryptosystems by better error detection correction strategies. In: Proc. of 8th IMA International Conference on Cryptography and Coding, pp. 245–267 (2001)
29. Seifert, J.-P.: On authenticated computing and RSA-based authentication. In: Proc. of ACM conference on computer and communications security 2005, pp. 122–127 (2005)
30. Walter, C., Thompson, S.: Distinguishing exponent digits by observing modular subtractions. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 192–207. Springer, Heidelberg (2001)

Weaknesses in BankID, a PKI-Substitute Deployed by Norwegian Banks

Kristian Gjøsteen

Department of Mathematical Sciences
Norwegian University of Science and Technology
kristian.gjosteen@math.ntnu.no

Abstract. BankID is a PKI-substitute widely deployed by Norwegian banks to provide digital signatures and identification on the internet. We have performed a reverse-engineering of part of the BankID system and analysed the security protocols and the implementation of certain cryptographic primitives. We have found cryptographic weaknesses that may indicate security problems, protocol flaws facilitating man-in-the-middle attacks, and implementation errors facilitating strong insider attacks. We also note that the system suffers from severe privacy problems.

1 Introduction

A public key infrastructure (PKI) is often defined as an infrastructure that provides users with public key encryption services, digital signatures etc. Deployment of a mass-market PKI is seen by many as a key enabler for efficient and secure online public services.

More narrowly, a public key infrastructure can be defined as an authenticated channel for transport of public encryption or verification keys. The main obstacle to efficient implementation of this authenticated channel is the challenging certificate revocation problem: how to prevent a public key from being used if the corresponding secret key is compromised. For mass-market PKIs using smart cards, the certificate must be revoked if the smart card is lost. One must expect a large number of certificate revocations.

One of the common solutions for key revocation, the certificate revocation list (CRL), becomes impractical when many certificates are revoked. Online certificate status checks or other online certificate lookup solutions are of course online and may also suffer from privacy problems.

Another significant obstacle to solutions based on smart cards is software and hardware support, exacerbated by the many possible combinations of hardware, operation systems and other user software.

BankID is a system designed to provide digital signature services on the internet with a simpler technical solution than most traditional PKI systems, greatly simplifying the revocation problem as well as most software and hardware related problems. It has been developed by Norwegian banks and is used as a login mechanism for many internet banks, ensuring mass-market deployment.

For digital signatures to have any legal force, it is a basic requirement that the user should have sole control of his signing key (see for instance Article 2 of [1]). The main simplifying idea in the BankID system is to ignore this requirement. The system stores the users' signing keys in a central infrastructure that makes signatures on the users' behalf when requested to do so. This means that the BankID system – in principle – can do anything with the users' signing keys, quite unlike a traditional PKI where the trusted certificate authority can only issue forged certificates.

To get the BankID infrastructure to make signatures, the user must authenticate himself. There are minor variations that depend on the user's bank, but essentially two-factor authentication is used. The user has a password and a token that outputs a one-time pin.

To identify himself on a merchant's web site, the user runs an identification protocol with the merchant, involving signing and verifying challenges. The user asks the bank infrastructure not only to do signature generation, but also signature verification.

The actual execution of the protocol is done by a digitally signed Java applet running in the user's web browser as part of the web site's login page. The user enters his credentials into the Java applet. When the identification protocol completes, the merchant considers the user identified for the session.

Clearly, any approach based on entering credentials into applets will be highly susceptible to phishing attacks, no matter how such a solution is implemented. We note that one of the counter-measures proposed by the banks [9] is to verify the signature on the Java applet.

There has been little independent public analysis of the BankID system. Hole et al. [7] identified several security and privacy problems, and did a risk analysis. Espelid et al. [4,5] described a strong phishing attack against BankID, defeating the banks' proposed counter-measures.

In this paper, we do a deeper analysis of parts of the BankID system based on a reverse-engineering of the Java applet. We describe a new man-in-the-middle attack against the identification part of the system. We describe flaws in the cryptographic subsystem that allow devastating insider attacks against all aspects of the system. We elaborate on the privacy problems in the system, and briefly discuss how some privacy problems can be eliminated.

This paper is structured as follows: Sect. 2 sketches reasonable security goals and attack models for BankID. Sect. 3 contains our analysis and findings. Finally, in Sect. 4 we discuss some implications of our findings.

2 Security Goals and Attack Models

A system's security goals describe undesirable system behaviour that should not occur, even in the presence of an attacker. In order to analyse a system, we also need to select an attack model that describes the attacker's capabilities.

For a digital signature infrastructure to be useful, it must provide a reasonably strong guarantee that every valid signature was intentionally produced by the user. A *signature forgery* is defined as the creation of a valid signature without the user's consent.

Likewise, every time a user is accepted as identified by some honest web site, a digital identification infrastructure must provide a reasonably strong guarantee that for any such session, the communicating partner must be the user's terminal. An *impersonation attack* succeeds when an honest web site accepts the user as identified in a session, but the user's terminal is not the communicating partner for that session.

The natural security goal of any digital signature and identification infrastructure must therefore be to prevent signature forgeries and impersonation attacks.

One very strong attack model is where we allow the attacker to compromise the user's terminal. Unless the system provides the user with at least a secure input device and a secure display, separate from the user's terminal, it is impossible to provide any meaningful security[1] in such an attack model. We shall therefore not allow the attacker to compromise the user's terminal.

So-called *phishing attacks* exploits the fact that most users are unable to correctly decide if a web page [6] is authentic or fake. While one can hardly expect a system to protect a user from signing some arbitrary document without looking at it, we can expect the system to prevent impersonation attacks, regardless of how the user (mis)behaves.

In other words, a model where the attacker can fool the user into identifying himself on a web site of the attacker's choice is reasonable. For digital signatures, we shall let the attacker fool the user into signing any document of his choice. We shall not consider a signature to be a forgery if the user could have had the document displayed on his terminal before giving his consent to the signature production.

For a system where the user's signing key is stored with a trusted third party, we also need to consider insider attacks, where employees of the trusted third party may attempt to abuse their access to the system.

It is hard to imagine a maintainable system for which there does not exist a coalition of insiders whose combined privileges can provide access to the user's signing key. This is highly problematic, since in principle, there is no technical obstacle preventing signature forgeries. But for our present purpose an attack model where no security is possible is not interesting.

We shall work with a weaker attack model, where we assume that any tamper-resistant hardware, such as hardware security modules (HSMs), function as intended, that is, the attacker cannot compromise them.

To summarize our attack model:

- The attacker can have the user run the identification protocol with any web site of his choice.
- The attacker can have the user sign any document of his choice.
- The attacker can compromise any part of the system except for the user's terminal or any part of the system specifically designed to be tamper-resistant.

[1] A smart card prevents theft of the user's private key, but not misuse.

Resistance against these attacks should not be the only design goal for an infrastructure for digital signatures and identification, privacy should also be an important design goal. The infrastructure operator should learn as little as possible about his users based on their use of the infrastructure. When signing keys are stored with the infrastructure operator, no practical solutions are possible where the operator does not learn when a signing key is used. The natural privacy goal must therefore be that the operator should not learn anything about the use of the system, except when the users' signing keys are used.

3 The Analysis

Our original goal was to determine exactly what kind of private information was sent from the BankID Java applet to the bank infrastructure. Since no public documentation is available for the BankID system, we decided to extract protocol information from the Java applet itself.

We did this by using a Java decompiler that transforms Java bytecode into Java source code. It then turned out that the Java source code had been obfuscated by giving most symbols meaningless names. Such obfuscation is basically a substitution cipher, where punctuation, strings and library calls are untouched and constitute plain text. Reading the source code without knowing the substitution is time consuming, but still fairly easy with appropriate tools.

3.1 Design Outline

When a user wants to use BankID to identify himself to some merchant, he directs his browser to the merchant's web site. The merchant's web site includes a Java applet signed by the banks. The documentation for the BankID system expects the user first to verify the digital signature on the Java applet [9]. If the signature is correct, the user enters his credentials (exactly what these are varies, but include a passphrase and one or more one-time pin) into the Java applet. The Java applet then runs an identification protocol with the merchant and the infrastructure. If the identification protocol concludes successfully, the Java applet redirects the user's browser to an URL supplied by the merchant.

The communication between the Java applet and the merchant is optionally protected by SSL[2], while the communication with the bank infrastructure is always protected using SSL. In both cases, the usual web PKI is used to authenticate the merchant and infrastructure public SSL keys.

The Java applet runs a mutual identification protocol with the merchant web site. The exact details of this protocol are unknown, but we know that the user and merchant both issue challenges, and that both sign something.

The Java applet does not have the user's signing key, but runs a protocol with the infrastructure to have it make the required signature. To further simplify the system, the Java applet does not verify the merchant's signature, but instead asks the infrastructure to do so.

[2] After we reported our findings in Sect. 3.3 to the banks, SSL was made mandatory.

The protocol used between the Java applet and the infrastructure is a cryptographic protocol. An RSA encryption key is embedded in the Java applet. Every time the applet wants to send a message to the infrastructure, it encrypts a randomly chosen symmetric key using the RSA encryption key, then encrypts the message using the symmetric key. Both ciphertexts are sent to the infrastructure through an SSL tunnel. The reply from the infrastructure is encrypted using the Java applet's symmetric key, sent through the established SSL tunnel.

This motivates the following conjecture, which has been confirmed by the banks. An SSL gateway sits on the edge of the bank network and the SSL tunnel stops. The SSL gateway forwards the applet's two ciphertexts to a signature HSM. The encrypted response is then sent to the SSL gateway and from there through the SSL tunnel to the Java applet.

Further clues inside the Java applet motivates the sketch of the complete system given in Fig. 1.

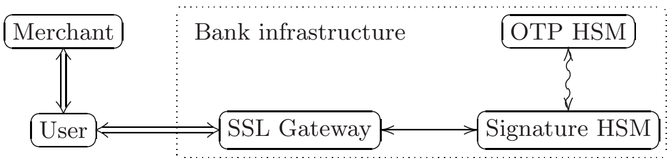


Fig. 1. The principal actors in the BankID security protocol and their communication lines. ($\Leftrightarrow$ denotes SSL tunnel, $\leftrightarrow$ denotes no SSL tunnel, $\leftrightsquigarrow$ denotes unknown protocol.)

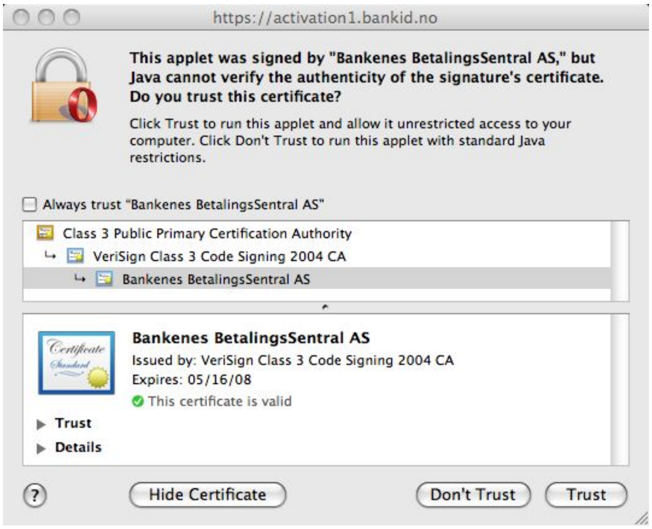
For public key encryption, RSA with PKCS#1 1.5 padding is used. Curiously, none of the easy countermeasures to standard attacks on PCKS#1 1.5 listed in [8] seem to be implemented.

For symmetric encryption, 3DES is used in CBC mode with an all-zeros initialization vector (IV). Since the same key is always used for two messages, we note that using an all-zeros IV may leak information, but this seems to be of minor significance. More interesting is that fact that there is no integrity protection, a very curious omission.

We note that we do not have enough information to decide if the above flaws lead to exploitable security weaknesses. But the lack of integrity protection will with high probability allow attacks on the PKCS#1 1.5 padding that can be used to compromise user passwords, and perhaps also steal a user's credentials.

3.2 Observations on Earlier Phishing Attacks

Verifying the signature on the Java applet, as instructed by the bank's recommended procedure for use of BankID [9], is intended to protect against phishing attacks. Given that the typical browser dialog is somewhat hard to understand (see Fig. 2), this protection is rather weak. It is also unclear to us, given the complexity of modern web browsers, if a conscientious and educated user following


https://activation1.bankid.no

This applet was signed by "Bankenes BetalingsSentral AS," but Java cannot verify the authenticity of the signature's certificate. Do you trust this certificate?

Click Trust to run this applet and allow it unrestricted access to your computer. Click Don't Trust to run this applet with standard Java restrictions.

Always trust "Bankenes BetalingsSentral AS"

Class 3 Public Primary Certification Authority
↳ VeriSign Class 3 Code Signing 2004 CA
↳ Bankenes BetalingsSentral AS

Bankenes BetalingsSentral AS
Issued by: VeriSign Class 3 Code Signing 2004 CA
Expires: 05/16/08
This certificate is valid

Fig. 2. Verifying the signature on a Java applet: Java cannot verify the authenticity of the certificate, but the certificate is valid

the bank's recommended procedure can expect BankID to provide security. But for the sake of argument, we shall assume so.

