David Chadwick
Gansen Zhao (Eds.)

LNCS 3545

Public Key Infrastructure

**Second European PKI Workshop:
Research and Applications, EuroPKI 2005
Canterbury, UK, June/July 2005
Revised Selected Papers**

Lecture Notes in Computer Science 3545

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Chadwick Gansen Zhao (Eds.)

Public Key Infrastructure

Second European PKI Workshop:
Research and Applications, EuroPKI 2005
Canterbury, UK, June 30 - July 1, 2005
Revised Selected Papers

Springer

Volume Editors

David Chadwick
Gansen Zhao
University of Kent, The Computing Laboratory
Canterbury CT2 7NF, UK
E-mail: {d.w.chadvick,gz7}@kent.ac.uk

Library of Congress Control Number: 2005935450

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

ISSN 0302-9743
ISBN-10 3-540-28062-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-28062-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11533733 06/3142 5 4 3 2 1 0

Preface

This book contains the proceedings of the 2nd EuroPKI Workshop — EuroPKI 2005, held at the University of Kent in the city of Canterbury, UK, 30 June–1 July 2005. The workshop was informal and lively, and the university setting encouraged active exchanges between the speakers and the audience.

The workshop program comprised a keynote speech from Dr. Carlisle Adams, followed by 18 refereed papers, with a workshop dinner in and guided tour around the historic Dover Castle.

Dr. Adams is well known for his contributions to the CAST family of symmetric encryption algorithms, to international standards from the IETF, ISO, and OASIS, authorship of over 30 refereed journals and conference papers, and co-authorship of Understanding PKI: Concepts, Standards, and Deployment Considerations (Addison-Wesley). Dr. Adams keynote speech was entitled 'PKI: Views from the Dispassionate "I",' in which he presented his thoughts on why PKI has been available as an authentication technology for many years now, but has only enjoyed large-scale success in fairly limited contexts to date. He also presented his thoughts on the possible future(s) of this technology, with emphasis on the major factors hindering adoption and some potential directions for future research in these areas.

In response to the Call for Papers, 43 workshop papers were submitted in total. All papers were blind reviewed by at least two members of the Program Committee, the majority having 3 reviewers, with a few borderline papers having 4 or more reviewers; 18 papers were accepted for presentation in 8 sessions. There were sessions on: authorization, risks/attacks to PKI systems, interoperability between systems, evaluating a CA, ID ring-based signatures, new protocols, practical implementations, and long-term archiving.

I would like to thank the authors for their submitted papers, the Program Committee and external reviewers for their conscientious efforts during the review process, the Organizing Committee for their tireless efforts to ensure the smooth running of the conference, and finally all the workshop participants, without whom the workshop would not have been the success that it was.

David Chadwick

Organization

Program Chair

David Chadwick (University of Kent, UK)

Program Committee

G. Bella (University of Catania, Italy)
A. Buldas (Tallinn Technical University, Estonia)
M. Burmester (Florida State University, USA)
G. S. Cooper (University of Salford, UK)
S. De Capitani di Vimercati (University of Milan, Italy)
S. Farrell (Trinity College Dublin, Ireland)
S. Foley (University College Corke, Ireland)
S. Furnell (University of Plymouth, UK)
D. Gollmann (TU Hamburg-Harburg, Germany)
D. Gritzalis (Athens University of Economics & Business, Greece)
S. Gritzalis (University of the Aegean, Greece)
H. Imai (University of Tokyo, Japan)
S. Katsikas (University of the Aegean, Greece)
S. Kent (BBN Technologies, USA)
K. Kim (ICU, Korea)
C. Lambrinoudakis (University of the Aegean, Greece)
A. Lioy (Politecnico di Torino, Italy)
P. Lipp (IAIK, Tech. University of Graz, Austria)
J. Lopez (University of Malaga, Spain)
C. Meadows (NRL, USA)
C. Mitchell (Royal Holloway, Univ of London, UK)
R. Molva (Eurècom, France)
E. Okamoto (University of Tsukuba, Japan)
R. Oppliger (eSecurity, Switzerland)
O. Otenko (University of Kent, UK)
A. Patel (University College Dublin, Ireland)
G. Pernul (University of Regensburg, Germany)
B. Preneel (Katholieke University Leuven, Belgium)
K. Sakurai (Kyushu University Japan)
W.Schneider (Fraunhofer SIT, Germany)
S. Smith (Dartmouth College, USA)
J. P. Stern (Cryptolog, France)
M. Yung (Columbia University, USA)
J. Zhou (Institute for Infocomm Research, Singapore)

Organising Chair

David Chadwick (University of Kent, UK)

Organising Committee

Uche Mbanaso (University of Salford, UK)
Tuan Anh Nguyen (University of Kent, UK)
Jenny Oatley (University of Kent, UK)
Wensheng Xu (University of Kent, UK)
Gansen Zhao (University of Kent, UK)

External Reviewers

George Angelis (University of the Aegean, Greece)
George Kambourakis (University of the Aegean, Greece)
Dimitris Lekkas (University of the Aegean, Greece)
John Iliadis (University of the Aegean, Greece)
Thomas Pornin (Cryptolog International, France)
Lazaros Gymnopoulos (University of the Aegean, Greece)
Dimitris Geneiatakis (University of the Aegean, Greece)
Mario Lamberger (IAIK/Stiftung SIC, Austria)
Bin Zhang (Chinese Academy of Sciences, China)
Torsten Priebe (University of Regensburg, Germany)
Bjoern Muschall (University of Regensburg, Germany)
Christian Schläger (University of Regensburg, Germany)
Margus Freudenthal (Cybernetica, Estonia)
Jose A. Montenegro (University of Malaga, Spain)
Thomas Quillinan (University College Cork, Ireland)
Gregory Neven (Katholieke Universiteit Leuven, Belgium)
Yoshifumi Ueshige (Institute of Systems & Information Technologies, Japan)

Table of Contents

Authorisation

Interoperability

Evaluating a CA

ID Based Ring Signatures

Practical Implementations

New Protocols

Risks and Attacks

Long Term Archiving

A Multipurpose Delegation Proxy for WWW Credentials

Tobias Straub[1], Thilo-Alexander Ginkel[2], and Johannes Buchmann[1]

[1] Darmstadt University of Technology, Computer Science Department, D-64283 Darmstadt, Germany
{tstraub, buchmann}@cdc.informatik.tu-darmstadt.de
[2] TG Byte Software GmbH, D-64625 Bensheim, Germany
thilo@ginkel.com

Abstract. Credentials like passwords or cryptographic key pairs are a means to prove one's identity to a web server. A practical problem in this context is the question of how a user can temporarily delegate the right to use a credential to another person without revealing the secret. Related to this is the issue of sharing a single credential among members of a group such that all of them may use the credential, but no one actually gets to know it. This paper presents and compares several solutions to solve these problems. One is a client-side approach, while the other three are man-in-the-middle architectures. We have implemented one of these, the HTTP proxy variant, as a prototype. Our *TLS Authentication Proxy* is capable of transparently authenticating with a target web server on behalf of users. It supports the major authentication methods used on the Internet, both for standard HTTP and SSL connections.

Keywords: Credential Delegation, Man-In-The-Middle, Usability, X.509 Certificate, WWW Authentication.

1 Introduction

Most client/server applications that can be found on the Internet today rely on HTTP as their communication protocol. Online stores, web-based e-mail and online banking are popular business-to-consumer services, whereas typical business-to-business solutions include virtual market places and enterprise portals. WWW technologies are also used in corporate intranets since basing applications on a web browser and a web server is a ubiquitous and cost-effective solution. For security reasons, each of the aforementioned use cases usually requires some form of user authentication. As a consequence, users need to possess and apply *credentials* like passwords or cryptographic key pairs in order to prove their identity to the respective server. The server itself or a dedicated back-end system subsequently assigns access rights to this identity such that the client is able to use the corresponding services.

Credential delegation is the focus of attention of the current paper. Temporarily delegating the ability to access a certain service or resource to a colleague is

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 1–21, 2005.

a natural requirement, but is rarely supported on the server side. Consequently, people have no choice other than sharing the respective credentials. Obviously, this has negative security implications, especially when the same secret is used for multiple purposes (for example, the same password is used on different systems or a key pair is used for authentication and encryption). Besides, this naive method does not allow us to restrict the proxy's capabilities to a certain period of time or prevent him from further distributing the secret.

In this paper, we present a practical solution addressing the matter of credential delegation for common authentication methods used with standard HTTP as well as with SSL. Especially, X.509 [1] client certificate credentials pose a particular challenge. Our approach handles both password- and certificate-based credentials, but does not require any changes on the server side and only minor changes on the client side. In addition, it provides a tool for *credential management* helping users to keep track of which services they have signed in on the Internet and what user names and passwords they have chosen.

Our work is structured as follows: In the next section, we motivate the subject and highlight characteristic problems by means of a small scenario. Related work is reviewed in Section 3. Four different system architectures are presented and compared in the following section. A prototype, which was developed as part of the project [2], is described in Section 5, before a discussion concludes the paper. We assume that the reader is familiar with the most common authentication methods on the Internet. A short survey for non-specialist readers is given in the appendix.

2 Credential Delegation

According to Oxford Advanced Learner's Dictionary, *credentials* are "documents showing that a person is what he claims to be, is trustworthy, etc." Similarly, the purpose of a credential in the digital world is to prove the identity of its owner toward another party, typically a server. Credentials can basically be grouped into three classes: Authentication may be based on the *knowledge of a secret*, usually a password, a PIN, or a passphrase, which have to be provided to the system in conjunction with a user name that serves as a (public) unique identifier. Another class is authentication based on the *possession of a token*. A typical realization is a challenge-response protocol using a cryptographic key pair. In this scheme, the user proves that he possesses the secret (the private key) to the server without revealing it. *Biometrics* give rise to a third class of user authentication methods, but have little importance on the WWW at the moment. Moreover, physical characteristics effectively tie a credential to its owner and prevent impersonation of any kind, so we do not consider this class further. We refer the reader to Appendix A for a detailed description of common authentication mechanisms used on the Internet.

2.1 Basic Scenario

The employees of ACME Inc., a medium-size IT company, access a couple of web sites protected by different user authentication schemes. For instance, the

company's manager, Alice, handles the corporate bank account via the Internet. The bank's web server is accessed through an SSL/TLS channel. It requires strong client authentication using a key pair and the corresponding X.509 certificate the bank had initially distributed in the form of a PKCS#12 [3] soft token. Alice wants her proxy Bob to manage the company's bank transactions during her summer vacation. Since she considers trust to be good, but control to be better, she also wants a way to monitor when Bob actually loges in on her behalf while she is absent.

Carl has subscribed to a commercial web portal on behalf of ACME. The services are charged on a per-request basis and require a logon with a user name and password. The management wants to keep track of how often members of the research department use the portal and restrict access to office hours. Another requirement is a possibility to easily withdraw the authorization of engineers leaving the company. Unfortunately, ACME has only a single company-wide account.

2.2 Requirements

The previous scenario highlights problems with credentials that are *delegated* or *jointly used* by a group of people. By delegating a credential, its owner permits another person to temporarily impersonate her toward the target server, i.e. to adopt the virtual identity of the owner. Group usage can be regarded as a generalization of this concept. Both applications require that the capability to impersonate the credential owner can be controlled and revoked when needed. Following the naive approach of handing out the credential to the proxy or sharing it with the group members is insecure. "Revoking" a credential based on knowledge can be achieved by altering the secret, although this is awkward and is only possible if the server permits it. In general, this possibility is ruled out for authentication schemes based on the possession of a token, since it is issued by the server's operator or a third party. In both cases, there is no way to prevent proliferation of the credential, nor a guarantee that it has been deleted. There is no possibility to control when and which protected resources are accessed and by whom.

Consider again the case where a credential consisting of an X.509 certificate should be delegated. The key pair might be used for a number of purposes including secure e-mail, file encryption, document signing etc. In this situation, it is intolerable that the proxy gets to know the private key, which is a valuable secret used over a rather long time (the certificate's lifetime may be in the order of years). We seek for a secure way to solve these problems. To sum up, a solution should meet the following requirements:

- ▷ Perform user authentication for an authorized person on behalf of the credential owner without revealing the secret to the former.
- ▷ Protect the credentials from unauthorized access.
- ▷ Keep track of the actual usage of a credential that is managed by the system.

- ▷ Support common WWW authentication mechanisms, namely Basic and Digest Authentication, form-based variants as well as SSL with client certificates.
- ▷ Operate transparently without additional software, neither on the server nor on the client.
- ▷ Require no re-configuration on the server and only minimal re-configuration on the client side. Provide an easy-to-use interface to manage users, credentials and access rights.

The first requirement suggests we introduce a level of indirection between credentials and users that are allowed to *dispose* of some of them. To fulfill the second one, credentials have to be stored in a trusted environment and revealed (or applied in case the credential is a key pair) during the communication with the server. The third requirement states that an audit log be in place. This allows us to find out who (mis-)used a credential at a certain point in time. Since we expect a growing importance of SSL client authentication on the Internet in the near future, we pay special attention to this issue and consider it as well as password-based schemes.

3 Related Work

The problem of credential management typically leads to identity management or *Single sign-on* (SSO) technologies. SSO provides a means to reduce the number of authentications a user has to go through. After an initial authentication with an authentication service provider (ASP), protected resources can be accessed without further authentication. Surveys on SSO architectures can be found in [4,5,6]. SSO solutions can be categorized depending on whether the target host is aware of the ASP's involvement or not. In the notion of [4], *true SSO* services like Microsoft's Passport[1] require an established trust relationship between the ASP and the target web server, as well as the implementation of the appropriate protocol. Opposed to this are *pseudo-SSO* services.

SSO infrastructures form an ideal basis to implement a delegation mechanism. For instance, Kerberos [7] supports this feature, but of course Kerberos was originally not designed as an authentication mechanism for the web. However, there is in fact an Apache module[2] for the so-called HTTP Negotiate authentication method. Microsoft Internet Explorer and the Mozilla browsers support this method, too.[3] Kornievskaia et al. [8] propose a system to access kerberized services through a web browser in order to leverage the credential delegation functionality natively provided by Kerberos for WWW resources.

SPKI/SDSI certificates [9,10] are designed to support delegation originally. *Greenpass* is an application that uses SPKI/SDSI certificates in an X.509 certificate environment to express the delegation of wireless network access rights.

[1] URL: `http://www.microsoft.com/net/services/passport/`

[2] URL: `http://modauthkerb.sourceforge.net/`

[3] Kerberos authentication is already supported natively by Mozilla and Mozilla Firefox on UNIX platforms, a Windows plug-in is available at `http://negotiateauth.mozdev.org/`.

A description is found in the paper of Goffee *et al.* [11]. Greenpass is compatible with standard clients, but nevertheless requires a re-implementation of the authentication protocol on the server side. The current version only supports X.509 certificates as credentials, but no password-based schemes.

Credential delegation is a natural requirement in grid computing as programs should be able to run autonomously on a user's behalf with a subset of her rights. The Grid Security Infrastructure (GSI) uses X.509 credentials in combination with the SSL protocol to mutually authenticate grid users and resources [12]. An entity can delegate rights to a third party with the help of X.509 proxy certificates [13] that have their own key pair as well as a particular X.509 extension and a rather short validity period. *MyProxy* is a credential repository designed to work with a grid portal that provides services for delegation management [6]. Weaker mechanisms like Basic or Digest Authentication are not supported. An advantage of this approach is the fact that users do not have to reveal their long-term credentials for delegation. However, the fact that this framework requires server-side support is an important downside that excludes this framework for our scenario. The same restrictions apply to the attribute certificate framework defined in the X.509 standard [1]. This approach requires a sophisticated Privilege Management Infrastructure (PMI) and limits the capability of delegation to so-called attribute authorities (AA) which are typically not end-entities. Plus, the attribute certificate profile [14] explicitly discourages the use of attribute certificate for delegation due to the high complexity of the verification process.

Impostor[4] follows a similar approach as a TLS Authentication Proxy in that it performs a MITM attack on SSL. Working as an SSO proxy, it also provides content filtering, but does not issue its own replica certificates (see below).

4 System Architecture

In this section, we present four different architectures that fulfill the requirements stated in Section 2.2. One of these architectures is a client-side solution (discussed in Section 4.4) while the others require a server acting as a *man-in-the-middle* (MITM). This setting is depicted in Figure 1. The *application server*, the *HTTP server*, and the *HTTP proxy* architecture come under this category. The illustration shows the communication between the client and server via an intermediate MITM host. The segments between client and MITM as well as between MITM and server can be secured to prevent eavesdropping and ensure data integrity.

In the following, we use the terms *gateway* for the MITM component and *target* server/host to indicate the actual connection endpoint, i.e. the machine that serves the requested web pages. Requests from the client to the target host go through the gateway where they are augmented with authentication credentials. In particular, credentials never leave the MITM host's protected environment. In order to allow a fine-grained access control, the gateway needs to authenticate

[4] URL: `http://impostor.sourceforge.net/`

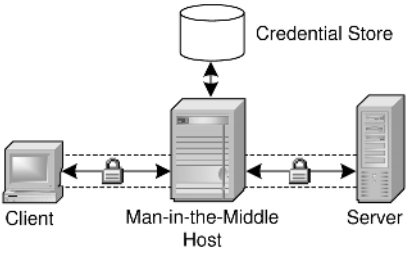

Fig. 1. Man-in-the-middle architecture

each user before permitting access to its services. The adopted authentication mechanism largely depends on the actual implementation and can range from plain-text passwords to cryptographically strong public key mechanisms.

The technologies described in Section 4.1 to 4.3 mainly differ with respect to the communication between the end-user's client and the gateway. Loosely speaking, an end-user works with a web browser running on the gateway instead of on her own machine in the *application server* variant. By way of contrast, in the other two variants she runs a web browser locally on the client-side. She either points her browser to the *HTTP server* or configures it to go through an *HTTP proxy* server in order to access the pages on the target host.

4.1 Application Server Variant

The application server concept is based on a machine that provides remote login facilities. Possible realizations range from a full-screen remote desktop (such as VNC[5]) to the forwarding of single browser windows (as is the case with X11 forwarding). A person, who wants to make use of a certain credential, has to login to the application server first. After the session has been established, the person connects to the target server with a standard browser running remotely on the application server. Most modern operating systems are shipped with tools for application server access, so this variant can be easily implemented. On the other hand, popular browsers such as *Microsoft Internet Explorer* or *Mozilla* provide certain functionality for credential management.

However, handling delegations properly or enforcing a consistent policy is difficult. Special precautions are necessary to ensure that a user cannot gain unauthorized access to credentials stored on the application server (and for instance export key pairs). Additionally, users need to work remotely, adding latency to the interaction and exposing them to an environment that possibly differs significantly from the one they usually work with.

4.2 HTTP Server Variant

In the HTTP server approach, the gateway acts as a web server where connections to protected resources are initiated by calling a particular URL on the

[5] URL: `http://www.realvnc.com/`

gateway. The gateway then retrieves the necessary credential and opens a channel to the target host. The target server's response is returned to the client to which it appears as coming from the gateway. As the URL requested by the client differs from the actual URL, the MITM host needs to modify content coming from the target host in order to redirect hyperlinks contained in HTML or JavaScript documents to the gateway's host name. Otherwise, subsequent requests would directly address the target server thereby skipping the gateway's proxy authentication.

A major advantage of this approach are the low deployment costs. There is no need to modify the client-side configuration and SSL can be handled as well (by equipping the MITM with a single X.509 certificate issued by a trusted certification authority). However, there are downsides: Binary data such as Macromedia Flash or Java bytecode makes content re-writing very difficult. Additionally, a Java applet that runs in a sandbox may only communicate with the server it was retrieved from by the client, so the applet's channel to the original server is cut.

4.3 HTTP Proxy Variant

A natural idea is to use an HTTP proxy and enhance it with certain functionality to handle authentication information (in the terminology of RFC 2616 [15], such proxies are called "non-transparent"). HTTP proxies are a well-known concept. They are typically used as so-called caching proxies in order to reduce network traffic or as an application level proxy in conjunction with a firewall. When processing standard HTTP requests, a proxy works as a forwarding agent in between the requesting client and the responding host, itself acting both as a server and client. This allows the proxy to modify requests in order to add authentication credentials if necessary. All HTTP-based schemes listed in the requirements (Section 2.2) can be handled that way.

Things get a bit more complicated with HTTPS requests. To initiate an SSL session through a proxy, there is a special command (the CONNECT method, see [16] for details). However, after a secure, i.e. encrypted and authenticated, tunnel between the client and the server has been established, the HTTP proxy is unable to change or eavesdrop on the data in transmission. The only way to work around this restriction, is to loosen the principle of *end-to-end* security. This can be done by letting the proxy pretend toward the client that it is the target host. In practice, such a *man-in-the-middle attack* is effectively prevented through the use of a certificate that identifies the server. However, if the proxy is given a *replica* certificate that

(a) binds *the proxy's* public key to the identity of the target site and
(b) is issued by a certification authority (CA) which is trusted by the client,

the proxy can successfully impersonate the target host toward the client. If the proxy furthermore has access to an authentication credential (a key pair and the respective certificate), it can also impersonate the credential owner toward the target host.

Compared with the HTTP server, the installation of the CA certificate as a new trust anchor on the clients causes slightly higher deployment costs, but content re-writing becomes unnecessary and support for any content type is thus guaranteed.

4.4 Client-Side Architecture

The idea behind the client-side approach is to equip an HTTP user agent with additional functionality to access a credential store. After retrieving the appropriate credential from the central repository, authentication can take place as usual, even without the need to prompt for user input. This variant does not need a machine sitting in between the client and the target host, but only an enhancement to the web browser. For open-source browsers like *Mozilla* or its *Firefox* branch, the extension can be integrated directly in the code. It should be possible to also implement the same functionality for *Internet Explorer* using its plug-in and extension COM interface. However, as we have not implemented any of these extensions, it is difficult to give cost estimates. Another alternative, which is independent of a particular user agent, is to fit up the client's TCP/IP stack or socket library with support for user authentication via the credential store. For instance, on the Microsoft Windows platform, this can be accomplished by a wrapper around the Winsock or WinInet DLL. Such a wrapper has to detect connections requiring authentication, retrieve the suitable credential from the credential store and transparently perform the authentication.

The major advantage of this client-side concept is the seamless integration with the user's web browsing environment. While credential use can and should be tracked by the central credential store, the actual communication between client and server is kept private when SSL/TLS is used. Unfortunately, this comes for the price that credentials need to be revealed to the client. Therefore, a malicious client could learn the secret information.

4.5 Comparison

In the following, we sum up the pros and cons of each architecture with respect to five categories. These include standard compliance and compatibility with existing software, transparency for the end-user, usability, security characteristics, and deployment costs. A quantitative score is given in Table 1 at the end of this section. We use the following notation for the rating: $\circ$ denotes an average, $+$ and $-$ denote slightly positive and negative results respectively. Excellent respectively insufficient grades are abbreviated by $++$ and $--$. Please observe that in some cases the actual rating may vary according to the actual system environment and the implementation variant. Details are given in the following text.

Compatibility and Standard Compliance. As the application server uses a standard browser, it supports a variety of protocols and formats, with the possible exception of plug-ins or ActiveX controls that require root privileges the end-user

has not been granted on the application server. Access to credential management facilities is generally restricted to software running directly on the application server. The limiting factor of the HTTP server is its need for URL rewriting as outlined in Section 4.2. In this regard, the HTTP proxy is superior as it only requires a user agent supporting HTTP proxies (which can be taken for granted). Besides, other applications can use the proxy's services through a standardized protocol, too. The latter statement also applies to the client-side approach with a modified TCP/IP stack or socket library (which are nevertheless no platform-independent solutions). A customized web browser gets a lower compatibility rating since it only supports a restricted set of applications.

Transparency. Transparency refers to the question to what extent the user gets in touch with the credential delegation infrastructure. Obviously, each of the MITM architectures requires an additional step of authentication toward the gateway. The rating for the application server approach varies significantly depending on the actual realization (see Section 4.1). A solution implementing a seamless integration into the user's work environment, such as X11 forwarding, receives a better rating than a classic remote-desktop solution like VNC. While HTTP server users can still work with their usual environment, accessing authenticated WWW resources is not seamless as it requires an additional step to point the web browser to the gateway page. In comparison, the HTTP proxy operates transparently after an initial configuration. The client-side approach is optimal if the user's favorite browser is supported, but may cause some inconvenience otherwise.

Usability. Usability is a measure of how easily a system can be used and how well it satisfies user's expectations and needs. The usability of the application server approach heavily depends on the actual realization (e.g. if the "look & feel" is the same) and on network latency. While the latter concern is a minor issue for the HTTP server, it nevertheless confronts the user with a non-standard way of accessing protected WWW resources: Instead of using the browser's controls such as the location bar to navigate to the target address, the user has to use an HTML form provided by the gateway to navigate to the desired page. The HTTP proxy is favorable from a usability point of view as it does not require any uncommon actions. So is the client-side approach as long as the user can work with his favorite browser.

Security Characteristics. The main security goals are to protect the stored credentials and to keep the transferred data confidential. All approaches have in common that they use a central credential repository, which is an attractive target for attackers. However, protecting a single system is still easier than caring for a lot of decentralized credential stores on the client systems, possibly having to deal with heterogeneous operating systems and user agents. A downside of the MITM architectures is the fact that payload data is available in the clear on the gateway – even if the target host is accessed via SSL. These architectures are thus vulnerable to insider attacks (e.g. by the system operator), but on the other

hand offer the possibility to audit transactions. The communication between application server and client should be analogously protected against wiretapping (e.g. by SSH port forwarding). It is questionable whether the application server environment can be sufficiently hardened to prevent attempts to read out stored credentials. Credential protection is a major weakness of the client-side approach. However, it is in fact the only solution to provide real end-to-end security for the communication between the client and the target host.

Deployment Costs. On a Windows XP or an X11-based client system, deployment costs for the application server approach are low since the necessary software is already pre-installed. For all MITM architectures, the gateway's public key has to be trusted by the clients. If this key is certified by a CA that is unknown to the clients, the appropriate trust anchor has to be installed on all systems. This step can be combined with the rollout of end-user certificates if such are used for authentication with the gateway. In the HTTP proxy realization, one has to have a key pair to issue replica certificates. This requires in turn a certificate with the *Basic Constraints* extension set to "CA". Commercial CAs charge a lot of money for certificates of this kind, so the HTTP proxy will probably rather rely on its own CA. A minor problem is the single change in the browser's proxy settings, which can be automated for some browsers (see Section 5.4). A client-side implementation comes with the highest deployment costs of all architectures as it requires software installation on all client systems.

Table 1. Comparison of the four architectures

	App. Server	HTTP Server	HTTP Proxy	Client-Side
Compatibility	○	○	+	○/−*
Transparency	+/−−*	○	+	++/○*
Usability	○	○	++	+
Security	−	+	+	−
Deployment	+/−−*	++/○*	○	−−

*Rating depends on the actual system environment and implementation variant, see text for details.

5 Prototype

We have implemented a prototype of the HTTP proxy variant, since we consider it superior to the other MITM architectures from a technical viewpoint and

estimated its development costs lower than those of a client-side variant. After giving an overview of the proxy's modular architecture, we go into technical details in Section 5.2. The user interface is described in the following section, deployment issues are discussed in Section 5.4.

5.1 Overview

An overview of the modules of the prototype, which we called *TLS Authentication Proxy*, is shown in Figure 2. The components can be grouped largely into two categories: A back-end and a front-end, which are shown on the left-hand and the right-hand side of the figure, respectively. The back-end is built around the underlying database management system (DBMS) and contains modules for the management of proxy users, credentials, and access control policies. Further components are a certification authority, a servlet-based web administration interface, and a logging module. The front-end comprises the networking and HTTP engines.

Our proxy is written in Java 2 SE 1.4 and uses the *FlexiProvider*[6] JCE implementation as cryptographic service provider. ASN.1 support is provided by the *Fraunhofer Codec*[7] package, which is relevant for parsing and issuing X.509 certificates. The relational database *PostgreSQL* is used as the back-end. All administrative tasks can be handled through a web interface, which has been implemented using the *jetty://* HTTP servlet container[8].

5.2 Technical Details

In this section we describe our implementation in a bit more detail providing information about the network communication schemes, session management as well as authorization and auditing. We refer the reader to [2] for a thorough treatment.

Network Communication. Upon receiving a request from the network, the proxy parses it in order to extract the target host and request method. Our implementation supports all request methods (GET, POST, CONNECT etc.) defined in RFC 2616 [15]. If the resource on the target host is known to require a credential, the proxy first authenticates the requesting client, looks up its permissions, and starts the authentication with the target host if applicable. Otherwise, TLS Authentication Proxy acts as a usual non-caching HTTP proxy.

In order to identify the user, the proxy may use SSL with client authentication – which we chose to implement for the current prototype – or Basic/Digest Authentication. The situation where a user requests a resource that is accessible via standard HTTP is somewhat special since it requires a redirection to an SSL page in our implementation. This case is therefore discussed later (see Figure 4).

[6] URL: `http://www.flexiprovider.de`
[7] URL: `http://www.semoa.org/misc/codec.html`
[8] URL: `http://jetty.mortbay.org/jetty/`

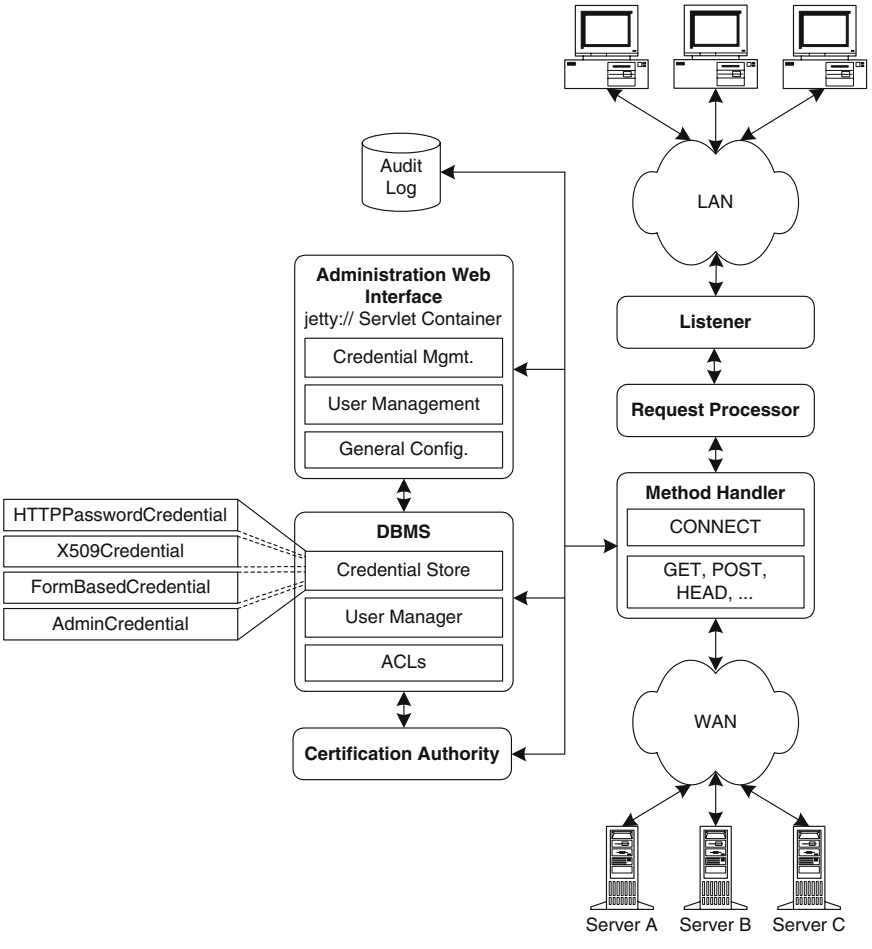

Fig. 2. Module structure of the TLS Authentication Proxy

We now describe the handling of target sites that are accessed via SSL. The necessary steps are shown in Figure 3.

By means of the standard CONNECT method, a user agent indicates to the proxy that it wants to access an SSL-protected website. As a first step, TLS Authentication Proxy and the server start an SSL handshake with a premature end after the server has sent its certificate. The CA module then produces a replica of the server's just retrieved X.509 certificate. This replica is then presented to the client in a subsequent SSL handshake where the proxy impersonates the original server. During this handshake, the proxy demands certificate-based authentication from the client. Having successfully established a cryptographic channel between client and proxy, the proxy selects the credential and goes through a complete SSL handshake with the server this time. If the credential is a key

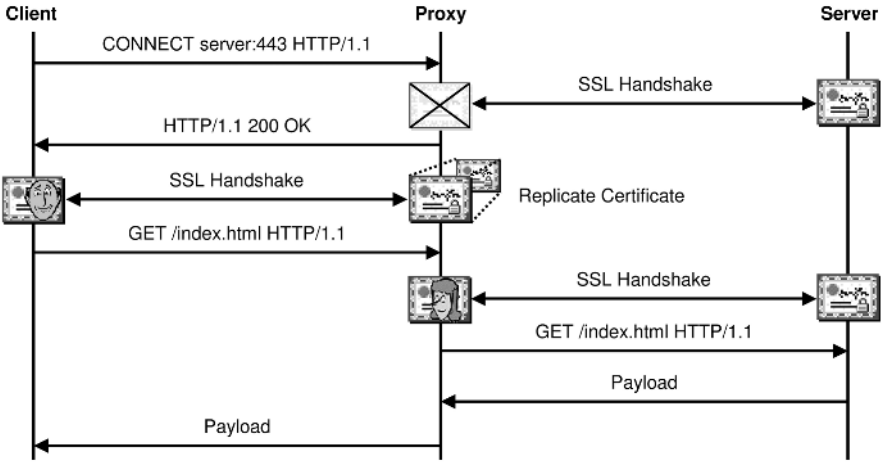

Fig. 3. Data flow for an SSL target web site

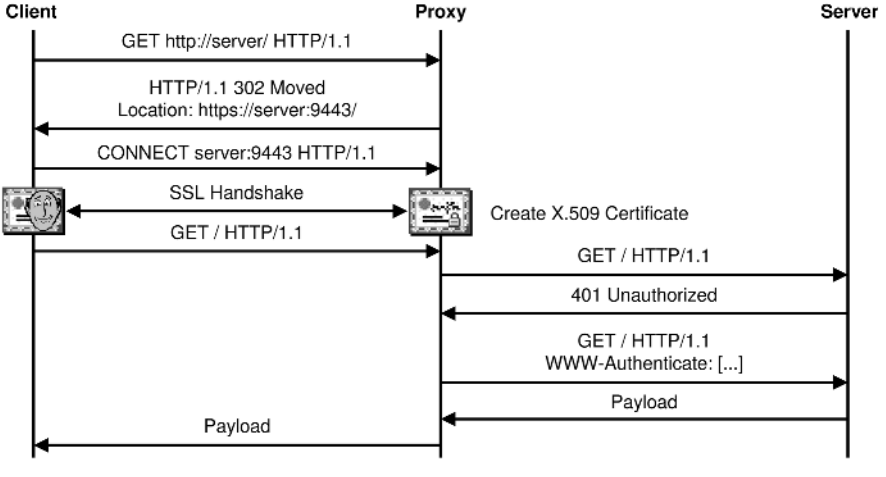

Fig. 4. Data flow for an HTTP target web site

pair, the proxy is already authenticated during the handshake. Otherwise, the last messages exchanged between proxy and server in Figure 3 look slightly different (like in Figure 4 for the case of Basic/Digest Authentication).

Figure 4 illustrates how the proxy identifies a user who requests a protected resource via HTTP. Instead of passing through the request to `http://server/` (as a usual HTTP proxy would do), TLS Authentication Proxy redirects the web browser to the URL `https://server:9443/` forcing it to CONNECT to

the SSL site. Here, the reserved port number of 9443 indicates to the proxy not to act as in the situation of Figure 3. When the proxy gets a subsequent request tagged with this port number, it connects to the target server using HTTP as originally intended – even though the connection between client and proxy has been secured by SSL. (The actual target port – if different from the default value 80 – is picked from the internal database.) A future extension might use the connection upgrade mechanism specified in RFC 2817 [17] to dynamically change an HTTP connection to HTTPS. This would allow an even more seamless user experience. However, current web browsers do not support this protocol extension yet.

Session Management. To speed up successive Basic Authentication requests, the proxy maintains an authentication realm cache. Instead of having to wait for the server to return a 401 status code, the proxy immediately sends the authentication headers with the request itself. This optimization is not applicable to Digest Authentication or other challenge-response schemes. In contrast, form-based authentication schemes usually do not require to each single request re-authenticate, but apply some form of session management instead. Popular implementations of such a session management mechanism include URL-based and cookie-based session tracking. Schemes that embed session identifiers into the URL can be grouped into two categories: After successful client authentication, either an HTTP redirect (Status code `302 Found` in conjunction with a `Location` header) sends the browser to a new URL containing the session ID or the browser receives personalized web pages with the ID embedded into each hyperlink.

Our prototype comes with full support for cookie-based session management and works with the redirection-based mechanism. New sessions are established either automatically when a web site is accessed for the first time (or the previous session has expired) or manually. This comprises a so-called trigger URL, which is an arbitrary URL on the target web site the user calls to initiate authentication. URL-based sessions cannot be established automatically, but require the use of a trigger URL. TLS Authentication Proxy keeps track of cookie-based sessions and transparently transmits the HTML authentication form data when needed. The server's login URL as well as the authentication parameters – which are usually entered into the HTML form – are stored along with the other credential information. As sessions often expire after a certain period of inactivity, form-based credentials have a configurable maximum session lifetime after which the proxy automatically re-authenticates.

Authorization and Auditing. When end-users connect to the TLS Authentication Proxy, they are identified by means of the distinguished name stored in their X.509 client certificate as already outlined. We now consider the remaining issues of user authorization and auditing.

Given the name of the authenticated user on the one hand and the requested target resource on the other hand, TLS Authentication Proxy decides whether to grant access or not as follows. Based on the host name, the port number and the

protocol name (i.e. "http" or "https"), the gateway checks whether it actually possesses a credential for the target host in question and whether the current user is allowed to dispose of it. Technically speaking, all access rights are modelled by a single database table indexed by a pair of user and credential ID. Constraints can be defined on a fine-grained basis for each user/credential pair (see next section). Our prototype uses a flag for each table entry to indicate whether sub-delegation of the particular access right is allowed. However, the system does not yet restrict the number of consecutive sub-delegations of a credential.

At the moment, there are only two administrative roles in the system, namely administrators and non-administrators. Administrator privileges are required for the setup of users or credentials and the initial delegation of a credential. Obviously, this has far-reaching security and trust implications as malicious gateway administrators could effectively impersonate users. This point should be addressed in a future version, e.g. by means of a more sophisticated role model or the enforcement of a four-eye principle for security-critical tasks. The security of the underlying operating system and the credential database was out of the scope of this work. For instance, a hardware storage module could be used to prevent system administrators from stealing credentials.

The task of the logging module is twofold. On the one hand, it audits all access to the web interface (which is described in the following section). On the other hand, every use of a credential stored on the proxy is recorded from the back-end. Both log files adhere to the de-facto standard *NCSA Extended/Combined Log Format*[9], which is used for instance by the Apache web server. A number of tools are available to parse and evaluate such files. Among other pieces of data, the time and date of access, the name of the authenticated user, and the requested resource and credential are kept. In the current version of the prototype, this information is not yet available to end-users, but only to an administrator.

5.3 User Interface

TLS Authentication Proxy offers a web-based user interface, which is accessible via the reserved URL `https://proxyadmin/`. Logins to this site always require a user certificate issued by the proxy's CA. The authentication process is similar to the way that other protected resources are accessed. End-users can browse this site to learn which credentials they are allowed to use and delegate. Assume that a user Alice delegates a credential to Bob. Along with this information, Alice may define a time frame in which Bob has access to the service assigned to this particular credential. This is the only constraint that is currently implemented, but others are conceivable, e.g. a maximum number of total/daily logins or a restriction to a subset of web pages. Due to the modular implementation, custom constraints can be added easily. If necessary, the right to use a credential can be revoked on a per-user basis.

Whether a person may also access the proxy's administration functionality is determined by a predefined *AdminCredential*, which can be added to a user's

[9] URL: `http://hoohoo.ncsa.uiuc.edu/docs/setup/httpd/LogOptions.html`

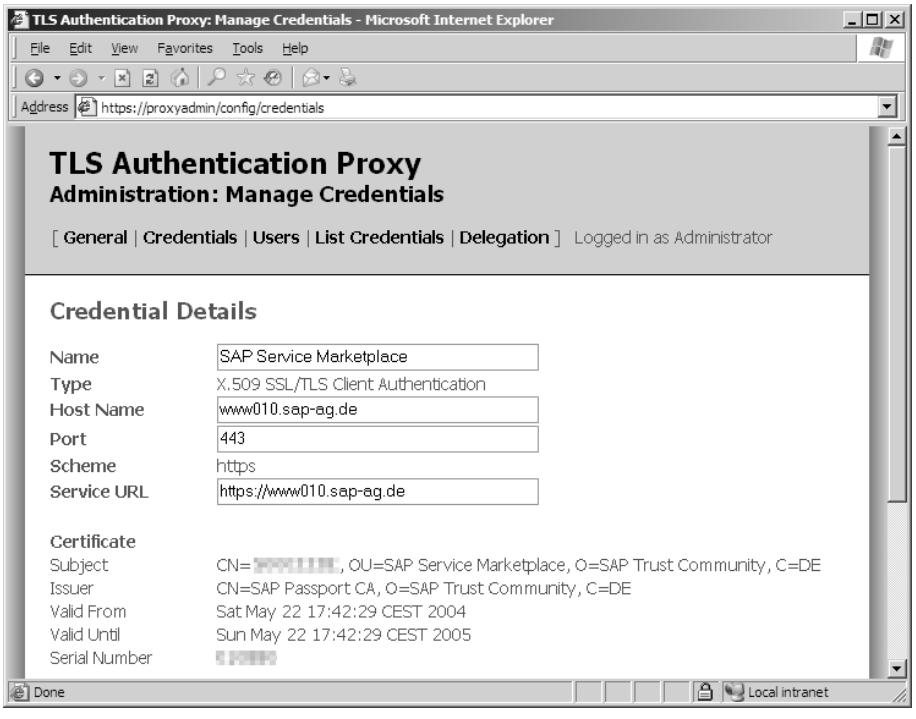

Fig. 5. Details for an X.509 credential

capabilities list like any other credential. Only persons who have been granted the right to use this special credential may access the administration GUI. Apart from general network and proxy settings (for instance the address ranges that are allowed to connect), the administration interface provides functionality for user and credential management. Creating a new user goes along with the instant generation of a key pair and the issuance of a certificate for that person. Both objects are contained in a PKCS#12 [3] file that the administrator can download together with the corresponding transport PIN. In the current implementation, certificates are renewed manually via the administration interface. Instead of implementing certificate revocation lists, we chose to give the administrator the ability to disable users temporarily or permanently, which is more flexible.[10]

Figure 5 shows an example of an X.509 credential for the SAP Service Marketplace imported into the proxy. The setup of form-based authentication is a bit more complicated since the parameter names and values still have to be extracted manually from the HTML source code. However, this could be automated with some effort.

[10] For security reasons, certificates issued by the proxy's CA carry a critical extension restricting their usage to the purpose of client authentication.

5.4 Deployment

There are two deployment problems with the HTTP proxy approach. One is the distribution of keys and certificates, the other is web browser configuration. Key pairs and certificates are delivered within a PKCS#12 container file, which also contains the certificate of the proxy's CA. When importing a personal key pair using Internet Explorer and the Windows CryptoAPI, the CA certificate is automatically installed, too. In contrast, Mozilla requires an additional step of explicitly introducing a new trust anchor to the system.

In order to further minimize deployment costs, our prototype supports the Web Proxy Auto-Discovery (WPAD) Protocol [18], which can be used to automatically obtain a Proxy Auto-Configuration (PAC) [19] file from the network. The PAC file contains JavaScript code, which dynamically tells the browser the URLs to use a proxy server for and what the address of that server is. WPAD is directly supported by Internet Explorer, a patch for Mozilla is available. As an alternative, either the proxy server's address or the URL of the PAC file can be configured manually in the browser.

To counter privacy concerns, the PAC file can be adjusted to only route traffic through the proxy if the target host requires a credential. Due to the way PAC works, the configuration files can be obtained anywhere in the local network and allow an adversary to learn which credentials are kept in the proxy's repository. As a countermeasure, the proxy is able to apply a cryptographic hash algorithm as a one-way function in order to obscure the host names. For this purpose, the prototype uses Paul Johnston's JavaScript implementation of MD5[11]. At runtime, the PAC file's code is executed to compute the hash of the current server name. If and only if the result matches one of the stored values, the proxy is used.

6 Conclusions

In this paper we have presented different architectures to implement a revocable delegation of WWW credentials for today's most common authentication methods on the Internet. Our work was especially motivated by the difficulty of delegating X.509 credentials, which we consider an important requirement in the near future. We have implemented an HTTP proxy with enhanced functionality, the TLS Authentication Proxy. It allows an efficient delegation and secure group usage of credentials. A major benefit of our approach is that only minimal client configuration changes and no changes at all on the server side are required. TLS Authentication Proxy is a zero footprint solution as no additional software is required neither on the client nor the server side.

Credentials are stored in a central, well-protected database rather than spread over many heterogeneous client systems. Delegation of credentials can happen in a secure and revocable manner as client systems never gain knowledge of them. Auditing facilities log who actually delegates or uses a credential,

[11] URL: `http://pajhome.org.uk/crypt/md5/`

when and for what purpose. In our opinion, the light-weight TLS Authentication Proxy represents a good compromise between usability and security.

Additionally, our prototype is a pseudo-SSO tool and facilitates the deployment of new credentials via a centralized roll-out. TLS Authentication Proxy also supports roaming scenarios where users move around among different machines. Surely, in this use-case the current X.509 user authentication is not optimal, but the implementation of a one-time password mechanism is conceivable.

Acknowledgements

We are grateful to the anonymous reviewers for their valuable suggestions. We also thank Andreas Pashalidis for pointing out relevant literature.

The work of the first author was supported by the German National Research Foundation (DFG) as part of the PhD program "Enabling Technologies for Electronic Commerce" at Technische Universität Darmstadt.