Unfortunately, Espelid et al. [4,5] showed that this procedure was insufficient. They noticed that the mutual identification protocol between merchant and user is run essentially inside an SSL tunnel, but without any binding between the identification protocol and the SSL tunnel.

The basic idea is to have the Java applet appear in the phishing web page, let the user enter his credentials into the applet (which he has been told to trust), and then hijack the established session. This is an impersonation attack.

The Java applet gets the address of the merchant's web page from its parameters. The phishing web page can therefore control this address and direct the applet to the phishing web site. The phisher starts a session with the merchant's web page, and simply forwards protocol messages to and from the user's Java applet to the merchant. Finally, the merchant considers the user to be identified, and the phisher hijacks the session.

Espelid et al. developed [4,5] proof-of-concept software for a phishing attack, notified the banks and demonstrated the attack for the Financial Supervisory Authority of Norway (Kredittilsynet).

By comparing the vulnerable version of the Java applet with the corrected version, we see that a counter-measure against such man-in-the-middle attacks is present in both versions, but the code has been disabled in the vulnerable version. This means that the protocol was originally designed so that the merchant and the applet would agree on the merchant web site's host name and IP address, but for some reason, this code had been disabled.

It is somewhat interesting that when the code to verify the merchant host name and IP address was turned back on (approximately eight months after the banks were first notified), the protocol messages were left unchanged, except for the applet version number. But the version number was never cryptographically protected. Further, the old Java applet was never revoked. It was therefore easy to modify [5] the original phishing attack to use the old Java applet and then change the version number in the message flow to make the merchant and the infrastructure believe they were talking to the new Java applet.

A new Java applet was released, this time with a modified protocol that cryptographically protects the version number. This patches this specific hole.

3.3 A New Man-in-the-Middle Attack

The Java applet can run the identification protocol with the merchant either inside an SSL tunnel or in the clear. A more careful study of the Java applet reveals the following fact. The cryptographic protocol ensures agreement on the merchant site's host name and IP address. There is no mechanism to ensure that the applet and merchant site agree on whether SSL is used.

This fact suggests the following new attack, where the attacker controls the network between the user and the merchant. He presents the user with a phishing web page containing the Java applet. The applet gets the correct parameters, except that the merchant URL is changed from `https://...` to `http://...`. The applet proceeds to contact the merchant using the unencrypted http protocol. The attacker intercepts these requests and redirects them to the real merchant server inside an SSL tunnel. The replies can easily be forwarded to the applet. After the merchant considers the user identified, the phisher hijacks the session.

Unlike Espelid et al. [4], we did not develop a complete attack. We tested the applet in a simulated attack, demonstrating that the attack would work. The banks have issued a new applet that requires SSL communication with the merchant. This patches this specific hole.

3.4 An Insider Attack

It is well-known that high-quality pseudo-random numbers are vital for cryptographic protocols. Many protocols break if an attacker can predict the pseudo-random numbers used to generate protocol messages. In particular, the protocol between the Java applet and the signature HSM breaks completely, since an attacker that can predict the applet's pseudo-random numbers can also predict the key used to encrypt the messages. Anyone who knows the encryption key and has access to the internal bank network, between the SSL gateway and the HSM, can alter message contents at will. It is also easy to recognize when you have the correct key, simply by comparing the resulting RSA ciphertext with the transmitted ciphertext.

The Java applet uses the Java `SecureRandom` class to generate pseudo-random numbers. Unfortunately, when run on older Java versions, the Java applet does not use the built-in seed generation, but instead generates the seed on its own.

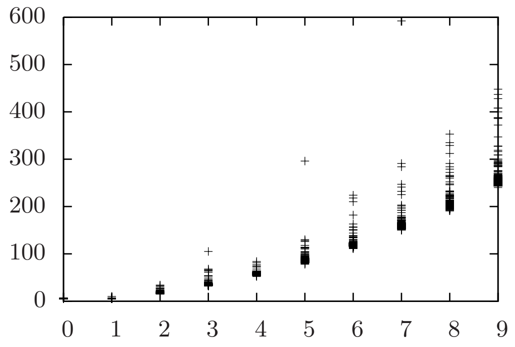


Fig. 3. Time measurement $\sum \delta_i$ plotted against the number of measurements c. Four outlayers have been removed out of 11 000 samples.

(Espelid et al. [4] noted that this seeding was suspicious, but did follow up on the observation.)

The seed is generated by first selecting a number $c \in \{1, 2, \ldots, 9\}$ and computing

$$ct + cm + \sum_{i=0}^{c-1} \delta_i \bmod 2^{64},$$

where t is the current system time in milliseconds, m is the amount of free memory in the Java virtual machine, and δ_i is a series of time measurements.

We ran the algorithm 11 000 times on a computer that was in ordinary use (web browsing, editing, etc.). We measured the sum of the time measurements $\sum \delta_i$. The results are shown in Fig. 3.

We must estimate the total entropy in the seed. It is quite clear that the variance in the time measurements is very small, a close look at the numbers suggests at most five bits of entropy, depending on c. The amount of free memory in the Java virtual machine seems to be a constant depending only on the implementation. Again, this will introduce at most a few bits of entropy.

The current system time is more difficult to analyse. The current system time is sent as part of a client hello message in the SSL protocol [3], and can hardly be classified sensitive information. A determined attacker will certainly be able to determine the system time within a second. Since milliseconds are used in the seeding process, this introduces at most 10 bits of entropy. Additionally, the attacker must determine the start time of the Java applet. This depends on whether or not user interaction is involved, but will most likely be on the order of minutes, introducing perhaps another 9-10 bits of entropy.

A more casual attacker can still make a useful estimate of the Java applet start time. Some users keep their system clocks synchronized with time servers (some operatings systems make it easy to turn such synchronization on). One would conjecture that only very few users let their system clocks run more than a couple of hours wrong. An absolute worst-case scenario for an attacker is an uncertainty of a few hours, leading to approximately 25-26 bits of entropy. A more realistic scenario would allow for an uncertainty of a minute or two, which is on the scale of the uncertainty resulting from user interaction.

The end result is that the system time contributes somewhere between 10 and 26 bits of entropy, depending on how determined the attacker is and whether user interaction is involved.

Finally, we need to consider c. It is sampled using the `java.util.Random` class, which is initialized with the current system time in milliseconds. We have already counted this entropy.

To summarize, a worst-case scenario for an attacker leads to a total of say 35 bits of entropy in the system. A more realistic scenario suggests approximately 25 bits of entropy, while one can assume no more than 15 bits of entropy against a determined attacker, perhaps less.

This is clearly not enough. For 15-20 bits of entropy, an adversary could most likely determine the random seed in real time. For 25 bits of entropy, a determined adversary might be able to determine the random seed without causing noticable delay. For 35 bits of entropy, an attacker that can predict an approximate access time might through precomputation mount a real time attack.

Alternatively, an attacker could passively monitor the protocol run and extract the user's password, a serious security failure.

3.5 Privacy Problems

First of all, BankID uses Norwegian personal identity numbers as user identifiers. These numbers are similar to US social security numbers. This means that any merchant who uses BankID must identify his users by their personal identity numbers. This makes merchant databases easily linkable, which is problematic from a privacy point of view. Ideally, an identification system should allow the user to select any number of pseudonyms with separate digital identities.

Furthermore, our study of the protocol between the Java applet and the bank infrastructure shows that it violates our privacy requirements, as defined in Sect. 2. The infrastructure learns not only that the signing key is used, but also the name of the merchant.

The protocol for digitally signing documents sends a hash of the message to be signed to the infrastructure. Given a hash of a message, it will be easy to decide if the message is on a list of candidate messages. For instance, the bank may be able to recognize if the user signs a competitor's standard contract. This also violates our privacy requirements.

These observations are of course trivial. It therefore is seems quite clear that the BankID protocols have not been designed with privacy in mind. We shall briefly explain how the protocols could be changed so that private information does not leak.

The signing protocol should obviously use blind signatures.

In the identification protocol, the user should not sign the names and challenges directly, but rather sign a hash of that data. Under reasonable assumptions on the hash function, this should hide all information from the infrastructure.

Verifying the merchant's signature is somewhat more difficult. Obviously, verification cannot be delegated to the infrastructure. This means that the applet must somehow obtain the merchant's certificate and check that it has not been revoked.

Obtaining the merchant's certificate is easy, we get it from the merchant himself. There are several possibilities for checking its revocation status. If a special certificate revocation list for merchant certificates is maintained, the number of revoked certificates may be very small, and downloading the entire CRL may be feasible. The user cannot run an online status checking protocol directly against the infrastructure, but it may be possible to run it via the merchant. As a third possibility, the infrastructure may issue certificates with a very short lifetime, eliminating the need for revocation lists.

4 Concluding Remarks

Any system that is to provide legally binding digital signatures and access control to sensitive private information, such as bank accounts and medical records, must be held to the highest standards.

The basic design of BankID is defective, given that the BankID infrastructure has control over the user's signing key. Furthermore, the technical solution chosen is highly susceptible to phishing attacks.

The implementation of the BankID Java applet is seriously flawed. The random number generator flaw is an astonishingly elementary mistake. While we cannot say positively that the curious lack of integrity and counter-measures to PKCS#1 1.5 attacks lead to attacks, we contend that the design is fragile and highly inappropriate for a system of this kind.

Having an identification protocol running both with and without an SSL tunnel without any cryptographic link between the two is again an elementary mistake. We note that automatic protocol verification tools, such as Scyther [2], will easily find our man-in-the-middle attack.

The privacy problems we point out are quite elementary and are easily avoided using standard cryptographic constructions. It is clear that privacy was not a requirement during the system design.

We remark that the cryptographic problems faced by the BankID Java applet are fairly standard. The threats faced by the infrastructure, such as insider attacks, are significantly more challenging. We have very little information about this part of the system, but what we know indicate that powerful insider attacks are possible.

Based on this evidence, it is clear that the current version of the BankID system should see only very limited use, in order to mitigate phishing attacks and privacy problems. No system developed by the banks should be allowed into wide-spread use without rigorous public evaluation by independent experts.

As a final note, we mention that our analysis was essentially carried out based on publicly available information without any assistance from the banks. The banks have been informed of our results, they have corrected one of the flaws we have pointed out, and are working to analyse and (perhaps) correct the remaining defects at the time of writing.

Acknowledgements

The author would like to thank Kjell Jørgen Hole for valuable help and encouragement in the analysis work, and the anonymous referees for many helpful comments.

References

1. Directive 1999/93/EC of the European parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. Official Journal of the European Communities, L13, 43, 12–20 (2000)
2. Cremers, C.J.F.: Scyther - Semantics and Verification of Security Protocols. Ph.D. dissertation, Eindhoven University of Technology (2006)
3. Dierks, T., Rescorla, E.: The transport layer security (TLS) protocol version 1.1, RFC 4346 (April 2006)
4. Espelid, Y., Netland, L.-H., Klingsheim, A.N., Hole, K.J.: A proof of concept attack against Norwegian internet banking systems. In: Proc. of the 12th International Conference on Financial Cryptography and Data Security (FC 2008), Cozumel, Mexico, January 28-31,2008 (2008)
5. Espelid, Y., Netland, L.-H., Klingsheim, A.N., Hole, K.J.: Robbing banks with their own software—an exploit against Norwegian online banks, September 8-10 (2008); To be presented at the 23rd International Information Security Conference (SEC 2008), Milan, Italy (2008)
6. Gutmann, P.: Security usability, Draft (February 2008), `http://www.cs.auckland.ac.nz/~pgut001/pubs/usability.pdf`
7. Hole, K.J., Tjøstheim, T., Moen, V., Netland, L.-H., Espelid, Y., Klingsheim, A.N.: Next generation internet banking in Norway. Technical Report 371, Department of Informatics, University of Bergen (February 2008), `http://www.nowires.org/Papers-PDF/BankIDevaluation.pdf`
8. RSA Laboratories. PKCS #1: RSA cryptography standard, version 2.1 (June 2002)
9. Trygg bruk av BankID (in Norwegian) (Feburary 13, 2007), `http://www.bankid.no/utskrift.db2?id=4062`

An Open Mobile Identity Tool: An Architecture for Mobile Identity Management

Konstantin Hyppönen

University of Kuopio, Department of Computer Science
POB 1627, FIN-70211, Kuopio, Finland
Tel.: +358 17 162587; Fax: +358 17 162595
Konstantin.Hypponen@cs.uku.fi

Abstract. We present an architecture for a flexible and open mobile electronic identity tool, which can work as a replacement for numerous ID cards and licenses. In addition, it can be used in various payment and user authentication scenarios. The tool is mobile phone based and uses a security element (e.g., a SIM card) for storing sensitive identity information. We follow the design for privacy principles, such as minimisation of data collection and informed consent of the user. The tool can be implemented using currently available handset technology.