References

1. CCITT: Recommendation X.509: The Directory - Authentication Framework. Technical report (1988)
2. Thilo-Alexander Ginkel: Entwurf und Implementierung eines Authentifikations-Proxys für das World Wide Web. Diploma thesis, Technische Universität Darmstadt (2004) URL: `http://thilo.ginkel.com/diplom/`.
3. RSA Laboratories: PKCS#12 v1.0: Personal Information Exchange Syntax. Standard (1999) URL: `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-12/`.
4. Pashalidis, A., Mitchell, C.J.: A Taxonomy of Single Sign-On Systems. In: Proc. Information Security and Privacy – 8th Australasian Conference, LNCS 2727, Springer-Verlag (2003) 249–264
5. De Clercq, J.: Kerberized Credential Translation: A Solution to Web Access Control. In: Proc. InfraSec 2002, LNCS 2437, Springer-Verlag (2002) 40–58
6. Basney, J., Yurcik, W., Bonilla, R., Slagell, A.: Credential Wallets: A Classification of Credential Repositories Highlighting MyProxy. In: Proc. 31st Research Conference on Communication, Information and Internet Policy. (2003) URL: `http://www.ncsa.uiuc.edu/~jbasney/credentialwalletTPRC.pdf`.
7. Kohl, J., Neuman, C.: The Kerberos Network Authentication Service (V5). RFC 1510 (1993) URL: `http://www.ietf.org/rfc/rfc1510.txt`.
8. Kornievskaia, O., Honeyman, P., Doster, B., Coffman, K.: Kerberized Credential Translation: A Solution to Web Access Control. In: Proc. 10th USENIX Security Symposium. (2001) 235–250 URL: `http://www.usenix.org/events/sec01/full_papers/kornievskaia/`
9. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: Simple public key certificate. IETF Internet Draft (1999)
10. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI Certificate Theory. RFC 2693 (1999)
11. Goffee, N.C., Kim, S.H., Smith, S., Taylor, P., Zhao, M., Marchesini, J.: Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In: Proceedings 3rd Annual PKI R&D Workshop. (2004) 16–30

12. Butler, R., Welch, V., Engert, D., Foster, I., Tuecke, S., Volmer, J., Kesselman, C.: A National-Scale Authentication Infrastructure. IEEE Computer **33** (2000) 60–66
13. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) – Proxy Certificate Profile. RFC 3820 (2004) URL: `http://www.ietf.org/rfc/rfc3820.txt`.
14. Farrell, S., Housley, R.: An Internet Attribute Certificate Profile for Authorization. RFC 3281 (2002) URL: `http://www.ietf.org/rfc/rfc3281.txt`.
15. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (1999) URL: `http://www.ietf.org/rfc/rfc2616.txt`.
16. Luotonen, A.: Tunneling TCP Based Protocols Through Web Proxy Servers. (1998) URL: `http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt`.
17. Khare, R., Lawrence, S.: Upgrading to TLS Within HTTP/1.1. RFC 2817 (2000) URL: `http://www.ietf.org/rfc/rfc2817.txt`.
18. Gauthier, P., Cohen, J., Dunsmuir, M., Perkins, C.: Web Proxy Auto-Discovery Protocol. Internet draft (1999) URL: `http://www.web-cache.com/Writings/Internet-Drafts/draft-ietf-wrec-wpad-01.txt`.
19. Netscape Communications Corporation: Navigator Proxy Auto-Config File Format. (1996) URL: `http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html`.
20. Rhee, M.Y.: Internet Security. John Wiley & Sons (2003) ISBN 0-470-85285-2.
21. Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Stewart, L.: HTTP Authentication: Basic and Digest Access Authentication. RFC 2617 (1999) URL: `http://www.ietf.org/rfc/rfc2617.txt`.
22. Glass, E.: The NTLM Authentication Protocol. (2003) URL: `http://davenport.sourceforge.net/ntlm.html`.
23. Microsoft Corporation: Microsoft NTLM. Microsoft Developer Network Library (2004) URL: `http://msdn.microsoft.com/library/en-us/secauthn/security/microsoft_ntlm.asp`.
24. Raggett, D., Hors, A.L., Jacobs, I.: HTML 4.01 Specification W3C Recommendation. (1997) URL: `http://www.w3.org/TR/html4/`.
25. Kristol, D., Montulli, L.: HTTP State Management Mechanism. RFC 2965 (2000) URL: `http://www.ietf.org/rfc/rfc2965.txt`.
26. Hickman, K.: SSL 2.0 Protocol Specification. Technical report, Netscape Communications Corp. (1994) URL: `http://wp.netscape.com/eng/security/SSL_2.html`.
27. Freier, A.O., Karlton, P., Kocher, P.C.: The SSL Protocol – Version 3.0. Internet draft, Netscape Communications Corp. (1996) URL: `http://wp.netscape.com/eng/ssl3/draft302.txt`.
28. Dierks, T., Allen, C.: The TLS Protocol – Version 1.0. RFC 2246 (1999) URL: `http://www.ietf.org/rfc/rfc2246.txt`.

A User Authentication Methods on the WWW

Authentication on a TCP/IP network should take place on the transport or application layer in order to identify a particular user (and not only her machine). We outline the most common authentication methods currently used on the WWW (see [20], [21], [22], and [23]) for a thorough treatment). Solutions based

on Java applets or ActiveX controls are deliberately omitted as they are of minor practical importance. Apart from NTLM HTTP Authentication (which is likely to fall in this category too), all methods described in the following are supported by the current prototype.

A.1 Basic Authentication and Digest Authentication

The hypertext transfer protocol offers two built-in authentication mechanisms, namely *Basic Authentication* and *Digest Authentication*. Both work in the following way: Each time the browser requests a protected resource, the web server returns a message with the status code `401 Unauthorized` indicating the need for user authentication and the method to be used. After having once prompted for a user name and password assigned to a certain *realm*, the browser has to include the same authentication information over and over again in each subsequent request of a protected resource. When using Basic Authentication, this information is transferred in the clear as part of the HTTP header – possibly multiple times – making this method extremely vulnerable to eavesdropping. In comparison, Digest Authentication is resistant to passive attacks since the password is concealed by a challenge-response protocol. The challenge consists of a server-created nonce, which has to be incorporated in the argument of a hash function (besides the user name and password) to obtain the response. Digest Authentication thus provides significantly more security concerning the authentication although it cannot withstand active attacks, nor does it provide confidentiality.

A.2 NTLM HTTP Authentication

NTLM HTTP Authentication is a proprietary authentication scheme working analogously to Digest Authentication, but requires an additional round of communication between client and server. Depending on the server settings, a client may answer the challenge with one or more responses using different algorithms [22]. The protocol is based on Microsoft's NTLM (NT LAN Manager), but platform-independent implementations of NTLM HTTP Authentication are available[12]. NTLM differs from the previous methods in that it does not authenticate single HTTP requests, but merely a whole session conveying session authentication information via the HTTP headers. Server and client are both required to support persistent connections either with HTTP/1.0 and the *keep-alive* feature or HTTP/1.1 (common web browsers do).

A.3 Authentication Based on HTML Forms

Using HTML forms [24] for user authentication is another method on the application layer and certainly the most popular nowadays. From a web designer's

[12] e.g. by Mozilla, see `http://www.mozillazine.org/talkback.html?article=3990`.

viewpoint, this is the method of choice since the authentication dialog can be embedded directly into the web page, thus avoiding extra windows popping up.

There are two transport methods for information that was entered into the form. While the GET method appends the data to the URL, the POST method transfers it in the body of the HTTP request. A disadvantage of the former method is that information may be unintentionally cached (e.g., in the browser's history or in log files on the web server or a proxy). Like all aforementioned methods, form-based authentication does not provide message confidentiality or authenticity by itself. Due to the stateless nature of HTTP, form-based authentication has to be used in conjunction with techniques for session management to convey authentication information in subsequent requests. HTTP cookies [25] or URL re-writing are typical such methods.

A.4 Authentication Using Public Key Cryptography

The methods considered so far are all knowledge-based. In order to use a stronger authentication mechanism based on the possession of a private key, SSL/TLS is the protocol of choice for the Internet. In this paper, we do not distinguish between SSL [26,27] and TLS [28], but simply speak of SSL instead as TLS v1.0 is closely related to SSL v3.0. SSL is already widely deployed, although mostly in a setting where only the server has a certificate, but users are authenticated by other means (those described in the previous sections). Let's assume that the user possess a key pair for signing and a corresponding certificate issued by a certification authority (CA) that is trusted by the server.

Before the actual data transmission takes place, client and server engage in a handshake protocol to mutually authenticate and agree upon a common secret. During the protocol, the client, which holds an RSA or DSA key pair, sends an X.509 certificate and proves its identity by signing a hash code based on the preceding messages exchanged with the server. As a result of the handshake, client and server have agreed upon a so-called premaster secret. A key for a message authentication (MAC) scheme is derived from this value. The MAC key is used to authenticate subsequent messages.

Secure Role Activation and Authorization in the Enterprise Environment

Richard W.C. Lui, Lucas C.K. Hui, and S.M. Yiu

Department of Computer Science,
The University of Hong Kong,
Pokfulam, Hong Kong
{wclui, hui, smyiu}@cs.hku.hk

Abstract. Role Based Access Control (RBAC) [3] is a popular approach to specify and enforce security policies in organizations. In large enterprise systems, the number of users, roles and permissions can be in hundreds or thousands and the security management can be a tedious task. One way to simplify the security management in RBAC is to allow the specification and the enforcement of dynamic constraints to be decentralized [7]. In this paper, we discuss the issues for supporting secure role activation and authorization when the decentralized approach to role activation management is adopted. Secure protocols are proposed to handle the processes of role assignment, role activation and authorization.

Keywords: Role Based Access Control, Role Activation, Digital Credential, Proxy Signature.

1 Introduction

Role Based Access Control. Role Based Access Control (RBAC) [3] is a popular approach to specify and enforce security policies in organizations. In RBAC, users are not directly assigned permissions but rather roles are used as the intermediary (see Figure 1). A role is a collection of permissions, which acts as an abstraction for the job duties in the organization. Users are assigned to roles based on their responsibility and qualification (U-R assignment). In addition, permission is assigned to the corresponding roles (P-R assignment). Relationships can be defined between roles to form a role-hierarchy. A role hierarchy is mathematically a partial order, where the senior roles inherit the permissions from the junior roles. A constraint imposes restrictions on the acceptable configuration of the different RBAC components. Examples of constraints include static constraints (e.g. cardinality of a role) and dynamic constraints (e.g. time constraint and dynamic separation of duty) [3].

Role Activation. In RBAC, a user does not get all the permissions of the roles assigned to him/her at the same time. In order to exercise the permission associated with an assigned role, the user should perform role activation for the corresponding role. During role activation, dynamic constraints (such as

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 22–35, 2005.

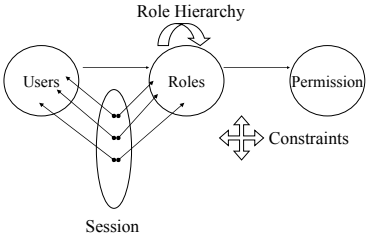

Fig. 1. Role Based Access Control Model

separation of duty and time constraints) are checked to ensure they are not violated at the time of role activation. For instance, a user may not activate the "cashier" and "cashier supervisor" role at the same time in a certain session due to the conflict of interest.

In most existing RBAC systems, the role activation process is often performed in an ad-hoc and uncoordinated manner. For instance, the role activation process may be missing (e.g. [12]) and a user is immediately granted the permission of all the assigned roles. Alternatively, the role activation process is implemented and enforced independently in each application (e.g. [13] and [2]). However, since there is no coordination between the role activation processes in different applications, it is difficult to enforce dynamic separation of duty constraints across different applications in a consistent manner. For instance, it is not easy to check whether conflicting roles are activated in different applications by a user at the same time.

As an alternative, the role activation process can be managed in a decentralized manner [7]. In this approach, an organization is divided into different administrative domains [4,6]. A domain is a group of resources (e.g. applications) with similar security requirements. Examples include departments, faculties and teams. A domain (the sub-domain) may reside in another domain (the ancestor domain), where the sub-domain will be subject to the policy of the ancestor domains. Different administrators can be assigned to specify the dynamic constraints for activating the roles in different domains. Also, the enforcement of the policy can be handled by multiple servers in a decentralized manner. Despite its flexibility, it cannot be supported by most existing RBAC systems.

Contribution. In this paper, we discuss the issues for supporting secure role activation and authorization when the decentralized approach to role activation management is adopted. For the sake of illustration we describe a protocol (the basic protocol) which is based on the use of digital credentials to handle the processes of role assignment, role activation and authorization. However, it does not support an efficient authorization process. Therefore, we combine the proxy signature [5] (which allows a user to delegate part of his signing rights to another user) and the 2Schnorr signing protocol [10] (which allows two parties to jointly produce signatures) to construct a more efficient scheme. The improved protocol

also supports more flexible management of the access rights, which cannot be supported by the basic protocol.

Organization. The rest of the paper is organized as follows: In Section 2, the decentralized approach to role activation management will be introduced. In Section 3, the issues and protocols for secure role activation and authorization will be discussed. In Section 4, role activation and authorization is supported with digital credentials (the basic protocol). In Section 5, an improved protocol to support role activation and verification is described. After that, the basic and improved protocol are compared in Section 6. Finally, in Section 7, the summary and future research directions will be given.

2 Decentralized Management of the Role Activation Process

In an enterprise RBAC system, there will potentially be a large number of roles and specifying the dynamic constraints for each role will be a tedious task. Ideally, the constraints should be defined in a way that the users are prevented from exercising any permission in an inappropriate context, but at the same time they should not be denied the permissions necessary for completing their tasks. For complex organization with many divisions (such as a bank where the divisions may be situated in different states or countries), the various divisions may be subject to different laws and may have evolved their own specific practices. Although the same role may exist across an organization, the responsibility of that role and the ways in which the associated permission are to be discharged may be quite different. For instance, a role can be activated during the office hours in a branch, but the same role can only be activated during the non-office hours in another branch. Since the security requirements of different domains/sub-domains may be different, different security administrators (having their own competence and knowledge) should be assigned to specify the dynamic constraints in the different domains.

The specification of dynamic constraints is not useful unless they are enforced. One important dynamic constraint to be handled is the dynamic separation of duty constraint [16]. To enforce dynamic separation of duty constraints, a repository which keeps track of the activated role set (ARS) for each user should be maintained. When a user intends to activate a role, the ARS should be checked to ensure that conflicting roles are not being activated at the same time. In order to enforce dynamic constraints across different applications in a consistent manner, a designated server can be assigned to handle all the role activation requests in the organization. However, a single server is not sufficient to handle the large number of role activation requests and the diversity of security requirements in the organization (e.g. the organization may intend that the activation of a critical role be activated in a more secure server).

This motivates the need for a model to support decentralized specification and enforcement of dynamic constraints [7]. In this paper, the various dynamic

constraints will be defined in the form of a role activating policy (RAP). In an organization, the RAP may be defined at multiple locations by multiple security administrators. The resources (data and applications) in the organization will be stored in multiple servers, which reside in a certain domain/sub-domain. The P-R assignment will be handled by the security administrator of the corresponding resource.

To access a resource in a certain domain, the user should activate the required roles (as defined in the P-R assignment) in one of the role activation servers (RAS) in the organization. A RAS is responsible for handling role activation requests from the users and determining if a role can be activated according to the RAP. The RAS of a given domain should enforce the RAP of that domain, as well as the RAP of the various ancestor domains. In this way, the enforcement of the role activation process can be decentralized to the various sub-domains, with the enterprise-level RAP being enforced at the same time. If the role activation is authorized, the ARS for the user can be updated to include the newly activated role.

3 Requirements for Secure Role Activation and Authorization

In most existing RBAC systems (e.g. [13] and [2]), the role activation process is implemented and enforced independently in each application. In this approach, each resource may maintain its U-R assignment and ARS. The RBAC process can be protected by simply securing the servers which host the resource. However, when the decentralized approach to role activation management is adopted, the U-R assignment database (where the U-R assignment is kept), RAS and the resource (together with the P-R assignment) may be handled by different servers. Since the various servers are usually interconnected with insecure links (e.g. in the Intranet environment), trust cannot be established unless the communication between these servers can be secured. In this paper, we address two important security requirements in supporting secure role activation and authorization.

Requirement 1: A user should not be able to activate a role at a RAS without being assigned to the role (or a senior role).

To address the requirement, a secure protocol to allow the RAS to retrieve the most updated U-R assignment information should be adopted. At the time of role activation, the RAS may initiate a secure connection to the U-R assignment database to verify the membership of the user in a role. The authenticity of the U-R assignment database should be verified to prevent malicious entities from releasing some false information about the U-R assignment. In addition, the integrity of the U-R assignment information should be protected to prevent attackers from modifying this information in transit. However, one limitation of this approach is that the RAS should contact an external server whenever role activation is to be performed and the role activation process will be inefficient.

Also, the server maintaining the U-R assignment will be a bottleneck because there may be potentially a large number of role activation requests in an organization. In addition, if an attacker is able to compromise the U-R assignment database, the attacker will be able to assign arbitrary roles to himself/herself. Since the U-R assignment will be used to perform access control in the organization, the security of the whole RBAC system will be compromised.

Alternatively, the U-R assignment information may be replicated in each of the RAS. In this approach, the role activation process is more efficient as there is no need to connect to an external server. However, synchronization of the U-R assignments among the various servers is required whenever there are updates to the U-R assignment information. Also, the security control of the RBAC system can be violated if the U-R assignment database is compromised.

As an alternative, the U-R assignment information can be represented in the form of digital credentials (e.g. attribute certificate [9], smart certificate [11]). A digital credential is an assertion describing the property of an entity with the integrity protected by digital signature (e.g. Schnorr Signature [15] and RSA [14]). In RBAC, a digital credential may be used to bind a user to one or more roles he/she is assigned to. When the user intends to activate a role, he/she may present the credential to the RAS to prove his/her membership in a role. This approach has the advantage that there is no need for the RAS to connect to an external server when performing role activation. Therefore, it supports a more efficient authorization process. Also, as the U-R assignment information is signed, it is protected from modification (even if the U-R assignment database is compromised). In this paper, we will adopt this approach to handle the U-R assignment.

Requirement 2: A user should not be able to access a resource, which requires a certain role to be activated, without actually activating the corresponding role (or a senior role) in a RAS.

To satisfy the requirement, a secure protocol for the resource to verify whether a user has activated a certain role in a domain should be adopted. To access a resource in a certain domain, the user may activate the required role in that domain (or an ancestor domain). Therefore, to verify whether the user has activated a certain role, the resource should make a secure connection to the RAS of that domain (and all the ancestor domains) where the ARS for the user is maintained. However, this process involves making a number of secure connections to external servers. Therefore, the authorization process will be inefficient (in particular, when the domain hierarchy involves many levels of domains).

Replicating the ARS in each resource is not a good solution as roles are frequently activated and deactivated. The overhead for synchronizing the ARS of a RAS and the copies kept by the resources will be high. Also, a single resource will not be interested in the role activation information for all the roles in the organization. As a result, the replication and synchronization may cause unnecessary burden to the network traffic.

Therefore, in this paper, we adopt the approach where the RAS will sign a digital credential to certify that a user has activated a role at a certain time. The user will make use of this credential to access the resource. The signature on the credential protects the integrity of the role activation information. Since there is no need for the resource to make connections to external servers when performing access control, efficient authorization can be supported.

4 Role Activation and Authorization with Digital Credentials

4.1 The Process for Role Activation and Authorization (the Basic Protocol)

In the discussion, we assume that each user and RAS in the organization is associated with a public/private key pair. In addition, we assume there is a public/private key pair (the role-assignment key pair) for certifying the U-R assignment in the organization.

Role Issuing Protocol:
A role activation certificate (RAC) is a digital credential to bind a user (with a certain public key) to a role such that only the user who has the knowledge of the corresponding private key may activate the role. To assign a user to a role, the security administrator should make use of the role-assignment private key to generate a RAC to bind the user to the role. The RAC should include a validity period during which the user may activate the role. Any attempt to activate the role using the RAC outside its validity period should be denied by the RAS. After generating the RAC for the users in the organization, the role-assignment private key should be kept offline (or in a computer not directly connect to the rest of the network) to reduce the chance of compromise by the attackers.

With the presence of role-hierarchy, a user assigned to a certain role r will also be indirectly assigned to all the junior roles in the hierarchy. In this paper, we adopt the approach where the security administrator will generate the RAC for all the roles directly or indirectly assigned to the user (i.e. a user issued a RAC for role r will also be issued the RACs for all roles $r' < r$). This approach allows the RAS and resources to verify the user's membership in a role directly or indirectly assigned without managing a local copy of the role hierarchy [1].

Role Activation Protocol:
To activate a role r, the user should authenticate himself/herself using his/her private key and present the RAC for r to the RAS. The RAS should evaluate

[1] As an alternative, only the RAC for r will be issued to the user. The RAS and the resources should keep a copy (or a part) of the role-hierarchy such that the user may activate a role $r' < r$ without the RAC for r'. This approach has the advantage that there is less RAC to be managed by the user. However, synchronization between the various copies of the role-hierarchy is required.

the defined RAP to check the compliance. If the role activation is authorized, the RAS generates an access certificate (AC) (which is a statement, signed using the private key of the RAS, to certify that the user has activated a certain role within a given time interval).

Access Protocol:
To perform operations on resources in different servers (which requires the role r to be activated), the user should authenticate himself/herself using his/her private key. Also, he/she should present the AC to the resource to show that he/she has activated r. In addition, he/she should present the RAC for r to prove his/her membership in the activated role.

4.2 Discussion

One important issue to be handled is the revocation of the RAC and the AC. The binding between users and roles in a RAC may become invalid as the responsibility of the user changes. Also, roles may be deactivated by the user when he/she completes a task or intends to activate a conflicting role to perform another task. One simple solution is to rely on a short expiry time for the certificate. However, revocation before the expiry time cannot be performed. As an alternative, an online revocation server can be set up to keep track of the credentials that are not expired but revoked.

In the proposed protocol, the role activation process is handled by the RAS and the authorization is handled by each individual resource. This provides a flexible revocation model as the handling of revocation in these two processes may be performed in different ways. For instance, the RAS can make use of online revocation servers to check the validity of a RAC if it considers the immediate revocation to be critical for activating a role. However, for the AC, since the period for activating a role is relatively short, the resource may rely only on the expiry time of the AC and there is no need for explicit revocation. As there is no need for the resource to contact a centralized server every time the resource is accessed, more efficient access control is possible.

We next analyze the trust model for the protocol. In the proposed protocol, all the U-R assignment information is signed by the role-assignment private key. By compromising the U-R assignment database, the attacker cannot modify any of the existing U-R assignment as the integrity of the RAC is protected by digital signature. As the role assignment private key is not known by the attacker (as the key is kept offline), he/she will not be able to generate any new RAC for himself/herself.

We also consider the compromise of the RAS. The private key of the RAS cannot be kept offline because it is required to sign the AC for role activation. Therefore, by compromising the RAS, the attacker can make use of the private key of the RAS to sign any AC and activate any of the roles he/she is assigned to (as he/she has the RAC for the corresponding role). Suppose the attacker intends to activate a role he/she is not assigned to directly or indirectly. Since he/she does not know the role assignment private key, he/she is not able to forge

the corresponding RAC to show that he/she is assigned to the role. Since both the RAC and AC will be verified when accessing a resource, the attacker will not be able discharge the rights associated with any of the roles he/she is not assigned to. Therefore, the potential damage caused by the compromise can be reduced.

Finally, we consider the case where the server hosting the resources is compromised. The attacker will be able to perform any operation on the resources in the server. However, as we assume the resources in the organization are distributed among many servers in the organization, the compromise of one server will not affect the resources in another server.

5 An Improved Protocol to Support Role Activation and Verification

For the basic protocol, the resource should perform three signature verification operations (for the access request, AC, and RAC) per access request. In this section, we will present an improved protocol, which is based on the 2Schnorr signing protocol [10] and the proxy signature scheme by Kim et al. [5] (which has been proved to be secure if the underlying signature scheme (i.e., Schnorr signature scheme) is secure [1]) to support more efficient access control.

5.1 Proxy Signature by Kim et al. [5]

Let p and q be large primes such that q divides $p-1$. Let g be a generator of a multiplicative subgroup of Z_p^* with order q, $h()$ denotes a collision resistant cryptographic hash function with range Z_q, $(x_A \in_R Z_q^*, y_A = g^{x_A} \ (mod\ p))$ be the key pair of Alice, and $(x_B, y_B = g^{x_B} \ (mod\ p))$ be the key pair of Bob.

Suppose Alice intends to delegate her signing right to Bob. Alice computes the proxy for Bob by generating the ephemeral key pair $(k_A \in_R Z_q^*, r_A = g^{k_A} \ (mod\ p))$ and computing the proxy $s_A = x_A\ h(w_A, r_A) + k_A \ (mod\ q)$ where w_A is the delegation warrant (which specifies the the public key of Alice, Bob, and the restrictions on the use of this delegation). Bob then verifies the proxy by checking if $g^{s_A} = y_A^{h(w_A, r_A)} r_A \ (mod\ p)$.

After the verification, Bob can generate the proxy private key $p_B = s_A + x_B\ h(w_A, r_A) \ (mod\ q)$ and the proxy public key $t_B = (y_A\ y_B)^{h(w_A, r_A)} r_A (mod\ p)$. To generate a proxy signature on a message M, Bob randomly generates an ephemeral key pair $(k \in_R Z_q^*, r = g^k \ (mod\ p))$ and uses the Schnorr signature scheme [15] to sign the message using the proxy private key p_B. By following the verification procedure of Schnorr signature, the verifier can check the validity of the signature with the proxy public key t_B.

5.2 2Schnorr Signing Protocol [10]

The 2Schnorr signing protocol is a two-party signature scheme to allow two parties, the client and the server, to jointly produce signatures. A 2Schnorr public

key is an ordinary Schnorr public key (p, q, g, y). However, the corresponding private key, x, is split between the client (x_c) and server (x_s), with $x \equiv x_s + x_c$. To sign a message M, the client selects a random element k_c and the server selects a random element k_s from Z_q. First, the server randomly generates an ephemeral private key k_s and computes the corresponding ephemeral public key $r_s = g^{k_s} \ (mod\ p)$. The server sends the client $h(r_s)$. Similarly, the client generates an ephemeral private key k_c and computes the corresponding ephemeral public key $r_c = g^{k_c} \ (mod\ p)$. The client sends $\{h(r_s), r_c, M\}$ to the server. The server checks that r_c belongs to the group specified by p, q and g by verifying the equality $r_c^q \ (mod\ p) = 1$. After that, the server computes $r = r_c r_s \ (mod\ p)$ and $s_s = k_s + x_s h(M, r) \ (mod\ q)$. The server replies to the client $\{r_c, r_s, s_s\}$.

The client computes $h(r_s)$ and verifies that it matches the value received in the first message. The client checks that r_s belongs to the group specified by p, q and g by using a method similar to the server. Finally, the client computes $s_c = k_c + x_c h(M, r) \ (mod\ q)$ and computes $s = s_c + s_s \ (mod\ q)$. The pair (r, s) is an ordinary Schnorr signature of M.

5.3 Protocol for Role Activation and Authorization

We denote the role-assignment key pair to be $(x_d, y_d = g^{x_d} \ (mod\ p))$, the personal key pair for user u to be $(x_u, y_u = g^{x_u} \ (mod\ p))$ and the key pair for the RAS to be $(x_a, y_a = g^{x_a} \ (mod\ p))$. In the discussion, we assume that all the RAS share the same key pair for encryption and decryption purposes. In this way, a user assigned a role can activate the role by using the RAS of all the domains in the organization. Alternatively, the RAC can be encrypted with a key shared by only a subset of the RAS in which the user may perform role activation for the corresponding role. Also, we denote $E(m, y)$ to be the encryption of m with the public key y, $S(m, x)$ be the signature of m using the private key x, and $a||b$ to be the concatenation of a and b.

Role Issuing Protocol:
Suppose the security administrator intends to assign a role r to a user u. To generate the RAC, the security administrator first computes a proxy for u by performing the proxy issuing protocol as described in Section 5.1. He/She randomly generates an ephemeral key pair $(k_d \in_R \ Z_q^*, r_d = g^{k_d} \ (mod\ p))$ and uses the role-assignment private key to generate a proxy

$$s_d = x_d h(w_d, r_d) + k_d \ (mod\ q)$$

where w_d is the delegation warrant (which specifies the delegator (role-assignment public key), the public key of the delegate y_u, the role r to be assigned to the end-user u, and the validity period). The proxy is encrypted with the public key of the RAS to form $E(s_d, y_a)$. The RAC is formed by $E(s_d, y_a)||w_d||r_d||y_d$. In this way, only the corresponding RAS is able to decrypt and obtain the proxy. Similarly, the RACs for all $r' < r$ should be generated for the user.

Role Activation Protocol:
Suppose user u intends to activate the role r' where $r' \leq r$. He/She presents the RAC for r' to one of the RAS. The RAS decrypts the encrypted proxy to obtain s_d. The RAS should check that $s_d = y_d^{h(w_d,r_d)} r_d \ (mod\ q)$. If the activation of the role does not violate the defined RAP, the end-user first randomly generates a temporary key pair ($x_j \in_R \ Z_q^*, y_j = g^{x_j} \ (mod\ p)$). Then, the RAS and the user cooperate to sign on a message of the form delegation warrant w_u (where user u further delegates the role r' he/she is assigned by the role-assignment key pair to the public key of the temporary key pair y_j for a certain period). First, the RAS randomly generates an ephemeral key pair ($k(S) \in_R \ Z_q^*, r(S) = g^{k(S)} \ (mod\ p)$). Similarly, user u also randomly generates an ephemeral key pair ($k(C) \in_R \ Z_q^*, r(C) = g^{k(C)} \ (mod\ p)$). The RAS and user u agree on the value $r = r(C)r(S) \ (mod\ p)$ using the 2Schnorr signing protocol. In the signing phase, the RAS uses s_d as the server private key to sign on w_u to form

$$s(S) = s_d h(w_u, r) + k(S) \ (mod\ q).$$

Meanwhile, user u uses $x_u h(w_d, r_d)$ as the client private key to sign on w_u to form

$$s(C) = x_u h(w_d, r_d) h(w_u, r) + k(C) \ (mod\ q).$$

The RAS sends $s(S)$ to user u. The proxy signature can be formed by (r, s_u) where

$$\begin{aligned} s_u &= s(S) + s(C) \ (mod\ q) \\ &= (s_d + x_u h(w_d, r_d)) h(w_u, r) + k(S) + k(C) \ (mod\ q) \\ &= (p_u) h(w_u, r) + k \ (mod\ q). \end{aligned}$$

Access Protocol:
To discharge the permission associated with the role r', the user u computes the proxy private key

$$p_j = s_u + x_j h(w_u, r) \ (mod\ q)$$

and uses it to sign an access request. The user sends to the resource $w_u||r||w_d||r_d||y_j||y_d||y_u$. The proxy public key used to verify the access request is computed by

$$t_j = ((y_d y_u)^{h(w_d,r_d)} r_d y_j)^{h(w_u,r)} r \ (mod\ p).$$

The resource should check that y_u is included in w_d and y_j is included in w_u. From this proxy public key, the resource can verify that the user receives role assignment from the role assignment key pair (the public key of the user is included in w_d), the U-R assignment is valid (the current time is within the validity period specified in w_d) and the role is currently activated (the current time is within the validity period specified in w_u).

5.4 Analysis of the Improved Protocol

First, we consider the role issuing protocol. The security administrator uses the role-assignment private key x_d to issue a proxy to certify that user u is assigned

to role r using the proxy signature scheme from [5]. Since x_d is only known by the security administrator, any other user will not be able to forge a non-existing U-R assignment if the underlying Schnorr signature scheme [15] is secure.

Since the proxy s_d is encrypted with the public key of the RAS, user u cannot make use of the proxy on its own. Therefore, using the proxy to sign messages requires the cooperation of the RAS. In the role activation protocol, s_d is revealed to the RAS. As y_j is specified in the delegation warrant w_u to be the delegate, based on the unforgeability property [8] of the proxy signature scheme, only the designated signer who knows the corresponding private key (x_j) will be able to discharge the rights associated with the proxy. Therefore, the RAS cannot make use of permission of the role, which is associated with the proxy, by itself.

Suppose a role activation request is authorized, the RAS and user u cooperates to generate s_u (which is a signature on delegation warrant w_u) using s_d. By the security of the 2Schnorr signing protocol, it is not necessary for the RAS to expose the server private key (s_d) when performing signing and the RAS does not gain knowledge of the temporary private key x_j. Therefore, user u is required to contact the RAS every time he/she intends to activate a role.

The basic and the improved protocol adopt a similar trust model. For user-role assignment, there is no single point of attack if the role assignment private pair x_d is kept offline after the generation of the proxy. In case the RAS is compromised, the decryption key x_a for the encrypted proxy will be known by the attacker. However, the attacker cannot make use of the proxy directly if he/she does not compromise also the private key for the corresponding user. Therefore, the attacker is not able to activate an arbitrary role (in which he/she is not assigned to) for himself/herself. Note that this cannot be achieved by the original proxy signature scheme [5], where a delegate (the RAS) can make use of the delegated permission associated with the assigned roles directly.

6 Comparison of the Basic and the Improved Protocol

Similar to the basic protocol, the revocation in the improved protocol can be supported by the use of short validity times and online revocation servers. However, the improved protocol supports more efficient authorization and better management of access rights.

In both of the protocols, dedicated role activation servers are deployed to handle the process of role activation. A user may activate a role only once and exercise the permissions which are associated with the role in multiple applications. When compared with the traditional RBAC approach, where the process of role activation process is handled by each individual application [2], there is no need for the users to activate a role multiple times in order to access the various applications when performing their tasks. Also, since the role activation requests can be handled by multiple RAS in the organization, scalability and fault tolerance can be achieved. For details on how the use of RAS may support scalability, fault tolerance and better security management, please refer to [7].

We also analyze the performance of the protocols. For the role issuing protocols, one signing operation is required for both the basic and improved protocol to generate the RAC and proxy respectively. In the improved protocol, one additional encryption operation is required to protect the confidentiality of the proxy. For the role activation protocols, in the basic protocol, one signing operation is required by the user to authenticate to the RAS. The RAS should perform two signature verification operations (one for verifying the authenticity of the user and the other one for the validity of the RAC) and one signing operation for the generation of the RAC. Meanwhile, in the improved protocol, one signing operation is required by the user to compute the client partial signature. The RAS should perform one decryption operation for the encrypted proxy, one signature verification operations for the proxy, and one signing operation for computing the server partial signature. Finally, for the access protocol, the verifier (the resource) should perform signature verification operations for the access request as well as the AC and the RAC (i.e. three signature verification operations have to be performed to determine if a user has the authorization to perform the access) in the basic protocol. In contrast, in the improved protocol, it is just required for the verifier to perform one signature verification operation for the access request with the proxy public key t_j. From the proxy public key, the verifier can be convinced that the end-user has the required U-R assignment and role-activation if he/she can sign an access request using the corresponding proxy private key. Since the number of access operations is usually larger than the number of role issuing operations (the permission associated with a role may be discharged multiple times) and role activation operations (a role may be activated once for access to multiple resources), the increased efficiency in the access protocol is critical for the performance of the RBAC system.

The improved protocol also supports better management of access rights. We denote the set of RAS in the organization to be S. In the basic protocol, we assume that a user may perform activation of a role in any of the RAS in S. The resource will only grant access if the AC is issued by a certain subset $S' \subseteq S$ of the RAS which are trusted by the resource. In the improved protocol, for each role r, the security administrator may also specify a certain subset $S'' \subseteq S$ of the RAS trusted to activate the role r by encrypting the proxy with the public key of the corresponding RAS in S''. As a result, only the RAS in S'' is able to perform decryption, obtain the proxy and perform role activation. Therefore, in order to discharge the rights associated with that role in the resource, the user should choose a RAS in $S' \cap S''$.

Also, in the basic protocol, if the user has to activate multiple roles to perform a task, the user has to keep his/her private key, as well as the RAC and AC for the various roles. In contrast, in the improved protocol, the end-user may activate multiple roles and delegate to the same temporary key pair (x_j, y_j). In this way, the user can travel around and make use of the permission associated with the various activated roles by just keeping a single key x_j (e.g. in his handheld device). Delegation of the permissions for the various activated roles

can be performed by simply revealing x_j to the delegate. These features are not supported by the basic protocol.

7 Summary and Future Research Directions

In this paper, we discuss the issues for supporting secure role activation and authorization when the decentralized approach to role activation management is adopted. In particular, we describe a protocol which is based on the use of digital credentials to handle the processes of role assignment, role activation and authorization. However, it does not support an efficient authorization process. Therefore, we combine the proxy signature and the 2Schnorr signing protocol to construct a more efficient scheme. When compared with the basic protocol, the improved protocol also supports better management of access rights in the organization.

This paper focuses on the process of role assignment, role activation and authorization. Despite its flexibility, decentralized role activation management cannot be supported by most existing RBAC systems. A possible research work is to design a RBAC system to support decentralized approach to role activation management. In particular, the system should take a number of important issues into account. For instance, the RBAC system should ensure that only the authorized users can manage the security policy in the organization, the various RAS should be able to synchronize the role activation information in a secure manner, and the secure binding between a RAS and a certain domain should be possible. In the system, a language should also be defined to specify and enforce the security policy and role activation constraints.

Acknowledgement

This research is supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-01/99), two grants from the Research Grants Council of the HKSAR, China (Project No. HKU/7144/03E and HKU/7136/04E), and two grants from the Innovation and Technology Commission of the HKSAR, China (Project No. ITS/170/01 and UIM/145).

References

1. Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *http://eprint.iacr.org/curr/*, 2003.
2. David F. Ferraiolo, John F. Barklery, and D. Richard Kuhn. A role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security, Vol. 2, No. 1, February 1999, Pages 34-64*, 1999.
3. David F. Ferraiolo, D. Richard Kuhn, and Ramaswamy Chandramouli. Role-based access control. *Boston : Artech House*, 2003.

4. Michael Hitchens, Vijay Varadharajan, and Gregory Saunders. Policy administration domains. In *ACISP*, pages 286–302, 2002.
5. S. Kim, S. Park, and D. Won. Proxy signatures, revisited. In *Information and Communications Security (ICICS'97), LNCS 1334, pp. 223-232, 1997. Berlin: Springer-Verlag*, 1997.
6. Gunhee Lee, Wonil Kim, Dong-Kyoo Kim, and Hongjin Yeh. Effective web-related resource security using distributed role hierarchy. In *WAIM*, pages 87–96, 2004.
7. Richard W.C. Lui, Sherman S.M.Chow, Lucas C.K. Hui, and S.M.Yiu. Role activation management in role based access control. In *the Tenth Australasian Conference on Information Security and Privacy (ACISP05), Brisbane, Australia, 4-6 July 2005, Lecture Notes in Computer Science (LNCS), Springer*, 2005. To appear.
8. Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures: Delegation of the power to sign messages. In *IEICE Trans. on Fundamentals, Vol.E79-A, No.9, pp.1338-1354*, 1996.
9. José A. Montenegro and Fernando Moya. A practical approach of x.509 attribute certificate framework as support to obtain privilege delegation. In Sokratis K. Katsikas, Stefanos Gritzalis, and Javier Lopez, editors, *EuroPKI*, volume 3093 of *Lecture Notes in Computer Science*, pages 160–172. Springer, 2004.
10. A. Nicolosi, M. Krohn, Y. Dodis, and D. eres. Proactive two-party signatures for user authentication. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium, pages 233– 24 , February 2003*, 2003.
11. Joon S. Park and Ravi S. Sandhu. RBAC on the web by smart certificates. In *ACM Workshop on Role-Based Access Control*, pages 1–9, 1999. citeseer.nj.nec.com/park99rbac.html.
12. Joon S. Park, Ravi S. Sandhu, and SreeLatha Ghanta. RBAC on the web by secure cookies. In *DBSec*, pages 49–62, 1999.
13. R. Sandhu R. Chandramouli. Role based access control features in commercial database management systems. In *21st National Information Systems Security Conference, October 6-9, 1998, Crystal City, Virginia*, 1998.
14. R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. Technical Report MIT/LCS/TM-82, 1977.
15. C.P. Schnorr. Efficient identification and signatures for smart cards. *Advances in Cryptology – CRYPTO '89, Lecture Notes in Computer Science, Vol. 435, pp. 239-252, Berlin: SpringerVerlag*, 1990.
16. Richard T. Simon and Mary Ellen Zurko. Separation of duty in role-based environments. In *IEEE Computer Security Foundations Workshop*, pages 183–194, 1997.

Towards a Unified Authentication and Authorization Infrastructure for Grid Services: Implementing an Enhanced OCSP Service Provider into GT4

Jesus Luna[1], Manel Medina[1], and Oscar Manso[2]

[1] Polytechnic University of Catalonia, Computer Architecture Department, Jordi Girona 1-3 08034 Barcelona, Spain
{jluna, medina}@ac.upc.edu
[2] CertiVeR, Technical Director, Diputacion 238, 08007 Barcelona, Spain
o.manso@certiver.com

Abstract. The OGSA definition of a Grid Service as a transient, stateful and dynamically instantiated Web Service introduced new authentication and authorization requirements beyond those already established for existing Grid environments. However such design features have begun to be developed currently following a pre-Web Services approach in two aspects: in the first place making a clear separation of authentication from authorization issues, and in the second place not designing them over the OGSI/WSRF defined mechanisms and specifications. In this paper we are proposing a new Security Framework that unifies identified common points of both features, Authentication and Authorization, into a mechanism called validation policy which is expected to improve service performance and security. Our framework seeks to implement these aspects over the Grid Service's Operations and Service Data concepts to fully exploit its functionalities. The paper also presents the integration of an enhanced OCSP Service Provider into the Globus Toolkit 3.9.4 as a first proof of concept.

1 Introduction

The "Physiology of the Grid" [1] introduced a clear separation between the protocols and messages required for interoperability among virtual organization (VO) components, and the nature of the *services* responding to those messages. The Grid was envisioned as an extensible set of *Grid Services* able to be aggregated in various ways to meet the needs of VOs, which themselves can be defined in part by the services that they operate and share. The *Open Grid Services Architecture* (OGSA) was defined as the alignment of Grid and Web Services, defining a Grid Service as a *transient, stateful and dynamically instantiated Web Service* that provides a set of well-defined interfaces and that follows specific conventions. This specification was standardized by the Global Grid Forum as OGSI (*Open Grid Services Infrastructure*), whose interfaces addressed the features shown in table 1.

Grid Services as defined by OGSI expose two basic mechanisms for interaction: *operations* (grouped into *portTypes*) and *service data (*composed of Service Data Elements -SDEs-). Later on we will return to these concepts and its advantages for a Grid Services' oriented Authentication and Authorization Infrastructure (AAI).

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 36 – 54, 2005.

Table 1. Features addressed by OGSI interfaces

Feature	Explanation
Discovery	Addresses those mechanisms needed for discovering available Grid Services and for determining their characteristics, so that they can configure themselves and their requests appropriately.
Dynamic service creation	Focuses on dynamic creation and managing of new service instances according to the OGSA model.
Lifetime management	Provides those mechanisms needed for reclaiming services and states associated with failed operations.
Notification	Allows Grid Services to communicate with each other asynchronously about interesting changes to their state.
Other interfaces	Addresses issues related with security, concurrency control and monitoring of potentially large sets of Grid Services instances.

OGSA's envisioned convergence between Grid Services and Web Services began to be achieved when a new standard called WSRF (*Web Services Resource Framework* [2]) was born. The first WSRF implementation will be in the Globus Toolkit 4 – GT4- (announced for the end of April 2005). From the security point of view, GT4 follows the traditional modular-oriented design from the pre-Web Services Globus Toolkit's versions so its Grid Security Infrastructure (GSI) implementations of Authentication (AuthN) and Authorization (AuthZ) features are independent, in such a way that the output from the first is used for the decision taking process in the former. Even though the previous approach has proved efficient in several ways, this paper will introduce our work in progress about the following two ideas:

1. Build a unified AuthN and AuthZ Framework – called AA Framework – designed to take advantage of Grid Services' features shown in table 1 and,
2. Optimize and enhance the security of AuthZ related processes (mainly related to the initiator's ability to invoke operations and access SDEs) through the use of new user-side and server-side AA rules defined in a *validation policy* built upon this unified Framework. As a proof of concept the CertiVeR *enhanced* OCSP Service Provider implementation into GT4 will be presented.

The rest of this paper is organized as follows: section 2 will review important design features required by AuthN and AuthZ mechanisms aimed for Grid Services; section 3 explains in more detail the basic ideas behind our proposal; section 4 reviews our experiences in implementing the CertiVeR enhanced OCSP Service Provider in GT4 as a first effort to build the unified AA Framework; later section 5 will review the main features of related AuthN and AuthZ Infrastructures currently available for Grid environments; and finally section 6 presents future work and conclusions.

2 Designing Authentication and Authorization Mechanisms for Grid Services

Unique features related with Grid environments in general and Grid Services in particular, pose some special needs on those mechanisms aimed to provide them with

Authentication and Authorization solutions. In this section we will summarize relevant Grid Services' AA design features and challenges; even though we are taking the mentioned traditional approach of considering them independent systems, in the next section this analysis will provide the base to introduce the benefits of a unified conceptual AA Framework.

2.1 Grid Services Authentication

Traditionally Grid environments have relied on Public Key Infrastructures (PKI) to provide Authentication services in a distributed way at the transport layer (via TLS) and message layer (digitally signing SOAP messages), even though implementations like the GT4's Grid Security Infrastructure [3] have also implemented non-PKI solutions (i.e. username and password mechanisms).

Despite its security advantages, PKI mechanisms have also generated the need to consider various design features to improve existing or build new Grid Services Authentication Infrastructures. As a first approach to ease their use, we have classified these features in the categories shown in table 2.

2.2 Grid Services Authorization

The Global Grid Forum introduced a conceptual Authorization Framework for Grid environments, which integrated several elements that we have considered important to review in the first part of this section as they will be extended in the second part to fit other authorization needs that we have considered relevant for distributed environments. Both the basic framework and our proposed extensions shall be considered as candidates to integrate with the authentication features from table 2 towards generating a unified AA Framework for Grid Services.

2.2.1 A Conceptual Grid Authorization Framework

This section will not cover a comprehensive analysis of such framework [5]. Instead it will focus specifically on the features needed to establish our proposal presented in section 3. First of all, some basic terminology will be introduced. Three basic entities are considered: *Subject*, *Resource* and *Authority*.

The component performing the evaluation of the executable policy by computing an authorization decision on behalf of the authorities is sometimes referred to as an AuthZ Server. Typically this entity may do a combination of: an authorization decision, an authorization lookup, and the delegation or proxy of an authorization decision to another AuthZ Server.

Also three models of authorizations (push, pull, agent and hybrid sequences) are presented, but at this moment we will focus our explanation on the Grid pre-WS pull sequence shown in figure 1.

This framework introduced two access control functions:

- Access control decision function (ADF): Makes authorization decisions about a subject's access to a service. It is equivalent to the Policy Decision Point (PDP) defined in [6].
- Access Control Enforcement Function (AEF): Mediates access to a resource or service. It is equivalent to the Policy Enforcement Point (PEP) defined in [6] also.

Table 2. Design features for Grid Services Authentication Infrastructures

Feature	Explanation
X.509 Credentials life-cycle management.	Deals with the way in which the PKI treats the issuing, publishing, status checking, revocation or cancellation, and renewal of End Entity Certificates (EECs) and Proxy Certificates. Special care should be taken when dealing with Proxy Certificates revocation as this is a source of potential security risks.
Single Sign-On.	Allows Grid Users to introduce their authentication credentials only once during Proxy Certificate lifetime. Also important to consider is the Single Sign-Off.
Delegation.	Also implemented through the use of standard X.509 Proxy Certificates that allow bearers of X.509 EECs to delegate their privileges temporarily to another entity. Limited delegation (i.e. Restricted Proxies) must be considered also.
Identity Federation.	An inherent feature of VOs is the fact that a Grid user may have more than one and possibly different identities on participant computer systems. Limited identity federation should also be considered for those systems in which the user does not want to Single Sign-On.
Trust conditions.	For an authentication service to be trusted by relying partners a level of trust has to be established and maintained. This applies mainly to X.509 credentials and participating PKI security features.
Privacy and anonymity.	The Authentication Infrastructure could support pseudononymous identifiers in conversations between the Grid's origin and target sites, thus protecting user's privacy. Also support for limited privacy could also be provided (only certain user identity's attributes being released).
Interoperability and extensibility.	Use of open standards is advisable.
Authentication Architecture.	The Authentication Infrastructure shall provide Grid Services with a reliable, scalable and fault tolerant service.
User-side and Server-side Authentication Policies.	Policies ruling Grid Service Authentication on the user-side (i.e. limited Single Sign-On and limited Delegation) and the server-side (i.e. trust conditions) are needed to enable a comprehensive security system. Both types of policies should be established only by the Grid user's home organization and the user itself, this may optimize its retrieving and authorization procedures (contrary to Authorization Policies, which are typically distributed as we shall see later). Policy management issues (writing, publishing, distribution, deletion, etc.) should be considered as well.

Table 2. *(Continued)*

Use of formal methods.	Designing sound and correct AuthN protocols and policies based on formal methodologies will improve its security features (i.e. using the BAN Logic [4]).
Authentication traffic.	Dense Grid environments may consume a lot of network bandwidth during AuthN processes, so a careful design of protocols shall be taken into account.