1 Introduction

In the modern community, the identity of every person is composed of numerous personal characteristics (biological, social), roles, licenses, and business relations. In the electronic world, the same can be seen in the multitude of user accounts in different services. The problem of abundant registrations and, ultimately, remembering numerous passwords, has been addressed by federated identity management solutions [1,2,3,4]. However, their applicability is mostly limited to various on-line scenarios, whereby a service provider can trust a third party (identity provider) to securely authenticate the user.

Identity management schemes have also been developed for mobile users. An extensive study of mobile identity management schemes has been conducted by FIDIS [5]. As an example, a tool called iManager [6] enables the user to manage her partial identities on a personal digital assistant (PDA), and provides an interface for selecting the suitable identity for every use case. iManager, as most other tools, is designed for on-line business scenarios.

Identity-related information and credentials can be stored on mobile devices in different ways. The unprotected memory of a hand-held device is suitable as a data storage for non-critical applications. For example, mobile browsers can store passwords and data for filling forms on the Internet. For more secure key provisioning and program loading, trusted platform modules [7] or secure mobile environments [8,9] can be used [10]. Yet another option is to securely store identity-related information on the subscriber identity module (SIM) card or another smart card attached to or embedded in the hand-held device [11].

In addition to security, issues of usability and trust are of major importance in mobile identity systems. Since hand-held devices such as mobile phones are constantly connected to wireless networks, they are seen by many as potentially insecure gadgets that can reveal sensitive information about one's identity to unauthorised parties. The application looks even more dangerous if it includes a payment component. Therefore, the implementations must ensure that the user has full control over data sent by the application, and can prevent disclosure of any information by the device. Moreover, the systems that need information about users must receive only a necessary minimum of it, and remove the information as soon as it is not needed [12].

Privacy-enhancing technologies (e.g. [13,14]) enable building identity management systems whereby a person can make flexible proofs about the values of identity attributes. For instance, one can prove that one is over 22 and below 65 years old without surrendering any more information. Furthermore, anonymous credential systems (e.g., [15]) make such proofs anonymous and unlinkable. However, such solutions tend to be an overkill for most everyday situations in the face-to-face realm. Moreover, computationally they are still rather heavy for hand-held devices [16]. In addition, the issues of their standardisation and acceptance for officially recognised documents are yet unclear. In contrast, solutions based on X.509 public-key certificates have been utilised in many national electronic identity schemes.

In order to become widely accepted, a mobile identity management system must be perceived by both users and service providers as a secure, trustworthy, usable and ubiquitous tool. It has to work in the person-to-person proximity scenarios as well as in the on-line world. In other words, it has to become a public tool [17], open to be joined and freely used by any participating party.

Our contribution. We present an identity management scheme designed for implementation on mobile phones. In addition, we discuss related infrastructure, implementation and security issues. Identity information and keys are stored on the SIM card. The scheme supports multiple partial identities of the person and can be used for identity verification, biometric authentication, and user authentication in various online and proximity scenarios, in official, business and personal contexts. We use currently available handset technology and open standards as building blocks for our open mobile identity tool.

The rest of the paper is organised as follows. In the next section we describe the architecture of our mobile identity tool and give details of the protocols. Next, in Sect. 3 we overview the infrastructure that has to be established for introducing the tool in everyday life, and make a note of a few implementation issues. In Sect. 4 we discuss issues related to security and usability. Section 5 reviews related work and Sect. 6 concludes the paper.

2 A Mobile Identity Tool

In this section, we describe the architecture and protocols for our mobile identity tool. In the description, we use the terminology given in Appendix A. Readers acquainted with the Pfitzmann-Hansen terminology [18] will find most of the terms familiar.

A flexible identity tool should allow quick selection of a suitable identity profile in the everyday life. Furthermore, it should allow adding and deleting profiles, updating profile information and revoking compromised profiles. For improved flexibility, self-created profiles can also be supported. In addition, usability is of paramount importance if the tool is designed for broad public. Restrictions of hand-held devices, such as small screen sizes and reduced keyboards, highlight the importance of simplicity and usability even more.

2.1 Architecture

Our identity tool is comprised of two parts: a tamper-resistant security element (e.g. a SIM card) and an *identity proxy* used for managing the security element. The identity proxy can be implemented on the mobile phone most easily in Java 2 Mobile Edition (J2ME), as suitable application programming interfaces (APIs) [19,20] are available. In addition to transferring data between the SIM card and identity verifier terminals, features of the proxy include providing a user interface and checking identity verifiers' credentials.

Often the same biometric (e.g., photo) or a pseudonym of the person can be used in different profiles. Therefore, we define first a *pseudonym pool.* Entries of the pool contain the following information:

1. Pseudonym type. A type describes the contents of a pseudonym. The semantic meaning of pseudonym types could be, e.g., "User's photo" or "Customer ID of ShopChain Ltd." Pseudonym types follow a naming convention; for example, the system of object identifiers (OIDs) defined in the ITU-T ASN.1 standard is a natural choice for naming.
2. Pseudonym content. The content can be, for example, a biometric template, a number sequence, or a string.
3. Pseudonym identifier. The identifiers, which will be used as subject names in certificates issued to a person, contain cryptographic hash values of corresponding pseudonym contents. For instance, in case of biometric templates this produces biometric identifiers.
4. Date of creation. The field is used for ensuring that pseudonyms are not outdated. A profile issuer may, for instance, refuse issuing a profile connected with a 10-year old photo.

Different profile issuers may have different demands on pseudonym validity terms, for example, on photo freshness. Therefore, we do not define the validity period for pseudonyms, but only specify their creation date. This may easily lead to a case when there are multiple patterns of the same biometric feature in the pseudonym pool, with different creation dates. However, we do not view this as a problem, because profile issuers may also have different requirements on pattern acquiring procedures, for instance, on photo lighting or set of fingers used in fingerprint patterns.

We define an *identity profile*, corresponding to a partial identity, as the following information set:

1. Pseudonym identifier, which works as a reference to the pseudonym pool.
2. Profile type. Different profiles must have different types. However, the same type of profile in identity tools of different persons must have the same name. OIDs can be used for profile type naming.
3. Private key, generated on-card. The key has a visibility indicator, either *open* or *PIN-protected*. The PIN code must be submitted by the user before an operation with a PIN-protected private key can be performed.
4. Certificate chain, containing the user's public key certificate (PKC). The certificate is issued by the identity vendor to the person's pseudonym and signed with the identity issuer's private key. The "Not before" and "Not after" fields of the certificate define the profile validity period. The certificate chain includes a trusted root certificate. For self-created profiles, root certificate is the same as the user's PKC.
5. Attribute certificates (AC) [21] associate attributes with the public key of the person. Values of attributes in the certificates are masked. Attribute certificates are signed by the identity issuer.
6. Attribute masks. The identity verifier can read an attribute value only if it has received the attribute mask (in addition to the attribute certificate).
7. Attribute visibility indicators. Each attribute can be marked as either *open* or *PIN-protected*. The PIN code must be submitted by the user for a PIN-protected attribute, before the corresponding mask can be retrieved.
8. Attribute values. In addition to the plaintext values of masked attributes, also non-authenticated values can be stored here. For self-created profiles (e.g., a business card profile) attribute certificates are not necessary, and attribute values can be stored as such.
9. Secret keys for symmetric encryption. Secret keys may also have visibility indicators, *open* or *PIN-protected*.

Attribute certificates contain masked values of identity attribute values. An attribute A is stored in the attribute certificate in the following form: $H(A, M)$, where $H(\cdot)$ is a cryptographic hash function, and M is the corresponding attribute mask. Having received the attribute certificate, the mask, and the value of the attribute, the identity verifier can check the attribute authenticity.

A more advanced profile could contain also a specific program for manipulating attributes and their values. In this case, profiles must be loaded into the identity tool as separate smart card applets. In our scheme, we skip this type of profiles, as their design and implementation are difficult due to security problems inherent to applet loading procedures.

The mobile identity tool is issued to the person on a SIM card by her mobile network operator (MNO). The identity applet has already been loaded to the SIM card by the MNO and contains no identity profiles, except maybe a profile issued by the MNO. It contains, however, a private key and a corresponding PKC named *profile-loading certificate*, issued by the MNO. The subject name in the certificate is the same as the card number. The certificate is used later for authenticating the card before loading any identity profiles to it. The user also installs the identity proxy application on the phone.

We highlight the fact that *any* identity issuer can load a profile. Furthermore, there are no checks on whether the identity issuer has been certified for issuing identities. Therefore, even the person herself can create a new profile on the card (e.g., a business card profile). "Official" profile vendors differ from others only by the fact that their public keys belong to an officially recognised public key infrastructure. The newly generated public-private key pair of the user becomes therefore part of the same infrastructure. If an officially recognised proof of identity is needed, a profile issued by an official identity vendor is invoked.

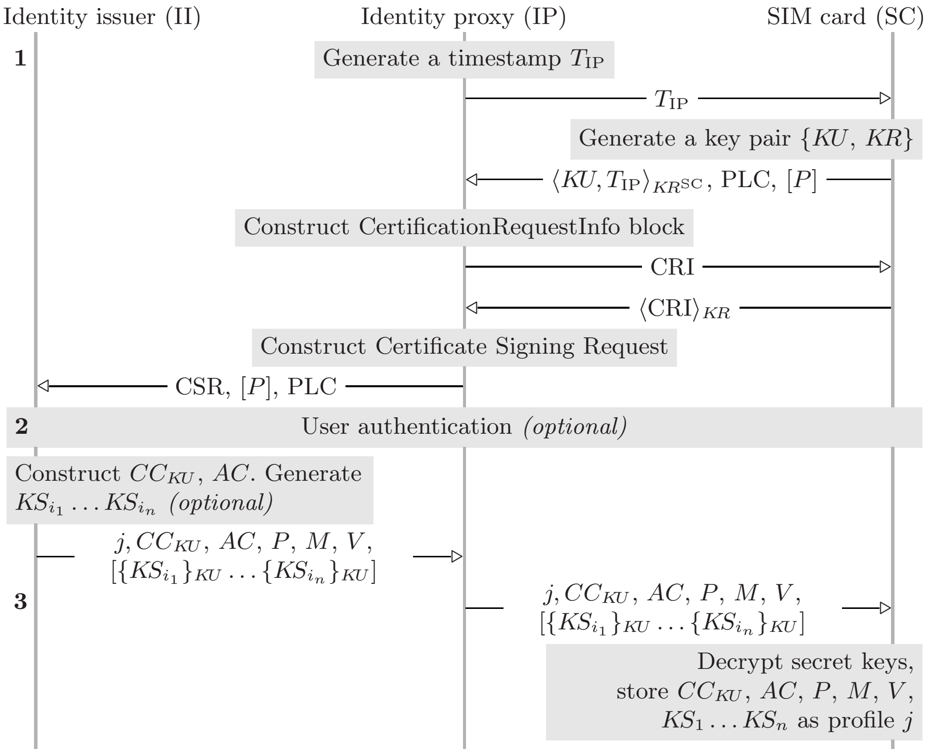
2.2 Loading an Identity Profile

In order to load a profile into the identity tool, a protocol comprised of three phases is followed. First, the SIM card and the identity proxy, acting together, create a certificate signing request (CSR). Second, the identity vendor constructs the profile. Finally, the new profile is loaded on the card. The protocol is outlined in Fig. 1, using notation given in Table 1. We concentrate on essential data sent between the parties, skipping descriptive information about profiles and keys (such as profile names and visibility indicators).