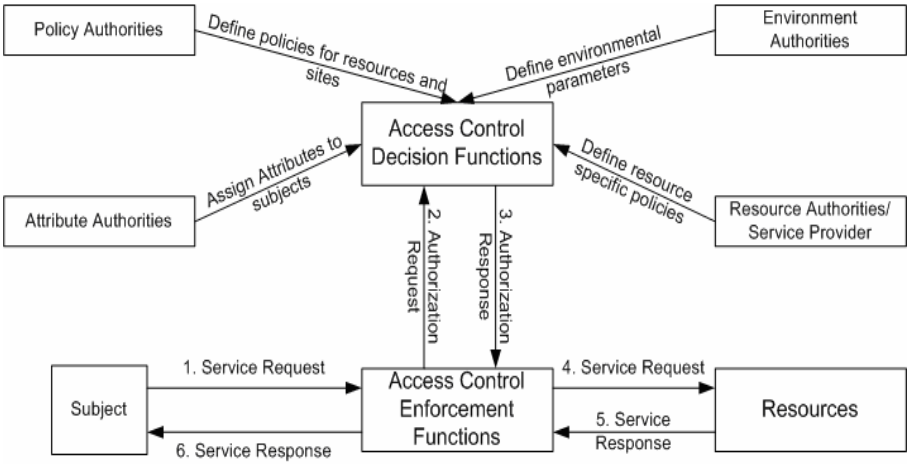

Fig. 1. Pull Sequence Authorization Architecture

Obviously the ADF, AEF, Subject and Resources may be embedded inside one or more administrative domains in a variety of combinations. In any case, there are three categories of information that may need to be passed between the subject, resource and the various attribute authorities: *Attributes, Policy flow* and *Authorization queries and responses*.

Table 3 presents the framework's components, which will be extended later in the next subsection.

2.2.2 Proposed Extensions for the Conceptual Grid Authorization Framework

The framework presented in the previous section not only settled authorization design features observed in existing Grid AuthZ systems or required by new ones, but also was built to be flexible enough so extensions can be added to cover security needs particular to certain environments, just like the new Grid Services reviewed at the beginning of this paper. Figure 2 and 3 precisely show OGSA's envisioned Push and Pull authorization sequences.

Table 4 contains a set of proposed extensions that we have considered important for Grid Services Authorization.

Table 3. Conceptual Grid Authorization Framework components

Feature	Explanation
Trust Management	In this framework trust management defines authorities shown in figure 1 and specifies what they should be trusted to do.
Privilege Management	Privilege management covers the definition, assignment, storage, presentation, delegation and revocation of both privilege and descriptive attributes.
Attribute Authorities	An attribute is granted to some entity by an authority for the relevant home domain of the entity. Sources Of Authority (SOA), their delegates and the domains that will accept the attributes must have a common understanding of the authority's scope. This should be expressed in a privilege management policy.
Privilege Assignment	This operation describes the process of defining who is allowed which access rights. Privileges can be assigned (or revoked) by issuing a policy component describing direct access rights to a subject or by Role-based Authorization mechanisms (which are clearly an emerging direction in Grid computing).
Attribute assertions management	Attribute assertions are proofs of the right to assert a descriptive attribute or privilege attribute. Centralized and distributed management features shall be considered
Policy Management	Policy management for static and even dynamic resources shall be defined.
Authorization Context	This consists of those properties of the Authorization Request which are neither provided via Authorization Attributes nor included in Authorization Policies, but which are relevant to the decisions made by the Authorization Server. The Authorization Context may include environment information and authentication information.
Authorization Server	The Authorization Server is an entity that evaluates authorization requests and issues responses, taking into account relevant attributes, policies and environmental parameters. A single AuthZ response or a set of AuthZ responses is typically the output from these algorithms.
Enforcement Mechanisms	These mechanisms limit the operations performed on resources on behalf of a subject to those permitted by an authoritative entity.

The previous table introduced some features that we have identified to improve Grid Services AuthZ operation's performance and security.

However let us emphasize the *validation policy,* which is the basis of our proposed unified AA framework because of the seamless integration of several authentication

and authorization operations that we are expecting to provide in its design in the form of rules. A rationalization about such a proposal is given next.

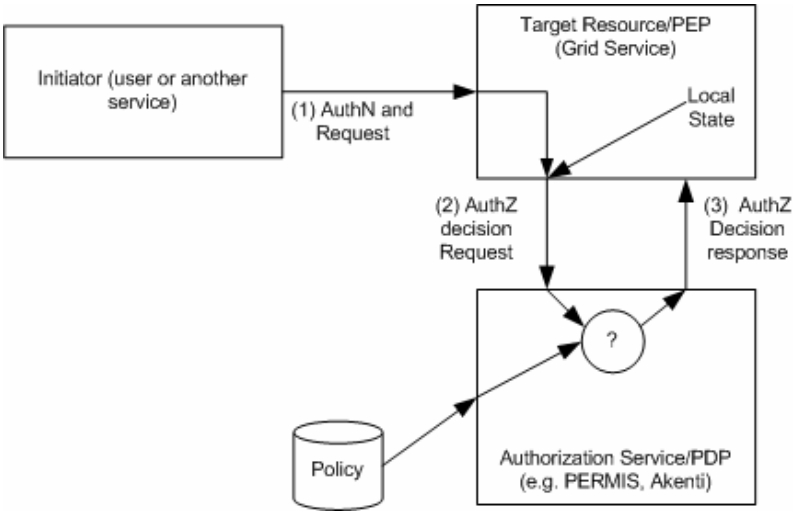

Fig. 2. OGSA Authorization Pull Model

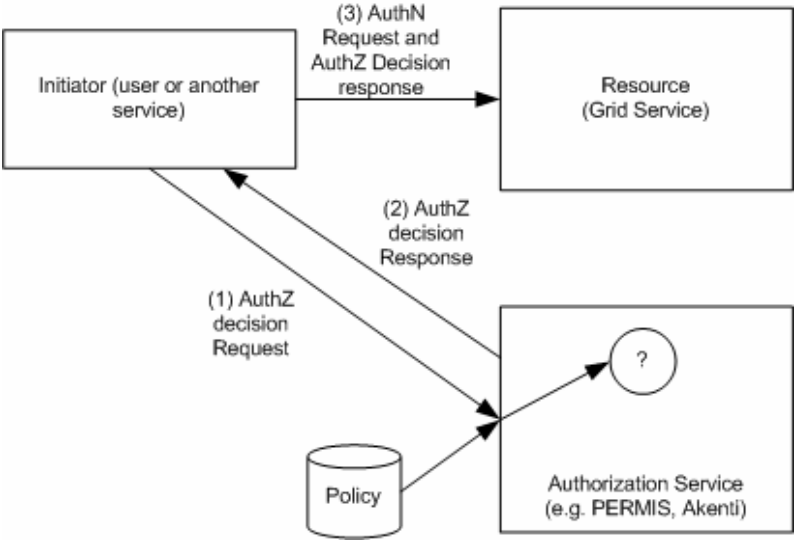

Fig. 3. OGSA Authorization Service Decision Push Model

Table 4. Proposed extensions for the Grid Services Authorization Framework

Feature	Explanation
Interoperability and extensibility.	While this is an open issue, it is preferred to implement AuthZ mechanisms able to interoperate with others from different VOs while maintaining its capacity to support additional features as needed. Worth mentioning is the SAML-callout AuthZ support integrated in GT4.
Use of formal methods.	Formal methodologies should be a preferred way for designing sound and correct AuthZ protocols and policies, in such a way that secure implementation can be obtained.
Policy writing.	A standard, generic, extensible and flexible language, able to create expressive AuthZ policies that may be understood by humans and computers shall be chosen at design time. Notions of hierarchies on domains, groups and roles like in PONDER [7] could be also supported by the preferred language. Support should be also considered for expressing obligation policies that are event triggered condition-action rules for policy based distributed systems.
Distributed Policy Management.	Grid environments, where more than one administrator in even different domains may control resource access, pose an additional challenge on AuthZ policy creation, discovery, retrieving, revocation and evaluation.
Subject-side and Resource-Side Authorization Rules.	Most Authorization rules for Grid Services are resource-side oriented, which means that the user can not determine his preferences about a particular set of resources in which he may be/may not be interested in for processing his job. Those subject-side AuthZ rules may be determined by the subject's home organization privacy policies, operating environment parameters, security parameters and so forth. Under our perspective these subject-side and resource-side AuthZ rules can be expressed in a Validation Policy, just as explained in the following sections.
Authorization Architecture and Performance.	Grid Services require an AuthZ architecture able to provide reliable, fault tolerant and scalable services. Also to consider are dense Grid environments that may consume a lot of network bandwidth during AuthZ flows, so a careful design of protocols used for this purposes shall be taken into account, while keeping a balance between performance and security (i.e. mutual AuthN protocols between PDP and clients, assertion signing, etc.).
Authorization assertion's security.	Closely related to the previous category we have found that AuthZ assertions traveling back and forth between Grid Service entities need to implement security related features such as integrity, non-repudiation and even confidentiality.

Table 4. *(Continued)*

Common Authorization schemes for Grid Services Operations and SDEs.	First mentioned in [8], Grid Service Operation's access refers to specify AuthZ policies on arguments of the Grid Service's operating invocation requests. Implementations around CAS [9] and SAML [10] are being focused toward this. Another facet of Grid Service Authorization encompasses policies expressed for both on specific SDEs and on subsets of the SDE space. Accesses should also be in the form of read and write access, however this raises issues like policies needing to identify both the Grid Service with which the SDE is associated and the SDE set being accessed.
Session-based Authorization.	Mentioned in [11], if initiators need to perform a series of operations on the target, then sending common information (i.e. initiator's details) only once to the AuthZ Server may optimize the AuthZ decision-making process.
Conditional replies.	Also mentioned in [11], AuthZ decisions need to be able to express not only permit or deny, but conditional policies in situations where the authorization service may not have sufficient information to make a decision.

3 Why Unify Authentication and Authorization Frameworks for Grid Services?

Previous sections introduced important AuthN and AuthZ design features for Grid Services environments in general, taking the classical approach of studying them as independent security issues. However, features like *delegation, session-based AuthN* and *AuthZ assertion's security* are so closely related in terms of authentication and authorization that it has become quite difficult to determine the exact location of these concepts in tables 2 to 4. This may cause not only design time confusions, but also implementation ambiguities resulting in poor performances and duplication of work. These problems motivate us to propose a unified AuthN and AuthZ Framework for Grid Services.

In other words our hypothesis establishes that important authentication and authorization optimizations can be achieved if common information to both frameworks is managed as a unified security feature in a validation policy that can also enhance Grid Service's security by including rules related to security parameters not considered anywhere else.

Such a validation policy can be thought of as a contract agreed before hand between interested parties (i.e. subject itself, subject VO and resource administrators). These rules must be satisfied in order to grant the requested action on the resource. The validation policy will contain rules from our proposed Unified AA Framework classified in two main sections:

1. Subject related validation rules: these will transport information related to user preferences on AA aspects like proxy certificates (i.e. renewal and limited delegation), use of roles (i.e. claimed or certified required by the resource), SOA

Location Hints (i.e. which Attribute Authorities should be contacted), End Entity Certificate validation (i.e. authorized OCSP responders), obligation policies referring to user-defined triggered actions (a concept that first appeared in [8]) and so forth.

2. Resource related validation rules: introduce verification of security parameters related with the resource (i.e. process user's job only in those systems with a determinate TCSEC classification), message and transport level security required (i.e. signature policies agreed before hand for signing SOAP messages with a proposed XAdES Profile [12]) and even environmental parameters related with resource availability (i.e. current processor load or available bandwidth).

Depending on the AuthZ model being implemented (figures 4 and 5) the steps to authorize a subject are:

1. The subject initializes a Proxy Certificate based on an agreed validation policy embedding signed evidence (i.e. OCSP Responses from authorized responder) and the validation policy URI being used in the `ProxyCertInfo` extension field. This is shown as step (1a) in both figures.
2. An action is requested to the Target Resource by the Proxy -step (1b) in figure 4-.
3. The Target Resource –figure 4, step (2b)- or the subject -step (1b) in figure 5- contacts the Authorization Service and request verification of the validation policy by supplying signed evidence (i.e. the Proxy Certificate itself).
4. The Authorization Service retrieves the validation policy (using the supplied URI and the Policy Repository) and also additional evidence from SOAs (i.e. environmental parameters referring to the target resource) to fully verify validation policy compliance (step 2a in both figures).
5. A final AuthZ decision is returned to the Target Resource –step (3) in figure 4- or the subject itself –step (2b) in figure 5-.

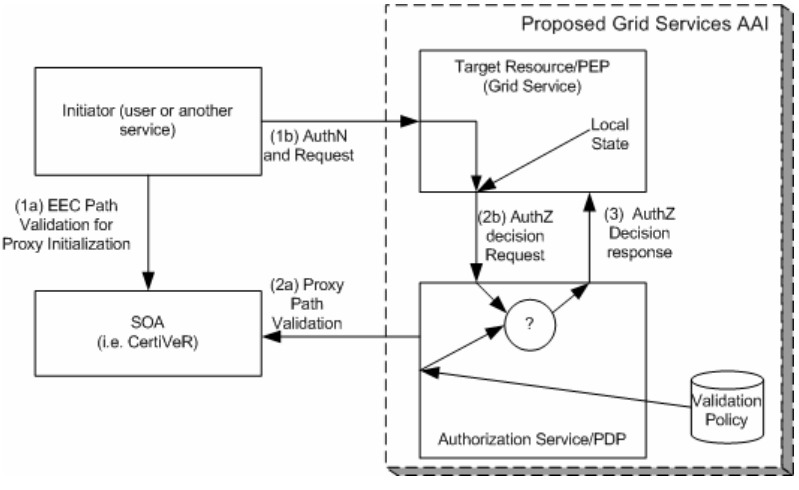

Fig. 4. Authorization Pull Model using CertiVeR and proposed Grid Service's AA

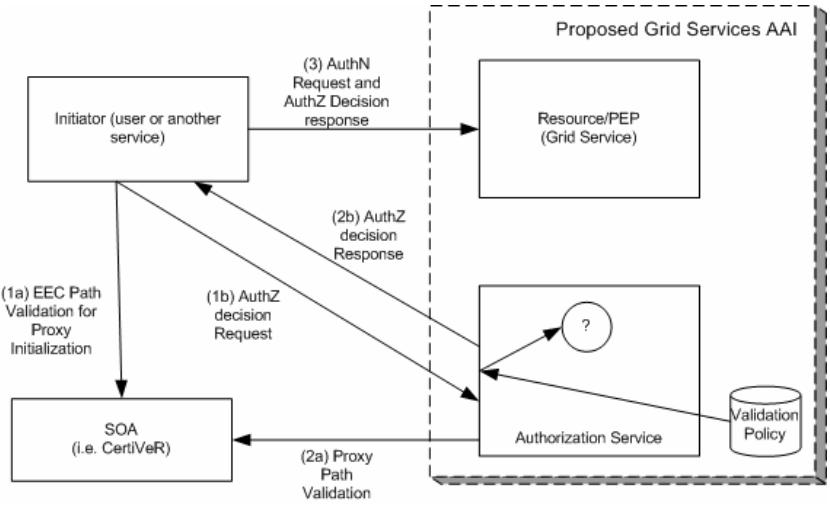

Fig. 5. Authorization Push Model using CertiVeR and proposed Grid Service's

For performance reasons we are taking the following measures in our designs:

- Build our AAI upon the concepts of operations and service data as defined by OGSI and implemented upon WSRF for Grid Services.
- Retrieve from our SOA – the enhanced OCSP Service Provider called CertiVeR – as much information as possible to verify the validation policy; embedding it into OCSP Response extensions (section 4 explains a proof of this concept).

Some final words about our unified AA framework:

- It integrates common AuthN and AuthZ related Grid Services operations into one validation policy, which means that the problem of supporting distributed AuthZ policies can be solved.
- It is sustained on standardized protocols. In our design, even though the SOA being used is implemented over not widely used concepts (i.e. OCSP Extensions) it is fully compliant with its recommendation (i.e. RFC 2560). Another example is the concept of validation policy which is related to ETSI's Signature Policy [13].
- We are committed to keep a balance between security and performance so our validation policy is not to deliver any more complexity to the Grid Services AAI world.

As we will mention in section 6, future work around this topic will be focused on researching those AuthN and AuthZ features which may be candidates to include in our unified AA Framework and subsequently in the validation policy definition. As a proof of concept, the next section presents the work that we have done to implement an *enhanced OCSP Service Provider* into GT4, so as we mentioned above, its responses may be able to transport validation information embedded into the OCSP Response extensions field, thus resulting in important optimizations.

4 An Enhanced OCSP Service Provider for the GT4

This section will cover the work done to date in the arena of implementing CertiVeR's enhanced OCSP Service Provider into GT4, as a Java based client-side module (when creating Proxy Certificates).

4.1 The CertiVeR Enhanced OCSP Service Provider

CertiVeR [14] is an EU funded project that offers a comprehensive validation service, that, apart from providing validity information of a X.509 certificate in real time through the Online Certificate Status Protocol[1] –OCSP-, it can also provide the following information:

1. *A customizable set of extensions on the OCSP response*, for example it can report the reliability of the validity response (i.e. Very-High: Maximum Delay –md- of 5 minutes, High: md of 1 hour, Medium: md of 1 day, and Low: md of 1 week or more) or the degree of trust in the issuing authority of the certificate (i.e. Gold: highly trusted registry and revocation procedures, Silver: highly trusted registry procedures, Bronze: low confidence registry procedures). These types of information may dramatically increase security and e-Trust.
2. *CertiVeR also allows the building of trust chains* between ACs connected to this service, as OCSP responses can be signed with a certificate issued by the same CA hierarchy as the certificates whose status is being asked for.

These features implement an *Enhanced OCSP Service Provider* –figure 6- that can be used by Grid Services environments not only to build trust relationships between PKIs that participate on the VO, but also to transport AuthZ related information in the AuthN protocol as explained below.

4.2 GT4 Implementation

CertiVeR´s OCSP API integration into GT4 seeks not only to create Proxy Certificates with the validation information mentioned in the section 3, but also looks for eliminating compromised or revoked End-Entity Certificates (EECs) - *and even Proxy Certificates* - from Grid environments.

The implementation process consisted in two tasks:

1. Integrate CertiVeR's Java API into the method used by the COmmodity Grid Kit –COG- Jglobus API [15] to validate the X.509 Certificate Path (already implemented on the `org.globus.gsi.proxy.ProxyPathValidator` class): This allow us to verify against our enhanced OCSP the EEC's certificate chain before generating the Proxy Certificate. WSRF Java Core also benefits from this code because it uses the same classes.
2. Integrate CertiVeR's C API into the GTK 3.9.4 C Core's Proxy Certificate Validation function (work in progress): With this modification Proxy's certificate path is validated prior to executing any task.

[1] RFC2560.

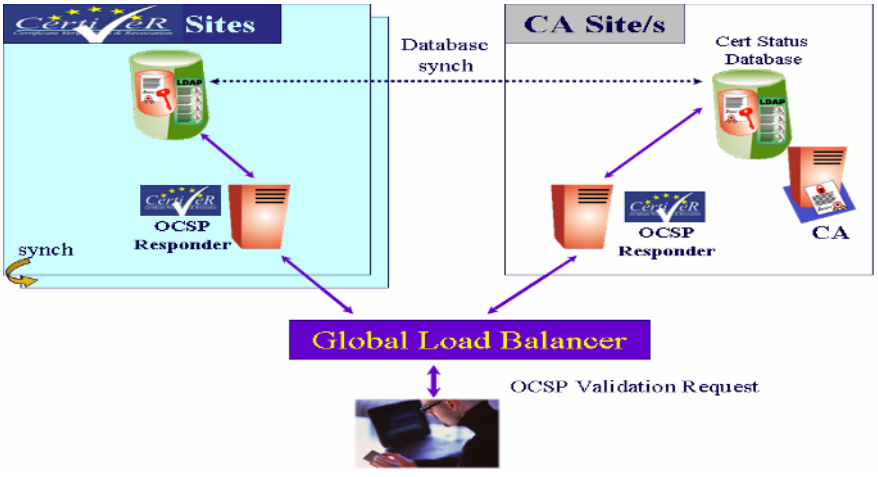

Fig. 6. CertiVeR enhanced OCSP Service Provider architecture

Section 6 explains most of our work in progress related with OCSP Response extensions being proposed for Grid Services AA tasks.

4.3 Results

As mentioned in the previous section, CertiVeR's Java API has been integrated into CoG Jglobus 1.2 Proxy initialization classes in such a way that EECs path validation is done through OCSP. For this scenario we have the following conditions:

- Validating two Grid user's certificates (one Valid and one Revoked) issued by a corporate CA which publishes its CRLs in a LDAP directory. The certification path length was 2 (which means that 3 certificates were being sent to CertiVeR's for validation, that is the Root CA[2], the subordinate CA and finally the EEC).
- Grid CoG software installed in a Linux-based PC which connects to CertiVeR's OCSP Responder (http://ocsp.certiver.com:7070) through an Internet link (shared ADSL@512/128 Mbps channel).
- Each test sequentially created sets of 50 Grid Proxy Certificates (25 for each user certificate) at eight different times during a day. This test was repeated during 5 days.
- Proxy creation time has been measured with and without validating against CertiVeR's.
- When using CertiVeR's, Proxy Certificates was created with 2 extensions parsed from a *signed and enhanced* OCSP Response.
- Also was measured in similar conditions CertiVeR's response time with the OpenSSL command line utility.

Figure 7 shows the average results obtained from the tests executed.

[2] Note that a self signed root certificate is not a certificate in the traditional meaning of the term.

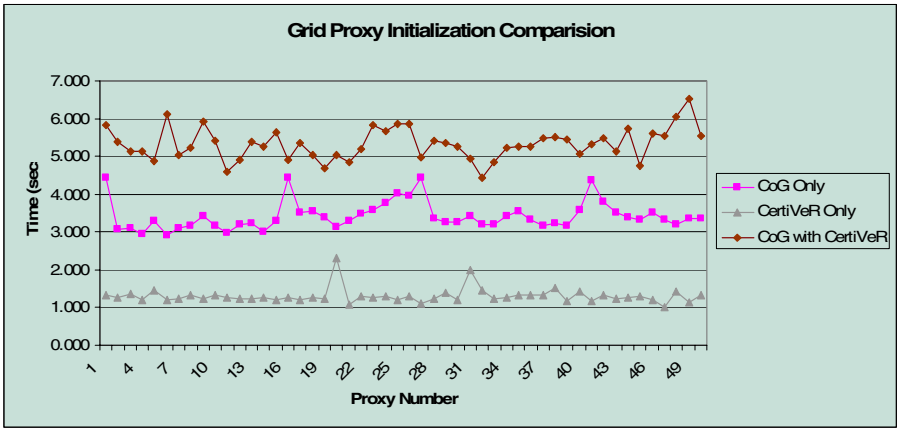

Fig. 7. Average results

The following numbers were also calculated from figure 7:

- Average Proxy creation time without CertiVeR: 3.424 secs
- Average Proxy creation time with CertiVeR: 5.326 secs.
- Average OpenSSL client validation with CertiVeR: 1.309 secs.

From figure 7 we have also observed that CertiVeR-only tests (OpenSSL client) were having a few peak values, which is due mainly to CertiVeR's CRLs refreshing process. We are also working toward improving this. On the other hand, peak values on the CoG-Only results may have been caused by system current usage. Even with these performance issues, it is important to note that most of the Proxies were created in between 5 and 6 seconds when using CertiVeR. Finally, we observed that all OCSP responses were correct for all queries (right certificate status was returned always).

5 Related Work

Grid Services oriented Authentication and Authorization Frameworks and implementations are still at their beginnings, as much of the related work is based on pre-Web Services versions of the Globus Toolkit. In this section we will review the Akenti, PERMIS, Shibboleth, CARDEA and VOMS systems as our design is related and they are also being integrated with the Grid Services environment, thanks in part to mechanisms like the *SAML AuthZ callouts* in GT4.

Akenti [16] is an authorization infrastructure from the Lawrence Berkeley National Laboratory in the USA, which can be considered a trust management infrastructure or even an AAI as most of its security is build around a previous authentication process. Akenti represents the authorization policy for a resource as a set of (possibly) distributed and hierarchical certificates (called Use-condition certificates and Policy Certificates) digitally signed by unrelated stakeholders from different domains. The policy certificates are independently created by authorized stakeholders. When an authoriza-

tion decision needs to be made, the Akenti policy engine gathers up all the relevant certificates for the user and the resource, verifies them, and determines the user rights with respect to the resource. Akenti integration with Grid environments appears in [17].