Table 1. Protocol Notation

KU	P*u*blic key
KR	P*r*ivate key
KS	*S*ecret key
K^X	Key of subject X
CC_K	Certificate chain for a public key certificate issued to key K
KS_i	Secret key of type i
PLC	Profile-loading certificate
CRI	CertificationRequestInfo block
CSR	Certificate Signing Request
I^j	Identity profile of type j
K^j	Key belonging to profile I^j
AC^j	Attribute certificate belonging to profile I^j
RC^j	Root certificate of chain CC_{KU^j}
T_X	Timestamp generated by X
N_X	Nonce generated by X
P	Pseudonym
P^j	Pseudonym used in profile I^j
A_i^j	Attribute of type (code) i from profile I^j
M_i^j	Attribute mask for A_i^j
V_i^j	Non-authenticated attribute of type (code) i from profile I^j
M	Set of attribute masks
V	Set of non-authenticated attributes
$\{\cdot\}_K$	Encryption under the key K
$\langle m \rangle_K$	Message m and its signature under key K

Creating a CSR. The process is started by the identity proxy which generates a fresh timestamp T_{IP} and submits it to the *profile manager* (an applet) on the SIM card. A new key pair $\{KU, KR\}$ is generated on card. The public key KU and the timestamp T_{IP} are concatenated, signed using the SIM card specific profile-loading private key KR^{SC} and sent to the identity proxy. If there is a suitable pseudonym P in the pseudonym pool that can be re-used, the identity proxy retrieves it from the card. Otherwise, it either asks the user to enter a new



Identity issuer (II)
Identity proxy (IP)
SIM card (SC)
1
Generate a timestamp T_{IP}
T_{IP}
Generate a key pair $\{KU, KR\}$
$\langle KU, T_{\mathrm{IP}}\rangle_{KR^{\mathrm{SC}}}$, PLC, $[P]$
Construct CertificationRequestInfo block
CRI
$\langle \mathrm{CRI}\rangle_{KR}$
Construct Certificate Signing Request
CSR, $[P]$, PLC
2
User authentication *(optional)*
Construct CC_{KU}, AC. Generate $KS_{i_1} \ldots KS_{i_n}$ *(optional)*
$j, CC_{KU}, AC, P, M, V, [\{KS_{i_1}\}_{KU} \ldots \{KS_{i_n}\}_{KU}]$
3
$j, CC_{KU}, AC, P, M, V, [\{KS_{i_1}\}_{KU} \ldots \{KS_{i_n}\}_{KU}]$
Decrypt secret keys, store CC_{KU}, AC, P, M, V, $KS_1 \ldots KS_n$ as profile j

Fig. 1. Loading a new profile. Numbers in bold indicate start of the protocol phases.

pseudonym, or leaves the creation of it to the identity vendor. The pseudonym is placed in the subject name field of the CSR. The signature of the newly created public key is written in the challenge password attribute of the CSR. The identity proxy constructs the CertificationRequestInfo block of the CSR and acquires a signature for it from the card (the newly created private key *KR* is used for signing). Finally, the CSR is submitted to the identity vendor, together with the profile-loading certificate.

Constructing an identity profile. Having received the CSR, the identity issuer verifies its signature and also the signature $\langle KU, T_{\mathrm{IP}}\rangle_{KR^{\mathrm{SC}}}$ of the submitted public key. Then it verifies the profile-loading certificate. Whenever needed, a normal user authentication procedure with regard to the presented pseudonym is performed. Namely, the user has to prove her connection to that pseudonym by either performing a biometric authentication or presenting a PKC issued to that pseudonym and proving the possession of the corresponding private key. Alternatively, the identity vendor can provide a new pseudonym or accept the one suggested by the user. The identity vendor finally creates a new public key certificate and submits it within its corresponding certificate chain to the identity proxy along with profile-specific identity attributes and/or attribute certificates. In addition, the identity vendor can supply a number of secret keys encrypted using the user's public key. If a new pseudonym is created, its description is attached to it.

Loading the profile. The identity proxy loads the new profile on the card. The card stores the certificates and identity attribute information without performing any checks on them. If a new pseudonym is supplied, it is stored in the identity pool. Also secret keys are decrypted and stored in the profile.

2.3 Identity Proofs and Digital Signatures

A profile manager (an applet on the SIM card) handles identity verifiers' requests arriving at it via the identity proxy and chooses suitable profiles for creating answers to them.

In an identity proof request, an identity verifier indicates the type of profile (j) to be used and a list of codes for authenticated attributes ($i_1 \ldots i_n$) and non-authenticated attributes ($k_1 \ldots k_m$) to be returned. A timestamp T_{IV} and the identity verifier's certificate chain CC_{IV} are attached to the request. As an optimisation, the root certificate can be omitted from the chain, as it is available on the SIM card. The request is signed using the identity verifier's private key KR^{IV}. The identity proxy verifies the public-key certificate by retrieving the corresponding root certificate from the SIM card and performing a normal certificate validation procedure of the certificate chain. Next, it checks the request freshness, and asks for a confirmation from the user, showing the list of requested information. The user can either comply with the request or reject it. If PIN-protected attributes are requested, PIN verification is performed. The identity proof, provided by the SIM card, includes relevant attribute certificates, values and masks. To prove the possession of the profile-related private key, the card signs the combination of two timestamps: the one provided by the identity verifier and the one generated by the identity proxy.

In proximity scenarios, biometric authentication of the user can be easily performed. If a biometric identifier is used as the user pseudonym, corresponding biometric pattern becomes available to the identity verifier, and can be used in normal biometric authentication. For example, if facial biometrics are used, the person's photo can be shown on screen to the identity verifier representative, who can thereon compare it with the person's face.

Another type of operation supported by the identity tool is creating a digital signature of a supplied message. This operation is useful for payments and other services that require authenticated user agreement with service terms. To request a digital signature for message G, the identity verifier (or service provider) submits it to the identity proxy, along with the request for identity-related information. The identity proxy shows the message on the screen. Having read the message, the user enters her PIN code if she agrees to sign it. Finally, message G is appended with a new timestamp and submitted to the SIM card for the calculation of a signature.

Different PIN codes may be used for identity proofs and for digital signatures, to make sure that the user is fully aware of the fact she is signing a message, and

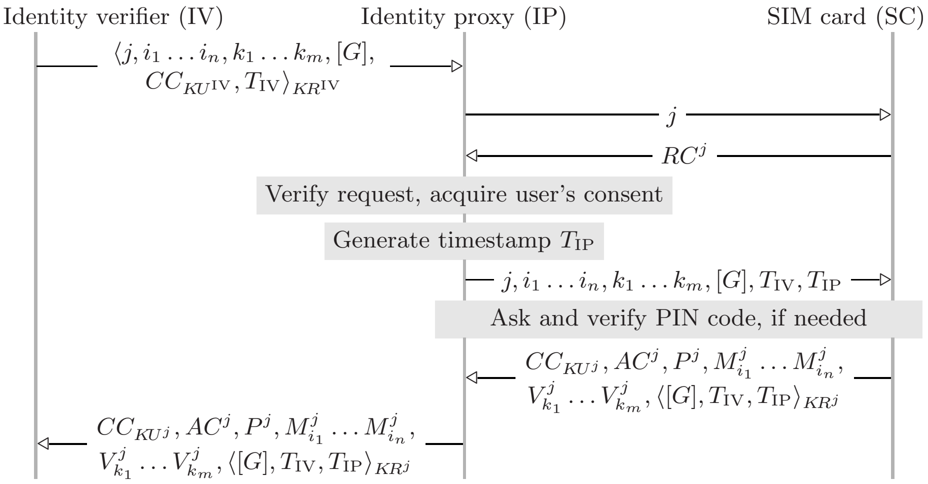

Identity verifier (IV)
Identity proxy (IP)
SIM card (SC)
$\langle j, i_1 \ldots i_n, k_1 \ldots k_m, [G], CC_{KU^{\mathrm{IV}}}, T_{\mathrm{IV}} \rangle_{KR^{\mathrm{IV}}}$
j
RC^j
Verify request, acquire user's consent
Generate timestamp T_{IP}
$j, i_1 \ldots i_n, k_1 \ldots k_m, [G], T_{\mathrm{IV}}, T_{\mathrm{IP}}$
Ask and verify PIN code, if needed
$CC_{KU^j}, AC^j, P^j, M^j_{i_1} \ldots M^j_{i_n}, V^j_{k_1} \ldots V^j_{k_m}, \langle [G], T_{\mathrm{IV}}, T_{\mathrm{IP}} \rangle_{KR^j}$
$CC_{KU^j}, AC^j, P^j, M^j_{i_1} \ldots M^j_{i_n}, V^j_{k_1} \ldots V^j_{k_m}, \langle [G], T_{\mathrm{IV}}, T_{\mathrm{IP}} \rangle_{KR^j}$

Fig. 2. Protocol used for acquiring identity proofs and digital signatures

not just providing information on her identity. For digital signatures, a longer (e.g., 6-digit) PIN code could be used, whereas for identity proofs a standard 4-digit PIN code is sufficient. The protocol used for acquiring identity proofs and digital signatures is outlined in Fig. 2.

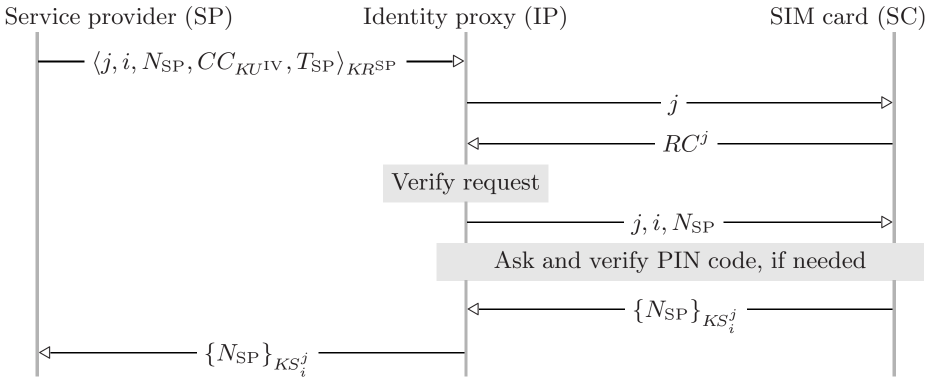
Note that requests with regard to several identity profiles can be easily combined in one transaction, with only a small modification of the protocol. Namely, the identity verifier sends a number of request tuples, all of which will be shown on screen. The identity proxy acquires user's consent and fetches several identity proofs from the SIM card. All proofs can be sent to the identity verifier in one bundle.

The issue of trust must be noted here. Normally, identity verifiers' certificates are validated against the trusted root certificate of the identity profile mentioned in the request. If proofs from several identity profiles are requested, the identity verifier may have to present several certificates, each chaining up to the corresponding root certificate in the identity profile.

2.4 Challenge Responses and One-Time Passwords

A service provider may also request encryption of a challenge with a secret key included in a profile. Using this functionality, the mobile phone can work as an authentication token for Internet banking services. Namely, it can emulate hardware tokens for two-factor authentication used in many network services and access control systems. Moreover, using proximity communication (e.g., NFC) electronic keys and other access tokens can be easily implemented.

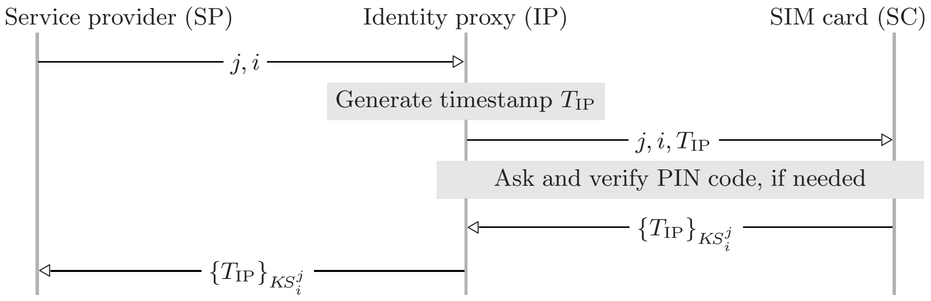
Normally, the challenge is supplied to the identity proxy by the service provider. For example, an electronic lock can submit a challenge to the identity proxy using NFC. However, it is also possible that the identity proxy uses the current time as



Service provider (SP)
Identity proxy (IP)
SIM card (SC)
$\langle j, i, N_{\mathrm{SP}}, CC_{KU^{\mathrm{IV}}}, T_{\mathrm{SP}} \rangle_{KR^{\mathrm{SP}}}$
j
RC^j
Verify request
j, i, N_{SP}
Ask and verify PIN code, if needed
$\{N_{\mathrm{SP}}\}_{KS_i^j}$
$\{N_{\mathrm{SP}}\}_{KS_i^j}$

Fig. 3. Challenge-response protocol

a challenge. The mobile phone can therefore be used as a secure authentication token, which shows a new one-time password every minute or half a minute. The protocol for one-time password generation is illustrated in Fig. 4. Note that the identity proxy does not necessarily need a connection to the service provider: the user can choose the profile j and specify the secret key type i himself, and submit the one-time password using a separate channel (e.g., enter it in a web browser on a computer). For acceptable usability, the one-time password in this case must be trimmed to only a few digits.)



Service provider (SP)
Identity proxy (IP)
SIM card (SC)
j, i
Generate timestamp T_{IP}
j, i, T_{IP}
Ask and verify PIN code, if needed
$\{T_{\mathrm{IP}}\}_{KS_i^j}$
$\{T_{\mathrm{IP}}\}_{KS_i^j}$

Fig. 4. One-time password generation

3 Infrastructure and Implementation Issues

Connection between the identity provider or verifier and the identity proxy can be established with either a proximity communication technology (e.g., NFC) or using a usual GPRS or WLAN connection. Note that NFC facilitates also peer-to-peer scenarios: for example, to exchange business cards participants can select their business card profiles and touch each other's phones.

At places where terminals with NFC are installed, the system can be used for fast customer identity checks, in loyalty applications and mobile payments. A terminal can also be used for loading profiles, if a profile issuer's private key is available at it, or if a certificate signing request can be processed quickly over a network.

An integral part of the certificate validation is checking whether the certificate has been revoked. For this, certificate revocation lists (CRL) or protocols like OCSP are used. Because for most proximity use cases we do not assume that the hand-held device is connected to a network, CRL-based revocation checks are by far the best option. We assume that the device can get connected to a network (e.g., through GPRS or WLAN) from time to time, so that the identity proxy can update the CRLs. For on-line use cases, and for identity verifiers' terminals, OCSP-based checks can be used.

Arguably, usability is one of the most important requirements for a mobile identity tool. If one has to go through the user interface jungle of a modern mobile phone for mere identification, such system would barely reach a wide audience. NFC phones that support Contactless Communications API [20] enable applications to get launched when a certain tag or reader is touched by the phone. This means that in order to provide information about herself the user has to touch with her phone the NFC reader attached to the verifier's terminal, check the request shown on the phone screen, and enter the PIN code, if needed.

The identity proxy can be implemented most easily in J2ME, as the API for application protocol data unit (APDU) messages exchange with the SIM card, Security and Trust Services (SATSA) [19], is available in many newer phone models, e.g., those based on Nokia Series 60 platform since 3rd Edition.

No operations with private keys are performed in the identity proxy. Signatures for identity proofs are generated on the SIM card; newer smart cards with cryptographic coprocessors are capable of computing 1024-bit RSA signatures in less than 200 ms. Operations with public keys are normally faster than operations with private keys. Nevertheless, as mobile phones do not normally have optimised cryptographic coprocessors, signature verification can take about 800 ms [22]. We note, however, that in identity verification scenarios the user normally needs a few seconds to study the verifier's request, and most cryptographic operations (except signature calculation) can be performed during this time.

Identity profiles are securely stored on the SIM card. EEPROM or Flash memory can be used for storing profile data; currently available high density SIM cards which provide tens to hundreds of megabytes of non-volatile Flash memory [23] have plethora of space for profiles. At the same time, requirements on random-access memory (RAM) for the card-side application are moderate: most operations can be performed within APDU buffer only.

4 Discussion

In this section we give an evaluation of our scheme, discuss its benefits and drawbacks, and provide an overview of security-related issues.

Identity profiles are securely stored on a tamper-resistant SIM card. Most currently used SIM cards have passed Common Criteria EAL4+ certification, thus providing security equivalent to bank cards. The level of protection provided by smart cards is sufficient for identity documents: many countries issue smart card based e-IDs. SIM cards has already been used in national mobile PKI infrastructures in several European countries (e.g., Finland, Slovenia and Turkey). In addition, the usability of the SIM card is somewhat better than that of built-in security elements in mobile phones. Indeed, the user can easily move her identities to another device simply by moving the SIM card.

Standardisation of the mobile phone as a payment card terminal is an issue that has to be investigated if the system is used for mobile payments. In particular, Europay-Mastercard-Visa (EMV) or similar standards might require the standardisation of mobile phone keypads for entering payment-related PINs. This issue can depend on a country where the system is deployed. For example, in Finland at least one bank group accepts national e-ID cards for the authentication of its customers. Among other uses, such authentication is accepted for confirming 3D Secure credit card payments, where the PIN is entered on a usual computer keyboard. In other countries more strict standards may be in force.

If the mobile phone is stolen or lost, the user has to inform her MNO, who will block the SIM card and place the profile-issuing certificate on the CRL. The operator can also automatically contact all known identity vendors and inform them about the blocked certificate, so that they can black-list identities associated with this certificate. Alternatively, the user can contact identity vendors directly and place her identity profiles on corresponding black lists.

The security of communication between the SIM card and the identity proxy is provided in the following way. To connect to an applet in the SIM card, the identity proxy must be digitally signed using the operator domain or a trusted third party domain certificate. In the SIM card there is an access control entry with a hash of this certificate. The implementation of SATSA on the mobile phone compares this hash with that of the certificate used for signing the J2ME applet. This security mechanism is in place to ensure that the J2ME applet has not been tampered with. Therefore, in order to steal and misuse identity profiles stored on the SIM card, an attacker has to develop a suitable malware, find a way to sign it by the operator or TTP, and install it on the user's mobile phone. Another option would be to find a serious vulnerability in the implementation of SATSA and exploit it.

An authentication token implemented using a mobile phone has certain benefits over usual security tokens. In particular, it is easier to provide time synchronisation with the service provider, because both the identity proxy and the service provider's authentication server can easily synchronise their clocks with the MNO. In many subscriptions this feature is enabled by default. Using mobile phones removes the need for separate hardware tokens, reducing expenditures on devices and logistical costs.

It is interesting to compare the security of our system with that of biometric passports. Earlier versions of passports were vulnerable to several attacks (see an overview in, e.g., [24]), including remote skimming of passport data due to only one-way authentication. In newer passports, a challenge-response mechanism was introduced for passport reader authentication. However, as the passport chip does not have a reliable source of time, it cannot reliably check whether the certificate presented by the reader is valid. This makes it possible for malicious terminals to use old expired certificates for compromised keys in some cases. In our system, the terminal certificate is checked by the identity proxy, which has a reliable source of time. In addition, the user can always see what information is being sent, and to whom, which increases usability, trust and security.

Electronic tickets with stored value, modified or rewritten by the terminal in every transaction cannot be implemented within our system. However, such schemes can be much easier implemented using NFC phones in the passive mode, as wireless smart card emulators. Also privacy issues can be reasonably taken into account [25]. At the same time, our system can be used for tickets valid for a certain period of time. Moreover, the validation of a ticket can be easily combined with the verification of the person's eligibility for a concession (e.g., belonging to a certain social group).

Clearly, the same architecture can be used on any other portable device (e.g., PDA), provided that it has a smart card slot, and an API for implementing programs that exchange APDUs with the smart card. Furthermore, the device must enable communication with terminals installed at identity verifiers' premises. From the security viewpoint, it is important to ensure that malware running on the device cannot access the smart card.

5 Related Work

Due to the personal nature of the mobile phone, it is seen as a suitable medium for storing personal information and credentials for access to various services. Indeed, researchers have been active in developing different ways for secure key provisioning and for storing identity information on the mobile phone.

Because the standard GSM authentication infrastructure is widely deployed, it is tempting to reuse it in new services. The 3GPP project has developed a framework called Generic Authentication Architecture (GAA) [26], which enables bootstrapping credentials from the cellular authentication key and reusing them for authorising access to other services. The system is easy to deploy, it is based on open standards, and it can be used in both on-line and proximity scenarios [27]. At the same time, the architecture is not open in the sense that service providers must sign agreements with the MNO in order to deploy GAA.

If the service provider has an agreement with the MNO, a system proposed in [28] can also be used. The work shows how a security agent comprised of two components, namely a J2ME applet and a SIM card applet, can be deployed on the mobile phone over-the-air. The system is universal and can be used by the MNO for deployment of arbitrary applications. This is an overkill for identity management, because if a service provider wishes to only install new credentials on the user's mobile phone, the operator must be contacted and a complicated two-phase security agent deployment procedure run. At the same time, the system can prove extremely useful for the installation of mobile identity management applications (e.g., the one presented in this paper).

The only fully open mobile credentials management system that we are aware of at the present time and that uses a secure environment for key provisioning and information storage is OnBoard Credentials platform [10]. In addition to the installation of credentials, the system can be used for provisioning confidential credential programs to a device. Sensitive information is stored within the M-Shield secure environment [9], which is a proprietary solution. In addition, at the time of writing, M-Shield has not passed a Common Criteria certification, which might be a problem for deploying bank-issued credentials. For most of applications, however, the platform is very useful and flexible. At the moment, credentials cannot be easily moved from a phone to phone, but there are no apparent obstacles for adding this feature in the future.

The idea of storing identity information and a keyring in the SIM was proposed in [11]. The architecture is similar to the one presented in this paper, however, the authors provided only a vague specification of protocols used in their system.

6 Conclusions

This paper described an architecture and protocols for a mobile phone based identity management system. The system is open in the sense that *any* identity or service provider can load and request verification of identity profiles (including pseudonyms and keys). The identity tool can also be used for payment applications and customer authentication in online banking and other similar services. In proximity scenarios, the tool supports biometric authentication of the users. The architecture is based on currently available handset technology.

We have set strict security requirements for our electronic ID, namely, minimisation of data collection (the verifier gets the minimum necessary amount of personal details), user consent (all actions regarding personal details have to be approved by the user), authenticity and integrity (all parties are strongly authenticated and data integrity is guaranteed), and easiness of revocation (in case of compromised credentials, it is easy to revoke those credentials).

Acknowledgement. The author thanks Marko Hassinen, Elena Trichina, and anonymous reviewers for constructive comments on earlier versions of the paper.

References

1. Liberty Alliance Project: Liberty Alliance ID-FF 1.2 specifications (December 2007), `http://www.projectliberty.org/liberty/specifications__1`
2. OpenID Foundation: OpenID authentication 2.0 - final (December 5 2007), `http://openid.net/specs/openid-authentication-2_0.html`
3. Scavo, T., Cantor, S.: Shibboleth architecture: Technical overview, working draft 02 (June 2005), `http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-arch-latest.pdf`

4. Chappell, D.: Introducing Windows CardSpace (April 2006) (accessed 8.4.2008), http://msdn2.microsoft.com/en-us/library/aa480189.aspx
5. Müller, G., Wohlgemuth, S.: Study on mobile identity management. FIDIS - Future of Identity in the Information Society, deliverable 3.3 (May 2005)
6. Wohlgemuth, S., Jendricke, U., Gerd tom Markotten, D., Dorner, F., Müller, G.: Sicherheit und benutzbarkeit durch identitätsmanagement. In: Spath, D., Haasis, K. (eds.) Aktuelle Trends in der Softwareforschung - Tagungsband zum doIT-Forschungstag, Stuttgart, pp. 241–260. IRB Verlag (2003)
7. Trusted Computing Group: TCG mobile trusted module specification, version 1.0, revision 1. TCG published (June 12, 2007)
8. Alves, T., Felton, D.: TrustZone: Integrated hardware and software security. ARM white paper (July 2004), http://www.arm.com/pdfs/TZ_Whitepaper.pdf.
9. Srage, J., Azema, J.: M-shield^{TM} mobile security technology—making wireless secure. Texas Instruments white paper (2005), http://www.ti.com/m-shield/.
10. Ekberg, J.E., Asokan, N., Kostiainen, K., Eronen, P.: OnBoard Credentials platform design and implementation. Technical report NRC-TR-2008-001, Nokia Research Center (January 2008)
11. Kalman, G., Noll, J.: SIM as secure key storage in communication networks. In: Dini, C., Dohler, M., Eltoweissy, M., Rui, T., Skouby, K.E. (eds.) Wireless and Mobile Communications, 2007. ICWMC 2007. Third International Conference on, Gosier, Guadeloupe, March 2007, p. 55. IEEE Computer Society Press, Los Alamitos (2007)
12. The Royal Academy of Engineering: Dilemmas of privacy and surveillance : Challenges of technological change. The Royal Academy of Engineering, 29 Great Peter Street, London, SW1P 3LW (March 2007)
13. Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. The MIT Press, Cambridge (2000)
14. Li, J., Li, N.: A construction for general and efficient oblivious commitment based envelope protocols. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 122–138. Springer, Heidelberg (2006)
15. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
16. Camenisch, J., Herreweghen, E.V.: Design and implementation of the idemix anonymous credential system. In: CCS 2002: Proceedings of the 9th ACM conference on Computer and communications security, pp. 21–30. ACM, New York (2002)
17. McEvoy, N.A.: e-ID as a public utility. Consult Hyperion, Guilford, UK (May 2007), http://www.chyp.com
18. Pfitzmann, A., Hansen, M.: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. version v0.29 (July 31, 2007)
19. Java Community Process: Security and Trust Services API (SATSA) for Java^{TM}2 Platform, Micro Edition, v. 1.0. Sun Microsystems, Inc., Santa Clara, CA, USA. (July 17, 2004), http://www.jcp.org/en/jsr/detail?id=177.
20. Java Community Process: Contactless Communication API, JSR 257, v. 1.0. Nokia Corporation, Espoo, Finland. (October 2, 2006), http://www.jcp.org/en/jsr/detail?id=257
21. Farrell, S., Housley, R.: An internet attribute certificate profile for authorization. Network Working Group, Request for Comments 3281 (April 2002)
22. Lee, Y., Lee, J., Song, J.: Design and implementation of wireless PKI technology suitable for mobile phone in mobile-commerce. Computer Communications 30(4), 893–903 (February 2007)

23. Handschuh, H., Trichina, E.: High density smart cards: New security challenges and applications. In: Pohlmann, N., Reimer, H., Schneider, W. (eds.) ISSE/SECURE 2007 Securing Electronic Business Processes, Vieweg, pp. 251–259 (2007)
24. FIDIS - Future of Identity in the Information Society: Budapest declaration on machine readable travel documents (MRTDs) (September 2006)(accessed 8.4.2008), `http://www.fidis.net/fileadmin/fidis/press/budapest_declaration_on_MRTD.en.20061106.pdf`
25. Desmedt, Y.: Position statement in RFID S&P panel: From relative security to perceived secure. In: Dietrich, S., Dhamija, R. (eds.) Financial Cryptography and Data Security, 11th International Conference, FC 2007. LNCS, vol. 4886, pp. 53–56. Springer, Heidelberg (2008)
26. 3rd Generation Partnership Project (3GPP): TS 33.221 Generic Authentication Architecture (GAA); support for subscriber certificates (December 2007), `http://www.3gpp.org/ftp/Specs/html-info/33221.htm`
27. Laitinen, P., Ginzboorg, P., Asokan, N., Holtmanns, S., Niemi, V.: Extending cellular authentication as a service. In: The First IEE International Conference on Commercialising Technology and Innovation, September 14-15, 2005, IEEE, Los Alamitos (2005)
28. Sirett, W., MacDonald, J., Mayes, K., Markantonakis, K.: Design, installation and execution of a security agent for mobile stations. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 1–15. Springer, Heidelberg (2006)
29. Santesson, S., Polk, W., Barzin, P., Nystrom, M.: Internet X.509 public key infrastructure qualified certificates profile. Network Working Group, Request for Comments 3039 (January 2001)

A Terminology

We recall here some of the Pfitzmann-Hansen terminology [18] and extend it with a few other notions regarding a person's identity.