PERMIS [18] is an authorization infrastructure from the EC funded PrivilEge and Role Management Infrastructure Standards validation (PERMIS) project. As in the case of Akenti, PERMIS is also considered a trust management infrastructure. Its compliance checker is invoked as a Java object, and credentials are built according to the X.509 standard. Being authentication agnostic, PERMIS leaves to the application the task to determine what type of mechanism should be used for this function. Its policies are held in one policy X.509 Attribute Certificate and has implemented role based access controls. A paper about experiences on implementing the PERMIS AuthZ Infrastructure into the GT 3.3 through the SAML based authorization API has already been published [19].

Shibboleth [20] is a joint project of Internet2/MACE (Middleware Architecture Committee for Education) and IBM. It aims to develop an architecture for standard-based vendor-independent web access control infrastructure that can operate across institutional boundaries. The focus of Shibboleth is on supporting inter-institutional authentication and authorization for access to web-based applications. The intent is to build upon existing heterogeneous security systems in use on campuses today, rather than mandating particular schemes. In this system the origin site (where the browser user belongs) is responsible for authenticating the user, and for providing attribute information about the user to the target. The target site (where the web resource manager belongs) is responsible for comparing these statements against the policy rules associated with the desired resource. Both sites will need to satisfy themselves as to the true origin of a request or a set of assertions. Shibboleth will use pseudononymous identifiers in conversations between the origin and target sites, thus protecting user's privacy. Currently the NFS Middleware Initiative (NMI) Grant called "Policy Controlled Attribute Framework", seeks to allow the use of Shibboleth-transported attributes for Authorization in NMI Grid built upon GT4; this integration has been named *GridShib* [21].

Cardea [22] is a distributed authorization system developed as part of the NASA Information Power Grid, which dynamically evaluates authorization requests according to a set of relevant characteristics of the resource and the requester rather than considering specific local identities. Potentially resource accesses within an administrative domain are protected by local access control policies, specified with the XACML [23] syntax, in terms of requester and resource characteristics. Further, potential users are identified by X.509 proxy certificates but only according to the characteristics they can reliably demonstrate. The exact information needed to complete an authorization decision is assessed and collected during the decision process itself. This information is assembled appropriately and presented to the PDP that returns the final authorization decision for the actual access requests together with any relevant details. The system is currently implemented in JAVA, and contains a SAML PDP, one or more Attribute Authorities, one or more PEP, one or more references to an Information Service (IAS), a XACML context handler, one or more XACML PAP and

a XACML PDP. Although all these components may be co-located on the same machine to use local communication paradigms, they may also be distributed across several machines and their functionality exposed as web service portTypes.

VOMS (Virtual Organization Membership Service) [24] introduced an authorization point of view, in which Grid Authorization is established by enforcing agreements between Resource Providers (RPs) and VOs, where in general, resource access is controlled by both parties with different roles, and indeed the main difficulty is to clearly separate these two roles. To solve this apparent dualism, VOMS classified the authorization information in: i) general information regarding the relationship of the user with his VO (groups he belongs to, roles he is allowed to cover and capabilities he should present to RPs for special processing needs); and ii) information regarding what the user is allowed to do at a RP owing to his membership of a particular VO. In VOMS the first kind of information is contained in a server managed by the VO itself, while the second is kept at the local sites, near the resources involved and controlled by extended Access Control Lists (ACL). The VOMS architecture uses the authentication and delegation mechanisms provided by the GSI and is focused on the frameworks of the DataGrid and DataTAG Pojects.

6 Future Work and Conclusions

This paper has introduced the need for a unified authentication and authorization framework which uses Grid Services features defined by OGSI's interfaces to implement a new AAI tailored for this environment. Our goal is to implement a validation policy that includes both subject related AA rules and resource related AA rules to enhance Grid Services performance associated with AuthN and AuthZ operations, while improving system-wide security. Such validation policy can be verified through signed evidence and other information provided in the OCSP Responses Extensions field from CertiVeR.

As a proof of concept this paper presented the implementation of CertiVeR's enhanced OCSP Service Provider into GT4 to grant not only real time X.509 credential validation to both Grid clients and authorities, but also to build trust chains between participant PKIs and to distribute information on OCSP Responses. These features were integrated as an *enhanced OCSP Service Provider*, whose functionality has already been implemented on the Path Validation module of CoG's Proxy initialization class.

Our approach still has some open issues that must be researched deeper to obtain the Grid Service AAI envisioned by our proposal. The main topics that should be addressed as future work includes:

- The enhanced OCSP service still needs to improve its performance as the validation process increments by about 50% the Proxy initialization time. However we have to considerer that OCSP Response extension's may be able to reduce subsequent AuthZ information retrieving and decision taking; future testing will help us decide about this.
- CertiVeR's API integration into GT 3.9.4 C Core is about to come in the following weeks.

- We are about to collaborate with Global Grid Forum's CA Operations Work Group on finishing the document "OCSP Requirements for Grids" [25] which recommendations are expected to be integrated into the CertiVeR API as soon as possible.
- Subject related validation rules are still being researched and we expect to begin its implementation based on ETSI's Signature Policies [26], and possibly the XAdES profile to sign Grid Service's SOAP messages [12], thus supporting our unified AA framework.
- Related with the former, Proxy Behavior Rules (mainly related to Limited Delegation) are still being researched as performance issues related with policy flow and processing.
- In general, Validation Policy Rules definition and in particular OCSP responses extensions are being revised to better fit our AAI requirements.
- Comparison of our proposal against other OCSP implementations and providers into GT4 is on the way, even though we are not aware of any other solutions that make use of Validation Policies as defined in this paper or even convey information into OCSP Response extensions.
- Future papers shall update our research on general issues about the proposed AAI for Grid Services, which is expected to take advantage of mechanisms like OGSI's Operations and Service Data; and even GT4's AuthZ SAML callouts.

Even though our proposal is aimed to Grid Services as it is designed to fulfill the requirements of such environments –mainly security, fault tolerance and high performance-, Web Services in general -not only those based on WSRF- can also benefit themselves from our AAI thanks to features like the enhanced EEC validation (through CertiVeR's service) and the reduction of the AuthZ traffic by using the concept of a Validation Policy. Our proposed AAI is being built upon standard protocols (i.e. SOAP) and easy-to-use APIs (available in Java and C), so its adoption can be done through a simple implementation that requires a minimum or even zero changes on already deployed applications.

On the other hand, development activities will continue towards providing new AA modules for existing GT 3.9.4. Finally, a fully functional prototype of our AAI proposal is expected in the next months as we are also waiting for the final release of GT4 in April 2005; however we are expecting no major modifications to our current designs and implementations.

References

1. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration". Foster, I., Kesselman, C., Nick, J. and Tuecke, S. Globus Project, 2002, http://www.globus.org/research/papers/ogsa.pdf
2. "The WS-Resource Framework". http://www.globus.org/wsrf/
3. "Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective". The Globus Security Team. Version 2. December 2004. http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf

4. "A Logic of Authentication", Burrows, M., Abadi, M., and Needham, R. M. ACM Transactions on Computer Systems, vol. 8, no. 1, Feb 1990, pp. 18-36. A Formal Semantics for Evaluating Cryptographic Protocols p 14 http://citeseer.ist.psu.edu/burrows90logic.html
5. "Conceptual Grid Authorization Framework and Classification", R. Baker, L. Gommans, A. McNab, M. Lorch, L. Ramakrishnan, K. Sarkar, and M. R. Thompson Global Grid Forum Working Group on Authorization Frameworks and Mechanisms. February 2003, http://www.ggf.org/Meetings/ggf7/drafts/authz01.pdf
6. "RFC 2904: AAA Authorization Framework". Vollbrecht J, et. al. August 2000.
7. "The Ponder policy specification language". N. Damianou, N. Dulay, E. Lupu, and M. Sloman. In Morris Sloman, editor, Proc. of Policy Worshop, 2001, Bristol UK, January 2001. http://citeseer.ist.psu.edu/damianou01ponder.html
8. "Use of SAML for OGSA Authorization". Von Welch, et. al. OGSA-Security Working Group, Global Grid Forum. Working document. February 2003. http://www.cs.virginia.edu/~humphrey/ogsa-sec-wg/
9. "A Community Authorization Service for Group Collaboration". Pearlman, L. et. al. Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02). IEEE Computer Society. 2002.
10. "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V1.1". OASIS Security Services Technical Committee. Version 1.1. September 2003. http://www.oasis-open.org/committees/security/
11. "OGSA Authorization Requirements". Welch, Von. Et. al. OGSA-Authorization Working Group, Global Grid Forum. Working document. June 2003. https://forge.gridforum.org/projects/ogsa-authz/document/
12. "OASIS: DSS Profiles Discussion Document". Cruellas, Juan Carlos. et. al. Working Draft 02, February 2004. http://www.oasis-open.org/committees/dss/
13. "ETSI TR 102 041: Electronic Signatures and Infrastructures (ESI); Signature Policies Report". ETSI ESI group. Version 1.1.1. February 2002. http://portal.etsi.org/esi/elsign.asp
14. "CertiVeR: Certificate Revocation and Validation Service". http://www.certiver.com
15. "A Java Commodity Grid Kit" Gregor von Laszewski, Ian Foster, Jarek Gawor, and Peter Lane, Concurrency and Computation: Practice and Experience, vol. 13, no. 8-9, pp. 643-662, 2001, http:/www.cogkit.org/.
16. "Certificate-based Authorization Policy in a PKI Environment". M. Thompson, A. Essiari, S. Mudumbai. ACM Transactions on Information and System Security, (TISSEC), vol. 6, Issue 4, November 2003 pp: 566-588, http://dsd.lbl.gov/security/Akenti/Papers/
17. "Distributed Security Architectures – Providing security services for Grids". M. Thompson, et. al. Working Document. March 2004. http://dsd.lbl.gov/security/SciDac-DistSec-2pager04.pdf
18. "The PERMIS X.509 Role based privilege management infrastructure". David Chadwick, Alexander Otenko. 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002). June 2002. http://www.acm.org/sigs/sigsac/sacmat
19. "Experiences of Applying Advanced Grid Authorisation Infrastructures". R. O. Sinnott, A. J. Stell, D. W. Chadwick, O. Otenko. Accepted for the European Grid Conference (EGC2005), 14th-16th February 2005, Science Park Amsterdam, The Netherlands. http://labserv.nesc.gla.ac.uk/projects/dyvose/publicity/eGridFinalv4.pdf
20. "Shibboleth Overview and requirements". Steven Carmody. Shibboleth Working Group Overview and Requirements Document, February 2001. http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html
21. "GridShib: A Policy Controlled Attribute Framework". http://grid.ncsa.uiuc.edu/GridShib/

22. “Cardea: Dynamic Access Control in Distributed Systems”. NASA Advanced Supercomputing Division. Technical Report. November 2003. http://www.nas.nasa.gov/News/Techreports/2003/PDF/nas-03-020.pdf
23. “OASIS eXtensible Access Control Markup Language (XACML) 2.0”. Tim Moses, et. al. Committee Draft 4. December 2004. http://www.oasis-open.org/committees/xacml/
24. “VOMS, an Authorization System for Virtual Organizations”. R. Alfieri,. et. al. Presented at the 1st European Across Grids Conference, Santiago de Compostela, Spain. February 2003. http://infnforge.cnaf.infn.it/voms/VOMS-Santiago.pdf
25. “OCSP Requirements for Grids”. Global Grid Forum, CA Operations Work Group. Working Document. February 2005. https://forge.gridforum.org/projects/caops-wg
26. “ETSI TR 102 038: Electronic Signatures and Infrastructures (ESI); XML format for signature policies”. ETSI ESI group. Version 1.1.1. April 2002. http://portal.etsi.org/esi/elsign.asp

A Heterogeneous Network Access Service Based on PERMIS and SAML

Gabriel López[1], Óscar Cánovas[1], Antonio F. Gómez-Skarmeta[1], Sassa Otenko[2], and David W. Chadwick[2]

[1] University of Murcia, Spain
[2] University of Kent, United Kingdom
{gabilm, skarmeta}@dif.um.es, ocanovas@ditec.um.es
{o.otenko, d.w.chadwick}@kent.ac.uk

Abstract. The expansion of inter-organizational scenarios based on different authorization schemes involves the development of integration solutions allowing different authorization domains to share, in some way, protected resources. This paper analyzes different emerging technologies. On the one hand, we have two XML-based standards, the SAML standard, which is being widely accepted as a language to express and exchange authorization data, and the XACML standard, which constitutes a promising framework for access control policies. On the other hand, PERMIS is a trust management system for X.509 attribute certificates and includes a powerful authorization decision engine governed by the PERMIS XML policy. This paper presents a sample scenario where domains using these technologies can be integrated allowing, for example, the use of attribute certificates in a SAML environment and the utilization of the PERMIS authorization engine to decide about the disclosure or concealment of attributes. In order to design this scenario we have based our work on a Credential Conversion Service (CCS) which is able to convert ACs into SAML attributes, and a User Attribute Manager (UAM) which controls the disclosure of credentials. These modules are governed by policies defining the conversion process (the Conversion Policy) and the disclosure of attributes (the Disclosure Policy).

1 Introduction and Rationale

Nowadays, authorization systems are more and more complex. They span domains of administration, depend on many different authentication sources, and the management of permissions and policies can be as complex as the system itself. Worse still, while there are many standards defining authentication mechanisms, the standards for authorization systems are less widely adopted and accepted, and tend to work only within homogeneous systems.

Today we often experience inconvenience and inefficiency like this: a professor of *University A* visiting *University B* is not allowed to use the latter's network even when there is an existing collaboration by means a research project between the two institutions. The main reason might be the use of different formats for

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 55–72, 2005.

representing the authorization information (credentials, policies, requests, etc.), and this could occur despite an existing service level agreement (SLA), a high level collaboration, between the two universities. Therefore, heterogeneity restricts the development of standard management tools and toolkits that serve common policy needs.

Authorization in a distributed system often requires cooperation among separate and autonomous administrative domains. Maintaining a consistent authorization strategy requires each system to maintain at least some knowledge of its potential collaboration with other domains. Therefore, any authorization decision that spans several authorization domains requires each party to be able to produce, accept and interpret authorization information from a group of potentially heterogeneous peers. This property can be achieved by establishing an agreement on protocols, syntax and semantics of shared pieces of authorization data to be exchanged.

In this paper we present a case study where we demonstrate how a PERMIS [6] based administrative domain can be integrated with a network access service based on SAML [13] attributes and XACML [9] policies. Our aim is to provide a feasible scenario allowing the PERMIS users to make use of the foreign network implementing a network access control mechanism based on the AAA (Authentication, Authorization, Accounting) architecture and SAML attributes [12], hereafter the NAS-SAML domain. This integration, which will be based on existing proposals like the Credential Conversion Service [7], illustrates that we can build an interdomain environment from separate and autonomous domains, which constitutes a valuable step towards the required interoperability between multiple existing authorization environments.

As we will show, while Role Based Access Control (RBAC) is an increasingly important component in distributed systems, it is one that is often hard to support in heterogeneous environments. In our proposal, this leads to the definition of loosely coupled multidomain environments, where a predefined set of role mappings to mediate interdomain accesses is defined. This approach requires the constituent systems to indicate the level of sharing they want to allow and to establish a consistent set of mediation rules for interdomain accesses. As we stated before, SAML is one example of a protocol that provides a framework for secure assertion of authorization data across domains, and it will be used in our proposal since it constitutes a key element of the Credential Conversion Service.

The main idea behind this paper can be summarized as follows. A PERMIS user willing to make use of the NAS-SAML domain has to demonstrate that he has gained the required X.509 attribute certificates. Those certificates should be provided by the PERMIS domain, and they must be translated into SAML credentials before processing the network access request. Therefore, from the architectural point of view, we have to define the entities (pertaining to the home and target domains) that will be involved in the integration process. From an operational point of view, we have to establish the way the attribute certificates will be disclosed and exchanged, and how they will be converted into equiva-

lent SAML attributes. As we will see, several design alternatives (push and pull based) are provided.

One central issue that our design addresses is the management of attributes and the circumstances under which a PERMIS attribute authority should disclose the user's attributes to another site. We define some simple rules about when and to whom the attributes may be revealed. Although this idea has been widely explored in the literature [15,3], our approach tries to minimize the impact of adding an overloading privacy management system to the PERMIS home domain. As we show, that privacy system will be built using the PERMIS technology itself, adapting the PERMIS policy to produce a disclosure policy that can be enforced using the PERMIS ADF (Authorization Decision Function). In this way, we make our infrastructure resilient in defence against security attacks such as data-mining, that is, the unauthorized gathering of information for improper use. Consequently, we designed a system that allows the PERMIS domain to have full control over the private information being disclosed. That is, administrators are given the opportunity to decide the kind of information to share with the foreign network provider.

Finally, we also present a simple conversion policy based on XACML that will be used to translate X.509 attribute certificates into SAML Attribute Statements. That conversion policy will be managed at the destination site (the NAS-SAML domain) and will be expressed in terms of the object identifiers related to the attributes, attribute values, SAML attribute designators, etc. XACML constitutes an appropriate tool to express these kinds of policies, as we will show in the following sections.

The rest of this paper is structured as follows. Section 2 provides an overview of the two different authorization scenarios involved in this paper, the PERMIS project and the SAML-based network access service. Section 3 describes the architectural elements, the disclosure policy and the conversion policy. Then, Section 4 presents the two different design alternatives based on pull and push models. Next, Section 5 includes the related work that informed our research. Finally, we conclude the paper with our remarks and some future directions.

2 Authorization Systems

This section provides an overview of the two different authorization scenarios that have been integrated in this paper.

2.1 SAML-Based Network Access Service

In [12] we present a network access control approach based on X.509 identity certificates and authorization attributes, which addresses some of the challenges derived from the integration of existing authentication systems with a flexible, scalable and manageable authorization system. The proposal is based on the SAML and the XACML standards, which will be used for expressing access control policies based on attributes, authorization statements, and authorization

protocols. Authorization is mainly based on the definition of access control policies including the sets of users pertaining to different subject domains which will be able to be assigned to different roles in order to gain access to the network of a service provider, under specific circumstances. The starting point is a network scenario based on the 802.1X standard [14] and the AAA (Authentication, Authorization and Accounting) architecture [8], where we centralize all the operations related to authentication, authorization, and accounting.

The system operates as follows. Every end user belongs to a home domain, where he was given a set of attributes stating the roles he plays. When the end user requests a network connection in a particular domain by means of a 802.1X connection, the request is obtained by the AAA server, and it makes a query to obtain the attributes linked to the user from an authority responsible for managing them. Finally, the AAA server sends an authorization query to a policy decision point, and that element provides an answer indicating whether the attributes satisfy the resource access policy. Furthermore, that policy can also establish the set of obligations derived from the decision, for example some QoS properties, security options, etc. This general scheme works both in single and inter-domain scenarios, and using both push an pull based communications.

2.2 PERMIS Project

PERMIS [6] is a trust management system. It uses X.509 Attribute Certificates to specify subject attributes such as roles and permissions, and defines a hierarchical role based access control (RBAC) policy language in terms of those roles and permissions. The PERMIS policy specifies who is to be granted what type of action on which targets, and under what conditions. Policy based authorization on the other hand allows the domain administrator (the SOA) to specify the policy for the whole domain, and all targets will then be controlled by the same set of rules. The policy is expressed in XML and comprises the following components:

- SubjectPolicy: this specifies the subject domains, that is only users from a subject domain may be authorized to access resources covered by the policy.
- RoleHierarchyPolicy: this specifies the different roles and their hierarchical relationships to each other.
- SOAPolicy: this specifies which SOAs are trusted to allocate roles.
- RoleAssignmentPolicy: this specifies which roles may be allocated to which subjects by which SOAs.
- TargetPolicy: this specifies the target domains covered by this policy. Each domain is specified as an LDAP subtree or using URIs (Uniform Resource Identifier).
- ActionPolicy: this specifies the actions (or methods) supported by the targets, along with the parameters that should be passed with each action.
- TargetAccessPolicy: this specifies which roles have permission to perform which actions on which targets, and under which conditions. Conditions are specified using Boolean logic and might contain constraints involving strings, integers, dates, or boolean expressions.

On the other hand, the privilege verification subsystem is responsible for authenticating and authorizing the remote user and providing access to the target. The primary component is the application gateway. The functionality of this is split into two components: an application-specific component termed the Access Control Enforcement Function (AEF), and an application-independent component termed the Access Control Decision Function (ADF). An application programmable interface (APIs) between the AEF and ADF has been defined based on the AZN API [10].

3 Architectural Elements and Policies

The integration of different authorization scenarios involves the definition of new components which act as a bridge between the participating domains, hiding from the rest of the components the knowledge of different authorization mechanisms. Those new components must respect the already defined systems and their components, and they should be able to interact in the most transparent way. Beside these components, it is necessary to define the policies used for the disclosure and conversion processes. This section describes the components, their functionality, and gives an overview of the needed policies.

The next figure shows an application scenario where two domains, using different authorization mechanisms, exchange credentials to deny or grant access to the required services.

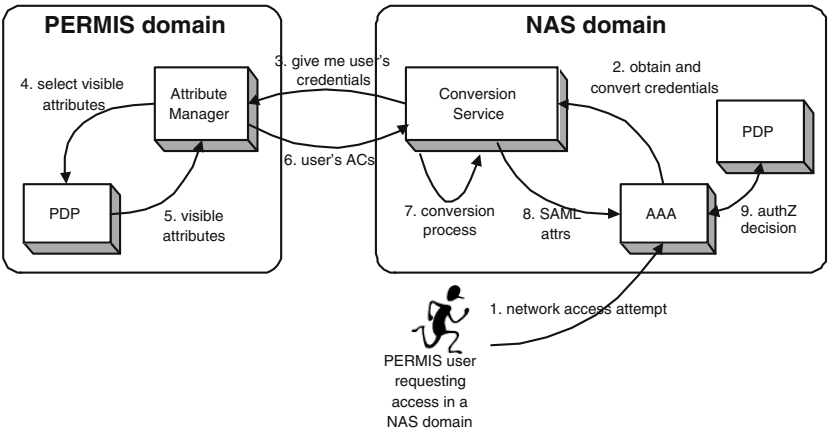

Fig. 1. Credential Conversion Scenario

3.1 New Components

Regarding the scenario described in Figure 1, several issues must be addressed. On the one hand, the user's home domain needs a module able to receive credential requests from an external domain, and to decide which of the user's attributes

must be revealed. For example, in the proposed scenario, the PERMIS domain needs a module able to manage the ACs requests from the NAS-SAML scenario. On the other hand, the NAS-SAML domain needs a component responsible for recovering from an external domain the user attributes, which are represented in a source format (for example X.509 ACs), and for translating them into internal credentials, represented in a target format, in this case SAML.

Those modules are the UAM (User Attribute Manager), used to deal with the attribute requests received from an external domain, and the CCS (Credential Conversion Service), which is in charge of translating the authorization credentials.

User Attribute Manager (UAM). One issue in distributed systems that serve users from multiple communities is determining which organization a particular user is from and hence the organization whose attribute authority can provide attributes regarding the user. This is often referred to as the *Where are you from?* problem. Although this could be implemented placing a pointer to the attribute authority in the user's X.509 identity certificate, this solution requires cooperation of the CA issuing the user's identity credentials, which may not always be available and also binds attribute information to the user's identity credential, which may raise problems if the lifetimes of these two elements differ. Our initial system implementation either assumes fixed UAM locations dependent on the requester or discovery via an information service query to a trusted source.

In the proposed scenario, the UAM is a module able to understand queries expressed in SAML, and able to create authorization responses in that format. Moreover, this module should return to the NAS-SAML domain only the visible credentials specified by the disclosure policy. The UAM module is defined with this intention, but its particular behavior will depend on the communication model.

When the pull model is used, the UAM receives attribute queries from the target domain, specifically from the CCS. The UAM obtains the user's attributes, for example, from an internal repository, and asks the local PDP (Policy Decision Point) about the visibility of those attributes. This module enforces a disclosure policy establishing which attributes will be revealed for that domain. Once the decision is obtained, the UAM returns a response message to the CCS containing the user's attributes in source format, in this case, X.509 ACs.