Biometric. A biometric pattern of a person. Examples of biometric are fingerprints, iris scan, DNA; the least intrusive (in modern society) biometrics are voice and photo image.

Biometric identifier. A cryptographic hash value of a biometric pattern. Preimage resistance (both first and second) of the hash function is important in our application.

Pseudonym. An identifier of a person other than one of the person's real names. We use mainly biometric identifiers in our system, but also other types of identifiers are supported.

Attribute. A quality or characteristic of a person. Attributes are the building blocks of a person's identity. Examples are surname, blood group, or employer's name. An attribute has a code and a value.

Attribute code. An identifier of an attribute. A system for naming the attribute codes must be chosen; we use standard X.509 attribute codes [29].

Identity. A subset of attributes of a person which sufficiently identifies this person within any set of people. Often it characterises only a particular aspect of the person, i.e., a role, position or status of the person in a given context. In this case only the identifiability of the role, position or status is needed, and the identifiability of the person is not required. In the text, "identity" refers to partial identity (see below), unless indicated otherwise.

Complete identity. The union of all attributes of all identities of this person.

Partial identity. A subset of attributes of the complete identity. Individuals use different partial identities in different situations.

Identity issuer or identity vendor. An organisation or company determining and attesting the attributes of an identity. Identity issuers can only issue partial identities (we assume that there are no complete identity issuers).

Identity profile. A partial identity along with certificates of its integrity and authenticity, and other partial identity related information, such as keys and certificates.

Identity verifier. A party to which a person reveals values of certain attributes from a partial identity or a number of partial identities, and proves their authenticity and integrity.

Identity management means managing various partial identities of a person, i.e., administration of identity attributes including the development and choice of the partial identity and pseudonym to be (re-)used in a specific context or role. Naturally, there are three parties in identity management: a person, an identity issuer, and an identity verifier.

Minimal identity. Minimal set of attributes and pseudonyms required by an identity verifier to identify a person or to correctly establish the person's role, position or status.

Data minimisation. This property means that in the release of personal data only minimal identities are used; for released data, as much unlinkability as possible is preserved.

Identity token. A storage device for one or more partial identities, used by a person to reveal and prove her identity. Examples of identity tokens are usual identity and loyalty cards.

Identity tool. An interactive device for storing and managing one or more partial identities, used by a person to reveal and prove her identity. Interactivity means that the selection of the used partial identity is possible.

Privacy-enhancing identity tool. An identity tool that can be used in identity management with data minimisation. A privacy-enhancing identity tool (a) informs the person about the set of information requested by the identity verifier, and (b) allows the person to comply with the request or reject (or modify) it.

PEACHES and Peers

Massimiliano Pala and Sean W. Smith

Computer Science Department, Dartmouth College
6211 Sudikoff Laboratory, Hanover, NH 03755, US
{pala,sws}@cs.dartmouth.edu

Abstract. How to distribute resource locators is a fundamental problem in PKI. Our *PKI Resource Query Protocol (PRQP)*, recently presented at IETF, provides a standard method to *query* for PKI resources locators. However the *distribution* of locators across PKIs is still an unsolved problem. In this paper, we propose an extension to PRQP in order to distribute PRQP messages over a Peer-to-Peer (P2P) network. In this work, we combine PRQP with *Distributed Hash Tables (DHTs)* to efficiently distribute contents over a dynamic P2P overlay network. In particular we present the *PEACH* protocol and a *PEACH Enabled System (PEACHES)* which are specifically targeted toward solving the PKI resources discovery problem. Our work enhances interoperability between existing PKIs and allows for easy configuration of applications, thus augmenting usability of PKI technology.

1 Introduction and Motivation

Public key cryptography has become, in many environments, a fundamental building block for authentication. Many applications already support the usage of *Public Key Certificates (PKCs)*—e.g. browsers, mail user agents, web-based applications, etc. PKCs can be requested and obtained from different *Certification Authorities (CAs)* which may live within a large infrastructure or may be deployed as isolated entities. However, locating services and data repositories related to a CA is still an open problem. The lack of a standard way to distribute resource locators to applications heavily (and negatively) impacts interoperability between PKIs and usability of applications.

In order to manage and extend trust among CAs and PKIs, different trust models have been studied and deployed.

Regardless of what the adopted trust model is, achieving interoperability between deployed infrastructure is very difficult and the management of the authentication infrastructure can be frustrating. One of the main reasons for these interoperability issues is the difficulty in discovering resources related to a Certification Authority.

An example of an interoperability nightmare is certificate validation. In order to grant access to its service, an application needs to verify that a certificate is still valid by, at the very least, retrieving the revocation information provided by the issuing CA.

This information is usually provided by the issuing CA by means of a *Certificate Revocation List (CRL)* or by an *OCSP* server. If no previous configuration exists at the application level, finding where the revocation information is available can be very difficult. As discussed in our earlier work [1], current solutions are not capable of solving the problem even in controlled environments like Virtual Organizations and Computational Grids.

Motivated by how this lack of resource pointers impacts many usability aspects in PKIs, this paper extends the PKI Resource Query Protocol (PRQP) in a peer-to-peer network in order to provide an efficient Discovery service for PKI resources—thus enhancing practical interoperability and usability of currently deployed PKIs.

The core contribution of this work is PEACH, a scalable system for PKI resources lookup.

2 Background

In order to provide a distributed and interoperable method to provide applications with pointers to PKI resources, we combine two existing technologies: the PKI Resource Query Protocol (PRQP) and a distributed hash table routing protocol, based on a modified version of Chord [2]. In this section, we provide the reader with a description of the background knowledge needed to understand our work.

2.1 The PKI Resource Query Protocol

PRQP is a simple query-response protocol designed to provide applications with the ability to query an entity for locators of specific services associated with a CA. In PKIs, the CA can grant other entities the authority to provide specific information to clients. An example of this is OCSP, where the CA delegates to the OCSP server the authority to provide information about the revocation status of its certificates. Similarly, in PRQP the CA identifies an entity to be authoritative for PRQP answers. The identified entity is called the *Resource Query Authority (RQA).* By querying the RQA, an application can discover the location (URL) of a service or of a repository associated with a specific CA.

PRQP identifies two different trust model for the RQA. In the first model, the RQA is directly designated by the CA by having the CA issue a certificate to the RQA with a specific value set in the `extendedKeyUsage` extension.

In the second model, the RQA acts as a *PRQP Trusted Authority (PTA)* for a set of users—e.g., users in an enterprise environment. When operating as a PTA, the RQA may provide responses about multiple CAs, without the need to have been directly certified by them. To operate as such, PRQP requires that a specific extension, i.e. `prqpTrustedAuthority`, should be present in RQA's certificate and its value should be set to `TRUE`.

A full description of the protocol and its details is provided in our RFC [3].

2.2 Distributed Systems: The Peer-2-Peer Revolution

One open problem in PRQP is how to find information about CAs without prior knowledge of the associated RQA. To solve this problem, we now provide a distributed discovery system for PKI resources.

However, designing, implementing and debugging a large distributed system is a notoriously difficult task. Consequently much of the current effort has been spent on easing the construction of such systems.

The simplest organizational model for distributed systems is the *Client/Server* model. This model is well-known, powerful and reliable. In this configuration, the server is a data source while the client is a data consumer. Simple examples that make use of this model are Web Services and FTP. The Client/Server model, however, presents some limitations. For one thing, scalability is hard to achieve. For another, the model presents a single point of failure. The model also leaves unused resources at the network edges.

Peer-to-Peer (P2P) systems try to address these limitations by leveraging collaborations among all the participating parties (peers). Within this paradigm, all nodes are both clients and server: any node can provide and consume data. Several characteristics make P2P systems very interesting for data or services distribution over the Internet. They can implement an efficient use of resources by equally distributing usage of bandwidth, storage and processing power at every node. Most importantly, P2P systems are scalable. Napster, Gnutella and Kazaa are popular examples of first-generation P2P applications known to be efficient in data lookup and distribution. These systems use different approaches to provide a lookup service about the data provided by the participating peers. Napster implements a centralized search service where a single server keeps track of the location of the shared contents. On the opposite side is Gnutella; in this type of network, search is implemented by recursively asking the neighbors for files of interest. The search goes on till a *Time To Live (TTL)* limit is reached. Other systems like Kazaa or Skype use a hybrid model where super-peers act as local search hubs. Super-nodes are automatically chosen by the system based on their capacities in terms of storage, bandwith and availability.

2.3 Distributed Hash Tables

The *second generation of P2P overlay networks* provide more advanced features: they are self-organizing, load-balanced and fault tolerant. A major difference over an unstructured P2P system is that these second-generation systems also guarantee that the number of hops needed to answer a query is predictable. These systems are based on *Distributed Hash Tables (DHTs)*. DHTs are a distributed version of a hash table data structure, which stores (*key*, *value*) pairs, where the *key* is the identifier of the resource while the *value* can be the file contents. DHTs provide an efficient way to look up data and objects in a P2P network. Each node in the network is assigned a *node-id* and is responsible for only a subset of (*key*, *value*) pairs. By using the DHT interface, these systems are capable of finding a node responsible for a given *key* and efficiently routing the specific request (e.g., *insert*, *lookup* or *delete*) to this node. There are several DHT routing protocols, often referred to as *P2P routing substrates* or *P2P overlay networks.* Chord [4,5] implements a concept of a circular address space and provides simple operations—$insert(key, value)$, $lookup(key)$, $update(key, value)$, $join(n)$, and $leave()$—to maintain and update the network routing tables. Pastry [6] implements a similar interface to Chord; however it considers network locality in order to minimize hops that the messages travel. CAN [7] is based on a "d-dimensional" Cartesian coordinate space on a d-torus. In CAN, each node owns a distinct one in the space and each key hashes to a point in the space. Other examples of DHTs include Tapestry [8], Kademlia [9], and P-Grid [10].

3 The PEACH Protocol

This section describes our *PKI Easy Auto-discovery Collaborative Hash-table (PEACH)* protocol. Our protocol specifies how to find the location of a specific RQA, how RQAs join the system, and how to update the system in case a node leaves.

The PEACH protocol is derived from the Chord protocol. Although there are many similarities with Chord, we designed PEACH in order to leverage unique features of PRQP to provide support for a P2P overlay network specifically for RQAs.

In particular, PEACH differs from the Chord protocol in two main aspects. First, we implement a completely new method to assign node identifiers to peers. As a consequence, the lookup protocol directly provides the requesting entity with the address of the authoritative RQA for a requested CA. Moreover, because of this new idea, PEACH is simpler than Chord in that it does not require any $insert()$ or $get()$ operations to locate the needed data—that is, to recover the network addresses of RQAs.

The second change is related to the way nodes join the PEACH network. In fact, the new $join()$ operation allows only authenticated RQAs to join the network.

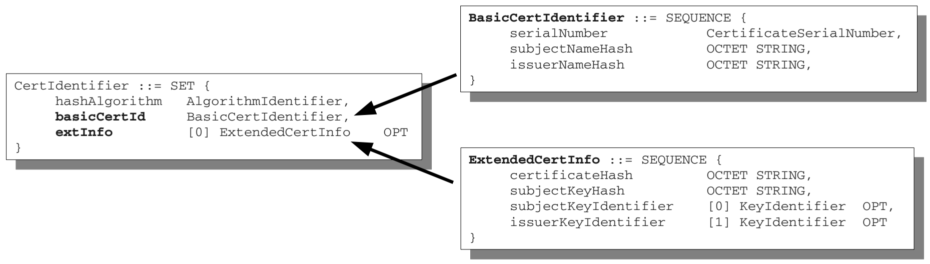
3.1 Overview

Similarly to Chord, PEACH uses a standard hash function to assign node identifiers. Differently from Chord, we use a PKI-specific method instead of network addresses to assign node identifiers.

In PEACH, the participating peers are RQAs. We assume that each RQA has a (cryptographic) key pair that has already being certified by its CA. Specific certificate contents—that is, the `extendedKeyUsage` field—allow the RQA to provide responses about resources related to that particular CA.

The basic idea is to leverage the direct link between a CA and its RA by building the node identifiers by using the CA's certificate fingerprint[1]. Since the CA's fingerprint is often used as part of a PRQP request, it is a perfect candidate

[1] The *fingerprint* of a certificate is calculated by computing a cryptographic hash over the DER- encoded certificate.



```
CertIdentifier ::= SET {
     hashAlgorithm   AlgorithmIdentifier,
     basicCertId     BasicCertIdentifier,
     extInfo         [0] ExtendedCertInfo    OPT
}
```

```
BasicCertIdentifier ::= SEQUENCE {
     serialNumber            CertificateSerialNumber,
     subjectNameHash         OCTET STRING,
     issuerNameHash          OCTET STRING,
}
```

```
ExtendedCertInfo ::= SEQUENCE {
     certificateHash         OCTET STRING,
     subjectKeyHash          OCTET STRING,
     subjectKeyIdentifier    [0] KeyIdentifier  OPT,
     issuerKeyIdentifier     [1] KeyIdentifier  OPT
}
```

Fig. 1. PRQP CertificateIdentifier data structure

for a node identifier. Moreover, this allows a node to provide authentication information that may be used by the *successor* node to verify that the joining RQA is authoritative for a specific CA.

This method of building node identifiers frees us from the requirement of having to store any value on the participating peers. Therefore, when a peer joins or leaves the network no data need be moved (or copied) among nodes and there is no need to implement operations for data storing/retireving to/from the network (e.g., *insert*() and *get*()). This increases the network reliability and lowers the number and load of operations needed in order to manage *join*() and *leave*() operations.

In order to guarantee that the lookup of nodes takes place within $O(\log N)$ steps, a list of m pointers is maintained at each node. This list of pointers is the equivalent of the fingers table in Chord and has the same purpose.

3.2 Certificate-Based Node Identifiers

In PEACH, we use a cryptographic hash function to generate node identifiers. In particular, to maintain compatibility with current PRQP specification, we use the same hash function that is implemented in PRQP. The current PRQP Internet draft uses SHA-1 [11] to provide CAs certificates identifiers. PEACH does not depend on the hash function itself, therefore it can be easily updated to future functions, like SHA-256, to build identifiers.

Although it is possible to use different hash algorithms to build identifiers, is must be noted that the chosen algorithm has to be shared across the whole PEACH network.

Because PEACH is specifically designed to be used in conjunction with PRQP, we leverage some data structures already present in PRQP to build PEACH node identifiers. In particular, PRQP allows a client to request pointers related to a CA by providing the `CertIdentifier` within a request. This data structure allows for several ways to identify the certificate of the CA of interest. Figure 1 provides the ASN.1 description of the CertIdentifier data structure. The `BasicCertIdentifier` carries the information needed to identify a certificate: that is, the serial number, the hash of the subject name and the hash of the issuer name. The optional `ExtendedCertIdentifier` field bears more detailed information about the CA's certificate. In PRQP, this field is optional as the client could ask for pointers related to a CA without having the possibility to compute the fingerprint of the CA's certificate.

In PEACH, each peer that joins the network is assigned with a node identifier which corresponds to the hash of its CA's certificate. In other words, when an RQA joins the network, it uses the fingerprint of the certificate of the CA that issued the RQA's certificate.

In PRQP, an RQA can be issued certificates from more than one CA. This enables the RQA to be authoritative for each of those CAs. To accommodate for this possibility, the RQA is assigned multiple network identifiers as described in Section 3.5.

The hash function generates values of m bits (which for SHA-1 is 160, while for SHA-256 is 256). Therefore, the node identifiers can be seen as laid on an ordered circular structure modulo 2^m. Hence, the possible values for node identifiers range from 0 to $2^m - 1$.

3.3 The Lookup Operation

The lookup operation in PEACH is derived from the one implemented in Chord. To limit the number of hops needed to find if a node is present on the network, each node keeps a table of entries where data about nodes present on the network are stored. This table is called the *pointers* table. It is important to notice that PEACH does not need the *pointers* table in order to function properly. However the *pointers* table reduces the complexity of the lookup operation from $O(n)$ to $O(\log N)$.

The number of entries in the *pointers* table is equal to the number of bits (m) in the node identifiers. If the SHA-1 hash function is used, the *pointers* table has $m = 160$ entries. The contents of each entry in the table is provided in Figure 2. The table is ordered on the *Entry ID* field. The Entry ID values for node n range $(id_n + 2^0) \mod 2^m$ (for entry 0) to $(id_n + 2^{m-1}) \mod 2^m$ value (for entry i). As in Chord, the concept behind the *pointers* table is to divide the PEACH network space into progressively growing slices.

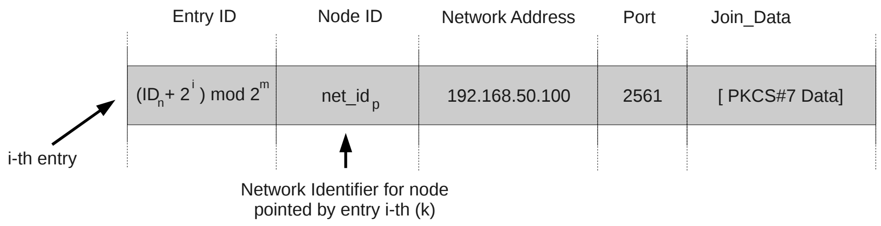


Fig. 2. i-th entry in the pointers table of node n

For example, given the node n and the ith entry in its pointers table, be x_i^n such as:

$$x_i^n = (id_n + 2^i) \mod 2^m$$

then the node-identifier space related to this entry is:

$$\gamma_i^n = [x_i^n, x_{i+1}^n)$$

Figure 3 shows two consecutive slices on the PEACH network for node n. To perform a lookup of a node address, the application would execute a *find_node*(). At first, the node n will perform a *lookup_table* (i) operation. This would look at the local pointers table and return the closest node in the network that *is equal* or *precedes* the node we are searching for. Be node k the closes match. If a perfect match is not found in the local pointers table, then a *find_node_ex*(k, i) operation is performed which asks node k to perform a *find_node*(i). In other words, the query recursively traverses the PEACH network and returns: (a) the address of the matching node (if it exists on the network) or (b) the closest preceeding match that is present on the network. The pseudocodes for the *lookup_table*() and *find_node*() operations are reported in Figure 4.

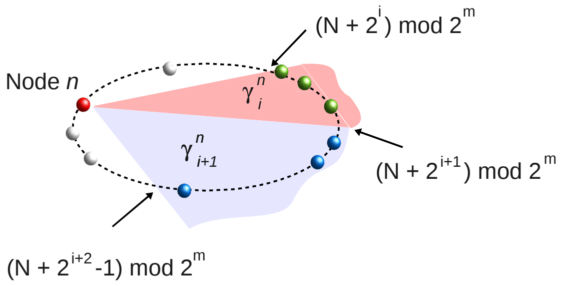


Fig. 3. γ_i^n and γ_{i+1}^n network identifier spaces. The size of the space doubles at each step.

The lookup function is different from the Chord protocol in that it looks for node identifiers, not for a key space assigned to a node.

The lookup function in a Chord-based network is generally used by *insert*() or *get*() operations to retrieve or store content to/from the network. Therefore, when searching for a particular key, a valid result (i.e., the node identifier that is responsible for the key space in which the searched key is in) is always returned on a lookup operation. In fact, the node whose identifier is the next on the Chord ring is the one responsible for storing all the values related to the specific key space.

In PEACH, instead, if no exact match is found on the network, the preceding closest node (RQA) is returned. When this happens, the lookup operation fails, meaning that the RQA that is authoritative for the particular CA is not present on the network.

```
function lookup_table ($id$, $\gamma$):
        for $j \Leftarrow m-1$ downto 0
            if ( $pointers[\ j\ ].id \in \gamma$ ) and
                                    ( $pointers[\ j\ ].id < id$ )
                // We found the closest match on the pointers table
                return pointers[ $j$ ]
            end if
        done
        // No suitable pointer found in the table
        return ( $null$ )


function find_node($id$, $\gamma$):
        if ( $id == self.id$ )
            return ( $self$ )
        end if
        $ret \Leftarrow$ lookup_table ( $id, \gamma$ )
        if ( $ret == null$ )
            // Current node is the closest on the network
            return ( $self$ )
        end if
        // Propagate the search with reduced space
        $\gamma \Leftarrow \gamma$ - $\|self - ret\|$
        return ( find_node_ex ( $ret, id, \gamma$ ) )
```

Fig. 4. Pseudocode for finding a node with the *id* indentifier. The closest match on the local table is found by using the **lookup_table**() function. The peer will then ask closest matching node (*ret*) to perform a lookup by using the **find_node_ex** () function. All the search operations are constrained on a seach space γ.

3.4 The Join Operation

One of the most important operations in PEACH is *join*(). In order to be able to find an authoritative RQA, the network has to be made aware of its availability. Therefore when an RQA wants to provide services to the network via PEACH it has to *join*() the network.

In PEACH , each node maintains, at minimum, the information about its *successor* and its *predecessor* on the network. For the network to operate properly, this information must be up to date. This is achieved via the *join*() and *leave*() operations. When a node joins the network it performs the following operations:

a. connects to the one of the available nodes
b. finds its *successor* on the PEACH network
c. informs its *predecessor* and its *successor* of its presence on the network
d. (optionally) builds the list of pointers to other nodes

In order to perform step (a.), applications need to be instructed on how to reach one node that participates in the network. PEACH does not specify how to provide applications with a list of active nodes to contact at startup. However, it is possible to provide a pointer to active nodes by using different methods, e.g. DNS SRV records [12], DHCP extensions [13], or by scanning the local network for active nodes. Another very popular method is to simply embed a list of URLs that applications can use and update upon startup (e.g., root DNS server addresses are usually embedded into DNS server software like Bind [14]). Currently, in our test environment we use the latter approach. Future plans are to set up some stable nodes that RQAs can use as "entry points" to the PEACH network.

To find its successor on the network (step b.), the joining RQA performs a *find_node_ex*(). Be n the joining RQA, and k the "entry point".

The *find_node_ex*() function asks node k to perform a lookup. The *successor* of node n is found by simply using the *find_node_ex*() on node k with $id_{n+1} = (id_n + 1) \mod 2^m$. Be p the returned node. If the returned value is a perfect match, the successor of node n is found. If the returned value is not a perfect match, the returned value is the closest preceding node. In this case, to find its successor, the node n contacts *ret* and asks it for his successor. Ultimately, the node p will return the first entry in its pointers table to node n.

In order to find its *predecessor* on the network (step c.), the node n simply performs a lookup for node whose id is id_{n-1}. More details on format of the data packet that are exchanged over the PEACH network is discussed in Section 4.

Now, the joining RQA asks p to be considered as the new *predecessor*. To do so, the n node sends to p a PKCS#7 [15] signed object. This object carries the details about the n's network address as the data payload. The object type used for the exchanged message is `SignedData`. The RQA's certificate and the certificate of the issuing CA (the one for whom the RQA is authoritative) are embedded in the `certificates` field.

The receiving node, before updating the details about its *predecessor* but after joining the PEACH network, verifies that the RQA's certificate is valid and that it has been issued by the CA for which the RQA will be registered to be authoritative, In order to do so, it verifies that the signature on the PKCS#7 object is valid and that is has been signed with the private key that corresponds to the public key present in the RQA's certificate. Secondly it verifies that the RQA's certificate is issued by the presented CA's certificate (by verifying its signature against the public key of the CA's certificate). As a last step, the receiving node calculates the CA's certificate fingerprint and assigns it as the identifier for node n as its predecessor. The new id, together with the network address data, is stored on the receiving node. Differently from Chord where only the successor of a node is informed when a new node joins the network, in PEACH the joining peer pushes its information to its predecessor as well. Also differently from Chord, as a result of our *join*() operation, participating nodes do not need to execute any *update*() function to discover new nodes.

Once an RQA joins the PEACH network, it is required to maintain open one socket to its *predecessor* and one socket to its *successor*. This enables nodes to immediately be aware of a node leaving the network without the need to run any *update*() operation.

3.5 Multiple Joins

PRQP allows a Resource Query Authority to be authoritative for multiple CAs. To be able to provide PRQP responses for multiple CA, an RQA needs to be registered with multiple network identifiers.

To be assigned multiple node identifiers, the joining RQA performs multiple *join*() on the network. Let n be the number of certificate the RQA possesses. The set of its certificates (ϕ) can be expressed as:

$$\phi = \{x_1, x_2, \ldots, x_n\}$$

Let θ be the set of network identifiers related to the joining RQA:

$$\theta = \{y_1, y_2, \ldots, y_n\}$$

where:

$$\forall i \in [1, 2, \ \ldots\ , n], \quad \exists x_i, y_i \ : \ x_i \in \phi \quad \vee \quad y_i \in \theta \Rightarrow y_i = H(x_i)$$

For each x_i the peer is authoritative for, the RQA has a different network identifier y_i which is based on the CA's certificate fingerprint. For each of these identifiers, the joining peers performs *find_node*() to find successor in the network ring and proceeds to register itself in the right position.