In the push model, the UAM receives the attribute query directly from the end user. Then, the UAM returns the allowed attributes, following the same process as described before, to the end user. Once the user has validated the attributes and, depending on the communication between entities, he may request the UAM to forward the conversion query to the appropriate CCS module, or he may present these attributes directly to the target domain. Different design alternatives are described in Section 4.

Credential Conversion Service (CCS). The CCS [7] integrates external authorization schemes (non SAML-based) into authorization scenarios which make

use of SAML as the main language for assertions. Our starting point is an end user requesting access to a resource secured in a SAML environment. We do not want the user authorities pertaining to non-SAML domains to issue SAML assertions, since they were not designed to perform that task. In fact, what we need is a service able to translate the external credentials into SAML assertions. CCS defines the different architectural entities involved in that process and their relationships. Moreover, it extends some standard SAML elements, such as SAML assertions and queries, to provide the needed syntax and semantics.

The CCS module is located in the NAS-SAML domain and it receives attribute conversion queries related to a foreign user.

In a pull model, where the user's attributes must be obtained by the target AAA server, it asks the CCS for those attributes, and the conversion process will be responsible for obtaining and translating them into the internal format. Following the described scenario, when the CCS receives an attribute query from the local AAA server, it has to discover the user's home domain, and the exact location of the UAM module. Then, it forwards the query to the UAM module, and waits for the user's attributes in a source format. When the CCS gets these attributes it has to use the Conversion Policy rules, which define how to translate the external credentials into SAML attributes.

In a push model, where the user requests his authorization attributes before accessing the target network, the CCS may receive the conversion query directly from the UAM, or from its local AAA server, according to the push model alternative selected. That is, when the user needs the authorization credentials to get access to a target domain, first he asks the home domain for his attributes. Then, attributes are forwarded to the CCS and hereafter the process is very similar to the one described above.

These modules allow the interaction between domains based on different authorization systems, and its corresponding behavior is completely based on two integration policies. The next section describes those policies and gives the guidelines about how they can be expressed.

3.2 Integration Policies

In order to define how these new components work, it is necessary to introduce the policies involved. The UAM module needs a policy specifying which attributes can be revealed to which target domains, for example, depending on the level of trust agreed with that domain. This policy is named the *Disclosure Policy*. On the other hand, the CCS needs a policy describing how attributes from the user's home domain must be mapped into internal attributes, to be used next by the PDP to obtain an authorization decision. This policy is defined as the *Conversion Policy*.

Disclosure Policy. When two or more domains are involved in a trust relationship, where users from one domain can request access to resources in the others domains, it is necessary to define which user's attributes could be revealed to those domains. For example, if the domain requesting those attributes is a

highly trusted domain, due to a previously established very restrictive security agreement, the home domain could reveal all the user's attributes. Otherwise, if the relationship established between the domains is not so trusted, the home domain could decide to conceal some of them.

The Disclosure Policy identifies the allowed external CCSs (domains), assigns specific roles to every domain based on the existing relationships between the two domains, and defines the set of attributes that can be revealed and under which conditions.

In the proposed scenario, we suppose the home domain is based on the PERMIS authorization infrastructure, that is, the use of Attribute Certificates to represent the subject-attribute pair. This Disclosure Policy uses the resource access control policy defined by PERMIS, which requires that every external domain must have, at least, an internal AC defining the role played by that domain.

The Disclosure Policy defines the following elements:

- *Subjects*:Set of external domains allowed to request internal user's attributes.
- *Roles*: The set of roles that can be played by external domains. Those roles might be organized hierarchically.
- *SOA*: Authorization Authority managing the ACs issued by the home domain for external CCSs.
- *Targets*: The set of users whose attributes are to be disclosed (the user requesting the access to the foreign network must be included here).
- *Actions*: Only *disclose* has been defined, and its parameter is the attribute that is to be disclosed.
- *TargetAccess*: Defines which attributes assigned to a particular set of users can be disclosed to which domains, and under which conditions.

Figure 2 represents a simple Disclosure Policy defined by *o=PERMISDomain, c=C*. The *SubjectPolicy* element specifies that only the external domain *cn=CCS, o=SAMLDomain, c=C* will be allowed to request attributes. As we explained before, the NAS-SAML domain must have an attribute certificate issued by the *PERMISDomain* SOA. This AC will contain the attribute type *permisRole*, with value *LongTerm-CCS*, specifying the role played by this domain, in this case, a stable and durable relationship is specified.

The *TargetPolicy* defines the set of PERMIS users who make use of external resources, that is, only attributes assigned to those users can be revealed by the UAM. There is only one allowed action, *disclose*, which uses an attribute type and an attribute value as parameters. Finally, the *TargetAccessPolicy* defines that only CCSs (*Subjects*) assigned to the *LongTerm-CC* attribute value will have access to the attribute *studentRole* type, with value *ERASMUS*, assigned to *Students*.

When the UAM module receives an attribute query it must generate a request message for each user attribute, specifying the target domain (subject and role), the attribute requested (type and value) and the required action. This request is sent to the PDP and it returns a response message indicating whether the attribute can be revealed or not.

```
<X.509_PMI_RBAC_Policy OID="2.6.2004.24.1.2005">
  <SubjectPolicy>
    <SubjectDomainSpec ID="SD_International_CCSs">
        <Include LDAPDN="  cn=CCS, o=SAMLDomain,c=C  "/>
    </SubjectDomainSpec>
  </SubjectPolicy>
  <RoleHierarchyPolicy>
    <RoleSpec Type="permisRole"
        OID="1.2.826.0.1.3344810.">
      <SupRole Value="ShortTerm-CCS">
      </SupRole>
        <SupRole Value="  LongTerm-CCS  ">
          <SubRole Value="ShortTerm-CCS"/>
      </SupRole>
      </RoleSpec>
  </RoleHierarchyPolicy>
  <SOAPolicy>
    <SOASpec ID="PERMISDomain_UAM"
      LDAPDN="  cn=UAM,o=PERMISDomain,c=C  "/>
  </SOAPolicy>
  <RoleAssignmentPolicy>
    <RoleAssignment>
      <SubjectDomain ID="SD_International_CCSs"/>
      <RoleList>
        <Role Type="  permisRole  "
          Value="  LongTerm-CCS  "/>
      </RoleList>
      <Delegate Depth="0"/>
      <SOA ID="PERMISDomain_UAM"/>
      <Validity/>
    </RoleAssignment>
  </RoleAssignmentPolicy>
  <TargetPolicy>
    <TargetDomainSpec ID="All-Users">
      <Include LDAPDN="o=PERMISDomain, c=C"/>
    </TargetDomainSpec>
    <TargetDomainSpec ID="Students">
      <Include LDAPDN="ou=Students,
        o=PERMISDomain, c=C"/>
    </TargetDomainSpec>
    <TargetDomainSpec ID="Professors">
      <Include LDAPDN="ou=Professors,
        o=PERMISDomain, c=C"/>
    </TargetDomainSpec>
  </TargetPolicy>
  <ActionPolicy>
    <Action Name="disclose" Args="role value"/>
  </ActionPolicy>
  <TargetAccessPolicy>
    <TargetAccess>
      <RoleList>
        <Role Type="permisRole"
          Value="LongTerm-CCS"/>
      </RoleList>
      <TargetList>
        <Target Actions="disclose">
          <TargetDomain ID="Students"/>
        </Target>
      </TargetList>
      <IF>
        <AND>
          <Substrings>
            <Arg Name="role" Type="String"/>
            <Constant Type="String"
              Value="studentRole"/>
          </Substrings>
          <Substrings>
            <Arg Name="value" Type="String"/>
            <Constant Type="String"
              Value="ERASMUS"/>
          </Substrings>
        </AND>
      </IF>
    </TargetAccess>
  </TargetAccessPolicy>
</X.509_PMI_RBAC_Policy>
```

Fig. 2. Disclosure Policy example

Conversion Policy. A target domain, which has to interact with a home domain based on a different authorization system, needs both to use the CCS module and to define the conversion policy for each domain. Following the design described in [12], this policy is based on XACML, and it defines the following elements:

- *Subject*: One or more subjects specifying the related home domains.
- *Resource*: The resource elements represent the credentials issued by the home domain that need to be translated into internal credentials.
- *Action*: This policy contains only the *translate* action.
- *Obligation*: Every permitted translation will imply an obligation, which specifies how to translate the credentials.

Figure 3 shows a simple Conversion Policy composed by the set of policies related to every home domain. For example, *PERMISDomainConversionPolicy* defines the whole set of attributes that can be translated from the domain *o=PERMISDomain,c=C*. There is only one allowed action, *translate*. The home domain is specified using the *Subject* element, and for each attribute of that domain it is necessary to define a conversion policy. This specific policy, for example the *PERMISDomainSimplePolicy1*, defines the *Rule* specifying the attribute to be translated (type and value), and a *Obligation* element specifying the internal target attribute. For example, the previous example defines that the *PERMISDomain* attribute type *studentRole* with value *ERASMUS* must be translated into

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=" access_control-xacml-2.0-policy-schema-
cd-04.xsd"
PolicySetId="GlobalConversionPolicy" PolicyCombiningAlgId="...">
   <Target>
      <Actions>
            <Action>
               <ActionMatch MatchId="...:string-equal">
                  <AttributeValue DataType="...#string">translate
                        </AttributeValue>
                  <ActionAttributeDesignator          AttributeId="..:action"
                        DataType="...#string"/>
               </ActionMatch>
            </Action>
      </Actions>
   </Target>
   <PolicySet PolicySetId="PERMISDomainConversionPolicy"
      PolicyCombiningAlgId="...">
       <Target>
            <Subjects>
               <Subject>
                        <SubjectMatch MatchId="...:string-equal">
                           <AttributeValue DataType="...#string">
                                 cn=UAM,o=PERMISDomain,c=C
                           </AttributeValue>
                        <SubjectAttributeDesignator
                           AttributeId="...:external:SOA"
                           DataType="...#string"/>
                        </SubjectMatch>
               </Subject>
            </Subjects>
      </Target>
```