3.6 Other Operations

In this section we provide some considerations on the efficiency of PEACH. We especially focus our attention on the network maintainance operations.

Leaving the Network. PEACH does not require participating peers to execute any *leave*() operation. Because each peer is required to maintain open two TCP connections toward its *successor* and its *predecessor*, when the node leaves the network, the required TCP connections are dropped. Hence both its *predecessor* and its *successor* would be aware of the leaving operation right away.

Creating the list of pointers. After an RQA has successfully joined the network, it needs to populate its *pointers* table. To do so, the joining peer queries the network for the entries in the *pointers* table. The algorithm is reported in Figure 5. Ultimately, the list of pointers is built by performing lookups for each entry id, that is:

$$entry_{id} = (id_n + 2^i) \mod 2^m$$

where the table size is m, and the peer network identifier is id_n. The returned value is stored in the entry only if it is in the space identified by the ith entry (γ_i^n). For instance, let the space of identifiers be 3 bits, let x be the joining node

```
function **update_table ()**:
        for ( $i = 1$; $i < m$; $i = i + 1$ )
            $ret$ = **find_node** ( $pointers[i].id$ )
            **if** $ret.id \in [pointers[i].id, pointers[i+1].id)$
                $pointers[i] \Leftarrow ret$
            **end if**
        **done**
```

Fig. 5. Pseudocode for building the table of pointers

and let 1 be its *id*. The nodes present in the network have identifiers 0, 2, and 3. The last entry in the *x pointers* table is:

$$\begin{aligned} x_{id} &= (1 + 2^{3-1}) \mod 2^3 \\ &= 5 \end{aligned}$$

Now, let k be the closest previous match on the network, and let 3 be its network identifier. The possible results of the performed search are:

$$res = \begin{cases} 5 & \text{if } x_{id} \ \exists \ \text{ on the network,} \\ 3 & \text{if } x_{id} \ \nexists \ \text{ on the network,} \end{cases}$$

Because no node with 5 as its identifier is present on the network, node k is returned instead. As the result should be stored only if it falls into the ith key space, the returned value is discarded.

Maintaining the list of pointers. When performing a *find_node*() operation, it may happen that an entry in the *pointers* table points to a node that is no longer available on the network. In this case, if the connection to the entry fails, the entry is simply removed from the table and the lookup operation is resumed. Occasionally a node may update its list of pointers in order to leverage the presence of new nodes in the network.

4 PEACHES Details

In this section we provide considerations about the implementation of the algorithm and an evaluation based on a PEACH simulator.

4.1 Network Communication

While building a PEACH enabled system, we also defined the format of exchanged messages over the network. To minimize the impact of the message format, we opted for a simple binary format.

The PEACH data packets are depicted in Figure 6. The format is consistent across the different commands that nodes exchange. In particular, each data packet has the following fields:

– command code
– packet length
– payload

The *packet length* is used to identify the lenght of the payload and it is 4 bytes long (type uint32_t). The *command code* is 4 bytes long as well and it specifies the action to be performed on the target node or the return code. The identified opt codes and their description are reported in Table 1.

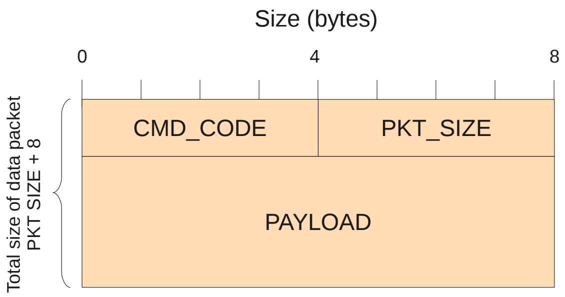


Fig. 6. Basic structure of data packets exchanged on the PEACH network. The payload depends on the specific command.

Table 1. PEACH opt codes values and description

Command Name	Code	Description
CMD_ERROR	0x200 + 0	General Error
CMD_SUCCESS	0x200 + 1	Cmd Successful
CMD_GET_NODE_INFO	0x500 + 0	Get node information
CMD_GET_NODE_SUCCESSOR	0x500 + 1	Get node successor
CMD_GET_NODE_PREDECESSOR	0x500 + 2	Get node predecessor
CMD_UPDATE_PREDECESSOR	0x600 + 1	Update predecessor info
CMD_UPDATE_SUCCESSOR	0x600 + 2	Update successor info
CMD_LOOKUP_NODE	0x600 + 2	Perform a lookup

4.2 The PEACH Simulator

To analyze the feasibility of building a PEACH system, we developed a simulator that implements the protocol's operations. The current version of the simulator is written in PERL. To implement the cryptographic functionalities, we use a command-line based tool that is built on top of LibPKI [16]. This tool loads keys and certificates and provides the simulator with an easy-to-use PKCS#7 object generator.

Because the focus of the simulation is to test the feasibility of the PEACH deployment, we concentrated our attention on the routing of the data in PEACH. Because our work targets PKIs, we reasonably think (based on our knowledge

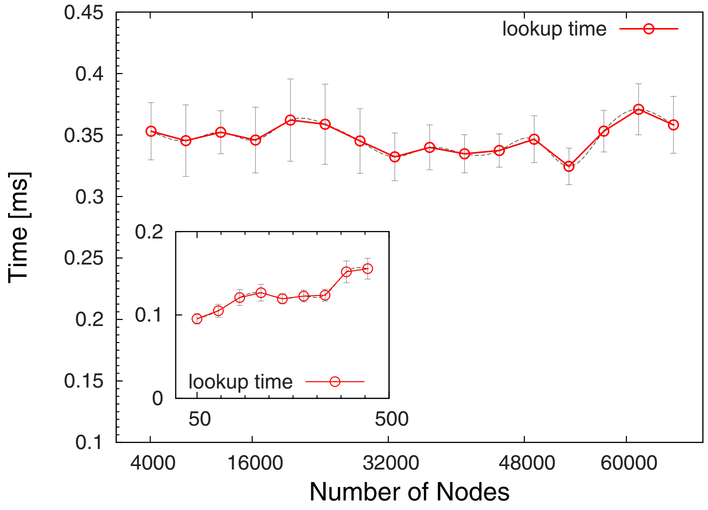


Fig. 7. Lookup times in the PEACH network simulator

of deployed PKIs) that the number of participating RQAs for a PEACH based system could reasonably set between few hundreds to few thousands of peers. We have been able to successfully simulate systems with more than 65000 nodes on a 2.4Ghz Core-Duo laptop equipped with 2GB of memory. Moreover we measured the performance of the *lookup* operation in PEACH. The results are reported in Figure 7. We simulated PEACH networks with an increasing number of nodes, starting from 50 to 65535. As expected, because PEACH makes use of DHTs, after an initial trend (reported in the small box in Figure 7), the *lookup* time grows very slowly with the the number of nodes present in the network. It is to be noted that our simulator does not take in consideration network-related delays, therefore in a real deployment lookup times may be sensibly larger because of communication constrains between nodes.

During our simulations, we also noticed that the overhead introduced by the need to provide signed messages when joining the network was relatively high. The main reason for this is that we use an external command-line based tool[2] to generate the signatures. In PEACH, a peer is require to generate two different signatures in order to perform a *join*(). Therefore it is not surprising that, to simulate a *join*() operation, it can take up to a second, although the signing time per each signature should take only a few milliseconds. A fully C-based PEACH implementation would not suffer from this problem. We envision that the total time for a *join*() operation to occur (without considering time spent on network operations) can be less than 100ms without requiring special cryptographic hardware.

[2] openca-sv, available as part of the openca-tools package from `https://www.openca.org`

5 Integrating PEACH and PRQP

In PRQP each client could use one of the configured RQAs to query for resources related to a CA. When the contacted RQA has no information about the requested service, the client has no alternative way to discover where to forward the request to. By integrating PEACH into an RQA server, the RQA would be enabled to forward the requests into the P2P network and retrieve the missing information.

The RQA can leverage the PEACH in two different ways. The first one is to have the RQA act as a PRQP client. When a request cannot be answered, such as the case where there is a lack of information about the queried CA, the RQA searches for the authoritative RQA on PEACH and, if found, issues a request to the RQA. The returned response is then parsed, and the signature (and the server data) is substituted before sending the response back to the client. The problem with this solution is that the original signature on the response is lost, requiring the client to configure the RQA as a trusted authority (i.e., the RQA is allowed to provide responses for different CAs without having to be certified by them).

The second option is to have the RQA act as a proxy for the client. In this case, the RQA forwards the requests that it cannot answer to the authoritative RQA, but only when it is present on the PEACH network. The response is then forwarded back to the client.

When PRQP and PEACH are integrated, the P2P network maps network addresses to PKI services similar to the way DNS maps logical names to IP addresses. The main difference between the DNS and the RQA network is the absence of a hierarchical system approach.

6 Conclusions and Future Work

In our work, we describe the PEACH algorithm and its application to PKIs. In particular, the presented approach introduces—for the first time—the idea of providing interoperable and collaborative peer-to-peer-based services in X509 PKIs.

We extend the PRQP protocol, which provides a PKI-specific protocol for resource discovery, by providing the starting point for the development of a PKI Resource Discovery Architecture. Under this novel system, different RQAs cooperate to access data that is not locally available.

In a more general sense, PEACH allows for the building of an authenticated peer-to-peer network for client-server-based PKI services. Furthermore, our work can be easily extended to provide other collaborative services. For example, the proposed PEACH protocol would allow for OCSP or TimeStamping services to be integrated among different CAs and PKIs.

In the future, we plan to investigate the applicability of PEACH to the real world. In particular, we plan to release an open-source version of the PEACH enabled server based on the OpenCA-PRQP daemon and to establish collaboration with other authorities to setup a public PEACH network. Overall, this new approach tries to enhance interoperability across PKIs and will be actively promoted within the IETF PKIX working group as a standardized protocol.

Ultimately, we believe that this work will have a signficant impact over the interoperability and usability of PKIs and that it will open up new X509 PKI models based on collaborative services.

Acknowledgments

This work was supported in part by the NSF (under grant CNS-0448499) and the U.S. Department of Homeland Security (under Grant Award Number 2006-CS-001-000001). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any of the sponsors.

References

1. Pala, M., Smith, S.W.: AutoPKI: A PKI Resources Discovery System. In: López, J., Samarati, P., Ferrer, J.L. (eds.) EuroPKI 2007. LNCS, vol. 4582, pp. 154–169. Springer, Heidelberg (2007), `http://dblp.uni-trier.de/db/conf/europki/europki2007.html#PalaS07`
2. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: SIGCOMM, pp. 149–160 (2001)
3. Pala, M.: The PKI Resource Query Protocol (PRQP), Internet Draft, (June 2007), `http://www.ietf.org/internet-drafts/draft-pala-prqp-01.txt`
4. Stoica, I., Morris, R., Karger, D., Kaashoek, F.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. SIGCOMM Comput. Commun. Rev. 31(4), 149–160 (2001), `http://portal.acm.org/citation.cfm?id=964723.383071`
5. Chord, `http://www.pdos.lcs.mit.edu/chord/`
6. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, p. 329. Springer, Heidelberg (2001), `citeseer.ist.psu.edu/rowstron01pastry.html`
7. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A Scalable Content-Addressable Network. In: SIGCOMM 2001: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, October 2001, vol. 31(4), pp. 161–172. ACM Press, New York (2001), `http://portal.acm.org/citation.cfm?id=383072`
8. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing, UC Berkeley, Tech. Rep. UCB/CSD-01-1141, # apr # (2001), `http://citeseer.ist.psu.edu/zhao01tapestry.html`
9. Maymounkov, P., Mazieres, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric (2002), `http://citeseer.ist.psu.edu/maymounkov02kademlia.html`
10. Aberer, K., Mauroux, P.C., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-Grid: A Self-organizing Structured P2P System. SIGMOD Record 32(3) (September 2003), `http://lsirpeople.epfl.ch/rschmidt/papers/Aberer03P-GridSelfOrganizing.pdf`

11. NIST, FIPS PUB 180-2 — Secure Hash Standard, Processing Standards Publication 180-2 (August 2002), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
12. Postel, J.: Domain Name System Structure and Delegation, RFC 1591, (March 1994), http://www.ietf.org/rfc/rfc1591.txt
13. Droms, R.: Dynamic Host Configuration Protocol, RFC 2131 (March 1997), http://www.faqs.org/rfcs/rfc2131.html
14. ISC Bind Server, Homepage, http://www.isc.org/index.pl?/sw/bind/index.php
15. Kaliski, B.: PKCS #7: Cryptographic Message Syntax, RFC 2315, (March 1998), http://www.ietf.org/rfc/rfc2315.txt
16. Pala, M.: The LibPKI project, Project Homepage, https://www.openca.org/projects/libpki/

Author Index