```
<Policy PolicyId="urn:ccs:PERMISDomainSimplePolicy1"
                  RuleCombiningAlgId="...">
         <Target/>
         <Rule RuleId="PERMISDomainSimpleRule1" Effect="Permit">
            <Target>
               <Resources>
                     <Resource>
                        <ResourceMatch MatchId="...:string-equal">
                           <AttributeValue DataType="...#string">
                                 studentRole
                           </AttributeValue>
                           <ResourceAttributeDesignator
                                 AttributeId="...:resource-id"
                                 DataType="...#string"/>
                        </ResourceMatch>
                        <ResourceMatch MatchId="...:string-equal">
                           <AttributeValue DataType="...#string">
                                 ERASMUS
                           </AttributeValue>
                           <ResourceAttributeDesignator
                                 AttributeId="....:value"
                                 DataType="...#string"/>
                        </ResourceMatch>
                     </Resource>
               </Resources>
            </Target>
         </Rule>
         <Obligations>
            <Obligation ObligationId="urn:ccs:Obligation1"
               FulfillOn="Permit">
               <AttributeAssignment DataType="...#string"
                     AttributeId="urn:saml:attribute:role:student">
                     ERASMUS
               </AttributeAssignment>
            </Obligation>
         </Obligations>
      </Policy>
   </PolicySet>
</PolicySet>
```

Fig. 3. Conversion Policy example

the internal attribute type *urn:saml:attr:role:student*, with value *ERASMUS*. It is worth noting that the addition of home domains involves additional *PolicySet* elements, and more attributes per domain requires more *Policy* elements.

When the CCS module receives a conversion query it must generate a policy request message, specifying the home domain subject, the source user's attributes and the required action. This request is sent to the policy manager and it returns a policy response message including whether that action over that resource has been allowed, and the target user's attributes as obligations elements. Then the CCS module returns to the petitioner a conversion response in the internal domain format.

4 Design Alternatives

Interactions among the different components described in the paper depend on the requirements imposed by the user to gain access to the network. On the one hand, the end user can follow a pull approach, which requires the minimum overload and is more suitable for limited terminals, such as PDAs or mobile phones. In this way, all the authorization tasks are performed by the system. On the other hand, following a push model, the user can present a particular set of attributes. The push model involves support for selecting and transporting attributes from the end user terminal, representing a more intrusive approach. In consequence, we provide solutions to these two different environments, including three different alternatives. Beside the pull model approach, we present to

alternatives for the push model. In the first one the user obtains his credentials using SAML, once the user AC's are translated. In the second one, he directly obtains his Attribute Certificates, which will be translated during the network access attempt. In both ways, before request access to the AAA server, the user selects which of them wants to present.

Independently of the selected approach, when the end user requests access in a target domain, he should be authenticated, before starting the authorization process, as described in [12], but it is out of the scope of this paper.

4.1 Design Alternative 1: Pull Model

This alternative provides to the user an authenticated and authorized connection in a transparent way. The management of the authorization data, that is, the conversion and validation process, are performed using a pull approach.

In this way, the first step is the authentication of the user, following the process required by the network access technology. We suppose this authentication is based on public key certificates, therefore, the user must present one of these during the authentication process and it must be validated by the target AAA server. In this scenario, the authentication is delegated, and might be based on the existence of a previously generated cross-certification relationship between both domains.

Once the user is authenticated, the authorization process starts. The AAA server has to discover that the user belongs to a home domain which is not based on SAML/XACML, and therefore his attributes must be converted. The user's home domain can be discovered using the DN attribute held in his certificate.

The AAA server sends the attribute query requesting the user's attributes to the CCS module. This request is formed by a *SAMLRequest* object containing an *AttributeQuery* and indicating that the response must be encoded using the *AttributeStatement* sentence. It also includes the name of the user (subject) requesting the access and, optionally, the type of attributes expected.

Once the CCS obtains the request, it has to discover how to contact the user's home domain. This information is stored in the *Resource Access Policy*, as described in [12], as additional information about the domains able to gain network access. The contact point specified in that policy is the UAM module, located in the user's home domain.

The CCS signs and forwards the attribute query, changing the expected response sentence to *WrappedStatement*. In this way, the CCS module indicates to the UAM that the user's attributes must be returned in the original format, but encapsulated in this statement, to be translated at the target domain.

When the UAM receives the attribute query, it has to return only the attributes allowed by the Disclosure Policy. To check this policy, the UAM obtains from the internal LDAP repository all the ACs issued to this user, and the ACs issued to the target domain. The first set of certificates contain all the user's attributes in his home domain, whilst the second one contains the roles played by the target domain. The CCS public key certificate, included in the XML Sig-

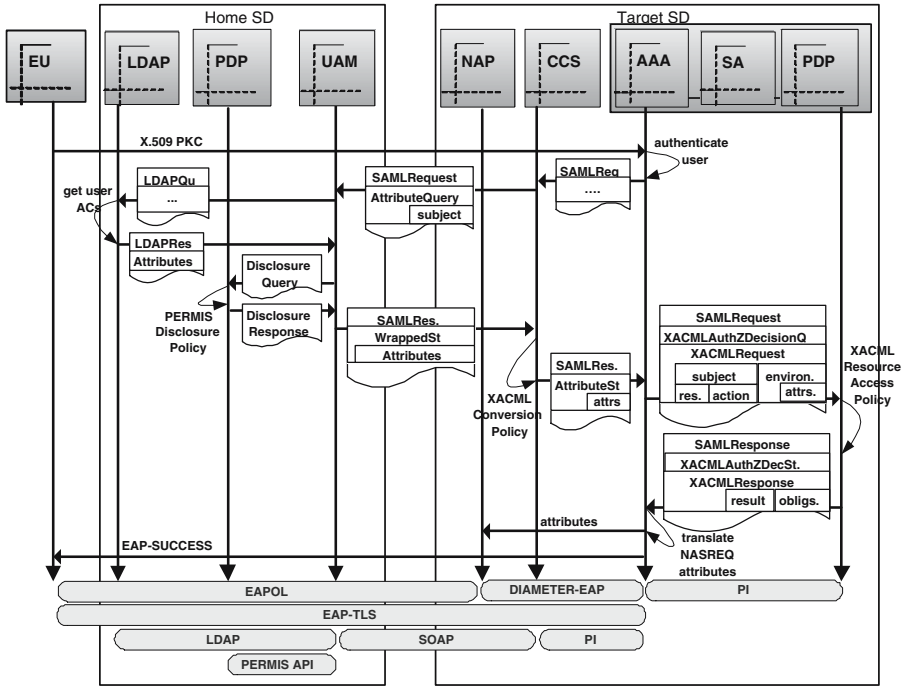

Fig. 4. Pull model

nature object of the request message, can be used to authenticate the CCS and obtain the target domain.

At this point, the UAM asks the PDP component, the decision point, for the attributes that can be revealed. Using the above ACs, the UAM issues a request per user's attribute, specifying: the target domain (*subject*), the attributes assigned to that domain, the user attribute and its value, and the requested action (*disclose*). These requests are sent using the PERMIS API.

Once the UAM knows the set of attributes that can be returned to the target domain, it generates a *SAMLResponse* including a *WrappedStatement* sentence, following the schema defined in [7]. This statement includes:

- *WrappedData*: The allowed user's ACs.
- *StatementType*: URI describing the type of the wrapped data, for example, *x509ac*.
- *Encoding*: URI describing the encoded format, for example, *Base64*.

Once the CCS module receives the response message, it must convert the received attributes into internal understandable SAML attributes. To obtain these, the CCS uses the Conversion Policy, described in section 3.2. The CCS obtains the SOA identifier and each attribute type and value pairs needed to be converted. All this information is included in the ACs. For each attribute type

and value pair, the CCS checks the policy and obtains the associated attribute designator and value in internal format (SAML).

Once all the attributes are translated, the CCS sends a *SAMLResponse* message including all of them as *AttributeStatement* sentences. When the AAA server receives the user's attribute it checks the Resource Access Policy as described in [12], to grant or deny the required service.

The pull model provides strong authentication of users, and a transparent authorization service based on ACs and SAML statements, avoiding the client software being modified to support these high level authorization schemes. This alternative has the disadvantage of providing no control to the user about the type of service required, that is, the user can not select the set of attributes to be presented during the network access request. In our opinion, this should not be seen as a disadvantage in most of the existing environments where default access is being provided or where the users do not want to get involved in authorization issues.

4.2 Design Alternative 2: Push Model Based on SAML Attributes

Using the push model, end users are able to present their authorization credentials during the network access request. In this alternative, those authorization credentials are expressed using SAML attribute statements containing the roles played by the user. This model is based on two steps: first, the user has to request his attributes from his home domain, specifying the desired target domain and service. This step involves the conversion process ending with the user holding the converted attributes. Then, the user presents those converted attributes to the target domain, requesting, for example, network access. This steps involves the authentication and the authorization decision processes. This section describes the first step since the last stage is fully described in [12].

In this model, the user requests his attributes from his home domain, specifying the desired target domain and service. The user directly accesses the UAM module, for example, through a web browser. The UAM, once the user is authenticated, follows the same procedure described in the pull model, only in this case the user acts as a proxy for the target domain. First, the UAM retrieves the user's ACs from the LDAP repository, and then asks the PDP module about the attributes that can be revealed to the target domain.

In this way, and following the push model described in [7], the UAM sends to the CCS a *ConversionQuery* sentence including the *WrappedStatement* described above. The *ConversionQuery* sentence contains the following elements:

- *Assertion*: Assertion including the *WrappedStatement* object.
- *Recipient*: Attribute defining the entity that will receive the assertions, for example, the CCS.
- *RespondWith*: This element, including in the *SAMLRequest* message, is used to specify type of SAML assertion that is expected to be generated, for example, the *AttributeStatement*.

The UAM waits for a *SAMLResponse* message including the converted attributes. The location of the CCS could be previously configured in the home

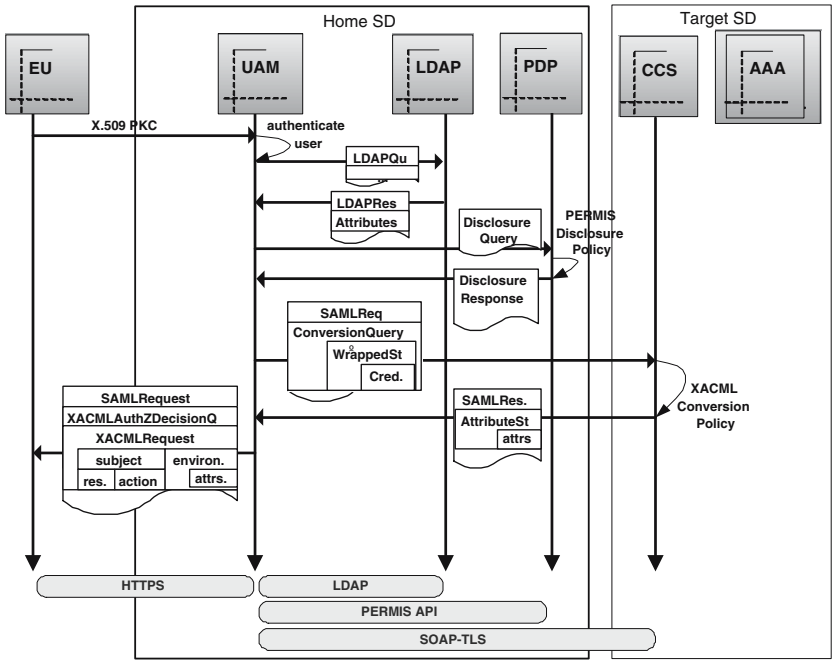

Fig. 5. Push model based on SAML Attributes

domain, as part of the Disclosure Policy, or could be obtained from the CCS public key certificate, as an *authorizationInformationAccess* X.509 extension. The CCS module, after enforcing the Conversion Policy, returns the converted attributes as an *AttributeStatement* to the UAM, which generates the *XACMLAuthZDecisionQuery* following the push model described in [12]. The user then forwards this to the AAA server.

It it worth noting that absence of revocation mechanisms for SAML statements, and its recommended usage for short-term sessions, suggests that the SAML documents should not be cached in intermediate entities, like a certificate repository.

The main advantage of this alternative is that it provides to the end user complete visibility and control of the authorization process, since he can select the type of connection, security properties, quality of service, etc. Moreover, he can provide personal information by means of references to some of his attributes. On the other hand, the software used by the client (usually referenced as supplicant) must be modified in order to deal with SAML statements during the second step, as we can find in other existing proposals [12].

4.3 Design Alternative 3: Push Model Based on Attribute Certificates

In this alternative, authorization credentials, presented by the end user to the AAA server, will be the user's Attribute Certificates, obtained from the UAM

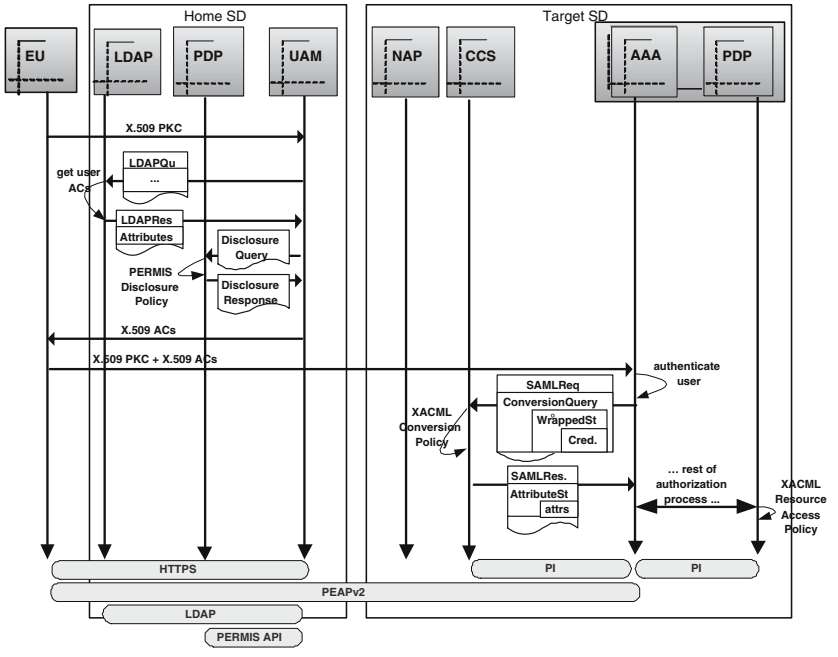

Fig. 6. Push model based on Attribute Certificates

in the user's home domain. This model is based on two steps: first, the user has to request his ACs from his home domain, specifying the desired target domain and service. Then, the user, after filtering those according to his privacy policy, presents them to the target domain, requesting, for example, network access. This last step involves the authentication, the credentials conversion, and the authorization decision processes.

In this model, the user requests his attributes from his home domain, specifying the desired target domain and service. The user directly accesses the UAM module, for example, through a web browser. The UAM, once the user is authenticated, retrieves the user's ACs from the LDAP repository, and then asks the PDP module about the attributes that can be revealed to the target domain. The UAM returns the selected ACs to the end user.

Once obtained the user's ACs, he requests network access connection in the target domain, pushing those ACs according to his internal policy. The user initiates the 802.1X authentication process with the foreign AAA server. In this case, we are going to use the PEAP (Protected EAP) protocol [2], which defines how to establish a TLS channel that can be used to authenticate the communicating parties and to protect further messages related to the authorization process. For example, the user's ACs would be base64 encoded and sent as normal attributes.

The AAA server that receives the network connection detects the use of non-SAML credentials, and sends a credential conversion request to the local CCS.

That is, a *ConversionQuery* sentence including the *WrappedStatement* described above.

The AAA server waits for a *SAMLResponse* message including the converted attributes. The CCS module, after enforcing the Conversion Policy, returns the converted attributes as an *AttributeStatement* to the AAA server, which generates the *XACMLAuthZDecisionQuery* following the push model described in [12].

This alternative presents the same advantages and disadvantages that the alternative 2 previously described. The main difference is the transport of ACs from the home domain to the target domain, where they need to be translated by the CCS module.

5 Related Work

Nowadays, due to the existence of several authorization schemes, their integration is becoming a common requirement in multi-domain scenarios. This section describes some of the main current integration solutions which have informed our work, including environments such as SAML, PERMIS or X-RBAC [11].

PERMIS is being used by other authorization schemes to improve the authorization decision process and to allow users holding an Attribute Certificate to interact with other environments, such as Shibboleth [4] or Grid Services [1].

Shibboleth defines an access control approach scenario for web environments, which is composed of three main entities: service providers offering target resources; identity providers maintaining the user's identities; and the end users. It offers user authentication and attribute-based authorization services based on SAML, as well as a SSO service. It is based on a single Attribute Authority, and has no generalized decision engine. One solution to improve Shibboleth is the integration with PERMIS. During the SIPS project [16], PERMIS was integrated to work with Shibboleth. There are two modes of operation: in one mode PERMIS uses X.509 Attribute Certificates to make authorization decisions, and in the other mode PERMIS uses plain Shibboleth attributes. The PERMIS team had to develop a module similar to the conversion service discussed in this paper, so that PERMIS would accept the plain-text attributes. The module that was developed as the result is not a standalone service, rather it is an extension to PERMIS that is invoked via the API. This only once again proves the necessity for credential conversion for efficient interoperation of Privilege Management Infrastructures.

PERMIS has also been extended to support the SAML standard for Grid Services, as defined in [5]. This describes how PERMIS has been adapted in order to integrate its authorization engine into the Globus Toolkit. This integration is based only on the exchange of authorization decision queries and responses between an authorization service acting as the Policy Decision Point (PDP), based on X.509 ACs, and the Grid infrastructure. In this scenario, the decision about the resource is taken by the PERMIS ADF module, following the PERMIS policy syntax. In this way, it takes the advantage of the SAML standard for

integration purposes. It is also able to use the SAML extensions proposed by the OGSA-Authz working group [17] for efficiency purposes.

Besides PERMIS, other scenarios describing the integration between different authorization schemes are being defined. An interesting solution for the authorization process in multi-domain environments, based on the RBAC model, is described in [11]. Although this model does not propose an integration scenario, because both domains are based on the same authorization scheme (X-RBAC), it does propose a policy mapping users' attributes from one domain to another. This proposal does not need a conversion service but it clearly shows that the relationship between the user's attributes or roles from different domains must be mapped in some way. In order to support mapping between different authorization schemes, the proposed policy should be extended allowing the definition of the source and destination authorization formats.

6 Conclusions

This paper proposes a solution to integrate two different authorization schemes, PERMIS which is being widely used due to its powerful authorization engine, and SAML which is becoming a *de facto* standard for authorization environments. We have presented a particular application scenario, the network access service, to demonstrate how the integration addresses, for example, authorization between domains based on a RBAC scheme.

Beside the architectural elements needed by the integration process, this paper presents the guidelines of the policies controlling the integration scenario, the Disclosure and Conversion policies. These policies are defined using the different authorization languages proposed by the two domains, the PERMIS XML authorization policy and XACML. In this way, there is no need to include additional authorization technologies to accomplish the integration process.

In order to offer a versatile solution, this paper presents three different RBAC designs, which can be individually selected in order to implement the access control service that is best suited for a particular environment. Authorization can be performed in a transparent way, from the user's point of view, using the pull model. The two push model approach slightly overloads the system in relation to the previous model, but it provides more options to the users.

As a statement of direction, although the proposed scenario is based on a PERMIS home domain and a NAS-SAML target domain, it could be easily adapted to work in reverse order, that is, based on a SAML home domain and a PERMIS target domain.

Acknowledgements

Partially supported by IST-2001-32161 (Euro6ix) and IST-2002-001929 (SEINIT) projects.

References

1. *Globus Toolkit*, February 2005. http://www.globus.org.
2. H. Anderson, S. Josefson, G. Zorn, D. Simon, and A. Palekar. *Protected EAP Protocol (PEAP)*, 2004. IETF Draft.
3. P. Ashley, S. Hada, G. Karjoth, and M. Schunter. E-P3P privacy policies and privacy authorization. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 103–109. ACM Press, 2002.
4. S. Cantor. *Shibboleth Architecture. Protocols and Profiles*, February 2005. http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-arch-protocols-06.pdf.
5. D. W. Chadwick, S. Otenko, and V. Welch. Using SAML to link the GLOBUS toolkit to the PERMIS authorisation infrastructure. In *Proceedings of Eighth Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, 2004.
6. David W. Chadwick, Alexander Otenko, and Ed Ball. Role-based access control with x.509 attribute certificates. *IEEE Internet Computing*, 7(2):62–69, 2003.
7. O. Cnovas, G. Lpez, and A. F. Gmez. A Credential Conversion Service for SAML-based scenarios. In *Proceedings of 1st European PKI Workshop*, pages 297–305, June 2004.
8. C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. *Generic AAA Architecture*. Internet Engineering Task Force, August 2000. Request for Comments (RFC) 2903.
9. S. Godik and T. Moses. *OASIS eXtensible Access Control Markup Language (XACML) Version 2.0*, February 2005. OASIS Standard.
10. The Open Group. *Authorization (AZN) API*, January 2000.
11. E. Bertino J. B. D. Joshi, R. Bhatti and Arif Ghafoor. Access-Control Language for Multidomain Environments. *IEEE Internet Computing*, 8(6):40–50, November 2004.
12. G. Lpez, O. Cnovas, A. F. Gmez, and R. Marn. A Network Access Control Approach based on the AAA Architecture and Authorization Attributes. In *Proceedings of International Workshop on Systems and Security Networks (SSN05)*, April 2005. To be published.
13. E. Maler, P. Mishra, and R. Philpott. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, September 2003. OASIS Standard.
14. LAN MAN Standards Committee of the IEEE Computer Society. *IEEE Draft P802.1X/D11: Standard for Port based Network Access Control*. IEEE, March 2001.
15. K. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Proceedings of Network and Distributed System Security Symposium*, April 2001.
16. SIPS. *Seamlessly Integrating PERMIS and Shibboleth*, February 2005. http://www.jisc.ac.uk.
17. V. Welch, R. Ananthakrishnan, F. Siebenlist, D. Chadwick, S. Meder, and L. Pearlman. *Use of SAML for OGSA Authorization*, February 2005. GGF Draft (OGSA-Authz WG).

Interoperation Between a Conventional PKI and an ID-Based Infrastructure

Geraint Price and Chris J. Mitchell

Information Security Group, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
{Geraint.Price, C.Mitchell}@rhul.ac.uk

Abstract. In this paper we consider how practical interoperation between a conventional PKI and an infrastructure based on ID-based cryptography might be achieved. Major issues arising from such interoperation are raised, and possible solutions are proposed.

1 Introduction

Suppose a domain (domain A) uses a 'conventional' public key infrastructure (PKI), with one or more Certification Authorities (CAs). Suppose also that a second domain (domain B) operates an ID-based public key infrastructure, with one or more trusted key generation entities. Now suppose that the members of the two domains wish to inter-operate, i.e. verify signatures generated by each other and/or send each other encrypted messages. This clearly presents a variety of problems, and it is the purpose of this paper to look at these issues.

Of course, one pre-requisite for such a scheme to work is that the members of domain A must support the ID-based crypto-algorithms used by members of domain B, and vice versa. However, this should be relatively straightforward to achieve either by pre-configuring all entities to support a sufficiently wide variety of algorithms, or by using mechanisms such as signed applets, etc.

The main focus of this paper is to address the underlying key management issues, together with the policies and practices that need to be put in place to support interoperation. That is, we investigate the management issues underlying the interoperation of a certificate-based PKI with an Identity-based one.

In particular we consider issues such as:

- What are the specific issues arising for interoperation in either direction?
- What are the security implications of interoperation?
- What lessons can we learn, and where do we go from here?

Essentially, understanding how to support interoperation comes down to understanding the differences in operation between the two types of infrastructure, and how this affects the security architecture.

The remainder of this paper is structured as follows. Section 2 gives an overview of public key and ID-based cryptography, highlighting the differences between certificate based infrastructures and identity-based infrastructures. This

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 73–85, 2005.

includes a brief review of the (limited) prior art in the field. This leads naturally into Section 3, where we provide a description of a certificate-based solution to the interoperation problem. This is broken down into three subsections: an overview of inter-domain operation in 'native' environments; how to get a ID-based solution to work for the certificate-based user; and how to get a certificate-based solution to work for the ID-based user. Section 4 moves on to look beyond the use of certificates to solve the interoperation problem, highlighting both some potential alternative solutions and other issues that might come into play (e.g. the use of the Al-Riyami-Paterson certificateless public key infrastructure schemes [1,2]). In Section 5 we provide an overview of the lessons learnt to date, and we outline possible future work. Finally, Section 6 gives some conclusions.

2 Technical Background

In this section we briefly review the technical background which we require for our subsequent analysis. We begin by reviewing the history of Identity-based (or Identifier-based) public key cryptography, outlining the differences between it and certificate-based infrastructures. We then close this section by reviewing what prior art exists discussing possible interfaces between certificate-based and identifier-based cryptographic infrastructures.

In practice, the management, distribution and use of public key certificates as part of a conventional PKI adds significant overheads. It was with this drawback in mind that, in 1984, Shamir proposed a new form of public key cryptography [13] which he termed Identity-based cryptography. In the 'traditional' model of public key cryptography, both the public and the corresponding private key are random in appearance; by contrast, in the ID-based model, the public key pair is generated from publicly verifiable data (with the generation of the corresponding private key requiring an additional secret parameter held by a Trusted Authority). Shamir pointed out that, if the public key is generated from an identifier of the entity to whom the key is assigned (e.g. a user's email address), this would obviate the need for certificates, and would simplify the management of public keys in a fielded system.

Until relatively recently the main limitation of ID-based cryptography was that, while Shamir and many others were able to propose a variety of ID-based signature schemes (see, for example, [11]), no efficient ID-based encryption scheme was known. This situation changed in 2001, when Boneh and Franklin [3,4] proposed a practical and efficient encryption scheme.

The main benefit from the use of ID-based encryption, as discussed in [13], is the ability to encrypt a message sent to a given recipient without first having to retrieve the recipient's public key. All that is required is knowledge of the recipient's identity. By contrast, the potential practical benefits of using an identity-based signature scheme are rather less. This is because, for signatures, the private key needs to be created before a signature can be created in both the conventional and the ID-based cases. This meant that, until Boneh and Franklin's discovery, very little work had been conducted on the practical use

of ID-based cryptography; however, since then, there has been a resurgence of interest in the field.

We next briefly discuss some of the main operational differences. Firstly, observe that, for ID-based encryption, the private key only needs to be generated when it is actually needed to decrypt data. Secondly, while there is no need for certificate revocation *per se* within the system, controlling the lifetime of valid identifiers results in very similar implementation problems to those encountered in a conventional PKI, as discussed by Paterson and Price [12].

Whilst this paper discusses the possible interaction between a traditional certificate-based public key infrastructure and an identifier-based infrastructure, there is almost nothing in the published literature that looks at this issue. We now briefly review relevant previous work.

Chen et al. [5] discuss the use of a hybrid scheme, with a traditional certificate-based PKI being used by the Trusted Authority (TA) at the domain/organisational level, but with identity-based cryptography used at the user level. They base their design on the premise that certificate-based systems are more efficient at the domain level and are less efficient at the user level, but that the inverse is true for identity-based systems. They note that they require cross-certification between the domain trust authorities, but do not provide detailed discussions of this issue.

Smetters and Durfee [14] propose an extension to DNSSEC, which is itself a recently proposed secure extension of DNS. They discuss the problems that would be faced if a global identity-based system were to be proposed. The main drawback they highlight would be the need for a central TA, and the management of a hierarchy of TAs under a root TA. In the case of identity-based systems, the keys of the mid-level TAs could be generated by TAs further up the tree, which might be an undesirable property. They propose that each individual trust domain runs its own TA, with their system parameters being distributed via DNSSEC.

While this prior work does consider the problems of trying to implement an identity-based scheme on a large scale, both proposals offer a tiered approach at a global scale, where it is assumed that peer-level communication is carried out using the same cryptographic mechanisms. In the remainder of this paper we analyse the technical challenges that would arise if users in two different organisations using different underlying mechanisms were to wish to communicate securely.

3 A Simple Certificate-Based Solution

In this section we consider a simple certificate-based solution to the technical challenge faced when users in domains with different types of asymmetric infrastructures wish to communicate with each other. The basic outline of the design is to have the two separate domain infrastructures (one certificate-based and one identity-based) use traditional cross-certificates to authenticate the CA to the identity-based domain, and the TA to the certificate-based domain.

We divide the discussion into three main parts. In Section 3.1 we consider what would happen if domains of the same type (i.e. certificate-to-certificate or identity-to-identity) need to interoperate. Then in section 3.2 we review our cross-certification design from a certificate-based user's perspective, and in section 3.3 we review our design from an identity-based user's perspective.

3.1 Core Principles

Before detailing our proposed solution to certificate-to-identity cross-domain communication, we review how cross-domain communication happens in native environments.

Certificate-Based: The traditional method of supporting inter-domain communication between multiple certificate-based environments is to use *cross-certificates*. Cross-certificates are certificates issued by a CA to certify a peer-level CA's root public keys. This allows the client software of users in one domain to validate the certificates of users within another domain by first verifying the cross-certificate signed by their own CA (or, more generally, by verifying a chain of cross-certificates). While this solution is relatively straightforward, it has problems. Most notably, these problems arise when the two CAs reside in logically separate security domains. As a result, certificate path discovery can sometimes be complicated. Even if the certificate path can be built, the analysis of Certificate Policies (CP) and Certification Practice Statements (CPS) [6] can make it difficult for the validation mechanism of a user in one domain to know whether the certificate they have is suitable for the task which the user wishes to use it for.

ID-Based: Although very little work has been done on inter-operation between identity-based domains, both Chen et al. [5] and Smetters and Durfee [14] propose the use of security mechanisms 'external' to identity-based cryptography to authenticate cross-domain credentials. Chen et al. use traditional certificate-based mechanisms, and Smetters and Durfee use DNSSEC. It seems apparent that identity-based cryptography itself is not suitable for supporting the authentication of inter-domain credentials. This view would appear to be validated by the work of Chen et al. and Smetters and Durfee on cross-domain communication.

Similar problems to those encountered in certificate-based interaction arise here, notably that the policies operated within the two separate domains might need to be analysed for compatibility. However, at least in the case of identity-based encryption (as opposed to signature), path discovery might be simpler as the recipient could be forced to retrieve a new private key from their local TA. This would mean that the sender essentially offloads path discovery to the TA in the recipient's domain.

As we see below, the problems highlighted here are likely to get worse as the two different types of domains interact. This is mainly due to the fact that certificate-based users need to be able to make use of, and make sense of, ID-based parameters, and vice versa. By necessity, this adds a layer of complexity to the system architecture.

3.2 The Certificate-Based User

For the reminder of this paper we consider two security domains. We suppose domain **A** runs a traditional certificate-based PKI, while domain **B** runs an identity-based infrastructure.

From our discussions in the previous section we note that using a cross-certificate would appear to be the logical choice for supporting peer-entity authentication between the CA and TA. In this section we look at the design of such a scheme from a certificate-user's point of view.

We start by providing a simplified overview of how the design would work.

- The CA for domain **A** generates a cross-certificate for the TA in domain **B**, containing the TA's public parameters.
- This cross-certificate could contain policy information for domain **B** of relevance to a user in domain **A**.
- A user in domain **A** validates the cross-certificate and checks the policy for acceptability (we discuss how this policy checking might be carried out below).
- As the certificate is likely to be obtained from a local repository, there will typically be a need for a certificate status check, and possibly some additional path validation, to be carried out on the certificate.

The idea of using a certificate to authenticate the public parameters of a TA is not new. Apocryphal evidence suggests that some organisations looking to implement identity-based encryption in practice intend to use a traditional PKI certificate to allow users within the system to verify the domain parameters for an ID-based system.

The above technique would allow members of domain **A** to derive public keys from entities within domain **B**. Hence, this would enable them to verify signed message from, and send encrypted messages to, members of domain **B**.

We now discuss some of the operational details of such a scheme. In doing so, we note some potential difficulties that arise from the fundamental differences between certificate and identity based infrastructures.

Certificate Type and Content. The first area we explore is the issue of which type of certificate to use for the cross-certificate. That is, should it be a traditional X.509 identity certificate or an X.509 attribute certificate. As well as the choice of certificate type, we also consider the contents of such a cross-certificate.

There are potential advantages and disadvantages associated with the use of both types of certificate. The arguments can be summarised as follows.

Identity Certificate: This would appear to be the more logical choice, with the identity of domain **B** being bound to the scheme parameters used within **B**'s identity-based implementation. This could be achieved by defining new extension fields for the X.509 certificate format to manage this additional information. This approach would provide the user in domain **A** with a mechanism

that most accurately reflects the way in which cross-certification is carried out in a certificate-based environment. However, it does come at a cost. Because of the very nature of identity-based schemes, there is no explicit means of revocation. One way of dealing with this problem is to periodically update the scheme parameters within domain **B** in order to refresh all keys in the system. This would increase the pressure on the certificate management procedures of the cross-certificate for domain **A**'s CA. This cross-certification could be made more lightweight by not including the scheme parameters in the certificate. Thus, we would be relying on the certification of the domain **B**'s identity alone.

Attribute Certificate: An alternative would be to use an attribute certificate to certify domain **B**'s TA's right to produce valid system parameters. This effectively extends domain **A**'s security associations to domain **B**, purely for the act of managing the identity-based parameters. This solution is more lightweight from domain **A**'s viewpoint, as it is up to the TA in domain **B** to update the system parameters and authenticate them within domain **A**. However, we believe that an attribute certificate will provide a weaker binding between the cross-certificate and the system parameters, as the CA in domain **A** is only certifying the identity of domain **B**, and not the system parameters directly. As well as the weaker binding, such a scheme would probably require the TA in domain **B** to maintain an additional set of certificate-based keys in order to authenticate the system parameters to the users in domain **A**. There is also the question as to whether the TA in domain **B** really has the 'right' to carry out an action in domain **A**. We believe that further discussions on this point could help resolve how the powers of domain **B**'s TA could be expressed within domain **A**'s security policy. However, this mechanism could be strengthened by having the TA's system parameters included within the certificate as an additional set of attributes.

As we can see from our discussion, both constructs have their benefits and drawbacks. Additionally, we have noted that both types of certificate can be modified to present a tighter or more lightweight binding within each certificate type.

We believe that the use of an X.509 identity certificate would be preferable in practice, with the system parameters for domain **B** included either as the public key content, or in a special extension field. This would also more naturally map the way in which a user in domain **A** might manage existing cross-certificates to certificate-based domains. Also, an identity certificate is traditionally associated with the provision of authentication, which more accurately captures the nature of cross-certification in this environment.

CPs and CPSs. The biggest difference between certificate-based infrastructures and identity-based infrastructures is in the way they handle policies. In a certificate-based PKI, great reliance is placed on the use of Certificate Policies and Certification Practice Statements [6]. Conversely, much is made within an identity-based infrastructure of off-loading policy creation to the end user,

while centralising policy enforcement. An example of this is the NHS trial for identity-based encryption which allows the sending party to encode the policy as the key [7][1].

In the case of a user in domain **A** sending an encrypted message to a user in domain **B**, which policy should the sender follow? If the identifier-based domain does not have the equivalent of the Object Identifiers (OIDs) in X.509 certificates, then how does the user in domain **A** know that the rules used in assigning private keys in domain **B** will conform to the desired policy?

We present two ways in which this problem could be solved.

- The TA in domain **B** could release a set of policy statements in a similar manner to a CP created within domain **A**. One of the stated benefits of ID-based cryptography is the ability for users within an ID-based domain to generate their own policies on the fly which can subsequently be verified at the TA before it issues the associated private key. In such cases, these client-generated policies are likely to be checked against more coarse-grained policies already held at the TA. We believe that such coarse-grained policies are likely to need refinement before being published — as in a CP. These policy statements could then be included in the cross-domain certificate, and hence can be checked by a user in domain **A**. While this would simplify the work for the user in domain **A**, it complicates the policy management within domain **B**.
- The sender within domain **A** could generate a new identity-based public key for the recipient, making part of the encoding a reference to the policy for the decryption and sending it with the message. The recipient in domain **B** would then need to go to their TA to retrieve the new decryption key. This would complicate the task for the sender, but would maintain the ability within domain **B** for the TA to validate the decryption request just prior to the decryption.

As can be seen from the above discussion, there are benefits and drawbacks to both mechanisms. We believe that the application requirements are likely to be a determining factor in deciding which mechanism to use in practice.

3.3 The ID-Based User

We next review the use of cross-certificates within the identity-based user's domain. A simplified overview of how the design would work is as follows.

- The TA for domain **B** generates a cross-certificate for the public key of the CA in domain **A**.
- This cross-certificate could contain policy information for domain **A** of relevance to a user in domain **B**.

[1] In this case the terminology changes slightly to that of identifier-based encryption, where the publicly verifiable information is an identifier that is used to encode the policy statement.

- A user in domain **B** validates the cross-certificate and checks the policy for acceptability.
- As the certificate is likely to be obtained from a local repository, there will typically be a need for a certificate status check, and possibly some additional path validation, to be carried out on the certificate.

This mirrors the design for the certificate-based user, as discussed in section 3.2. While this would give us the functionality we require for inter-domain interaction, we believe that it is a relatively inelegant solution. The main drawback for the users in domain **B** is that, before making use of the recipient's public key, they will be required to interact with another entity to fetch the certificate for the user in domain **A**. This means that one of the key advantages of using ID-based encryption is lost.

An alternative would be to have the TA in domain **B** issue an identity-based signing key to the CA within domain **A**. The CA could then dual-sign the certificates for its users, such that a user in domain **B** could validate the signatures directly using domain **B**'s system parameters. However, we believe that this approach is unlikely to be used in practice. The main reason for this is that, because of the very nature of identity-based cryptography, a signing key issued by the TA is escrowable (unless an approach such as CL-PKC, as outlined in Section 4.2, is used). Thus, a corrupt TA in domain **B** could falsify certificates or messages from the CA in domain **A**. The CA is unlikely to want to put itself in a situation where this is possible.

We now look at the certificate content, policy issues, and revocation from the perspective of an identity-based user.

Certificate Content. If we assume that the certificate is an X.509 cross-certificate, then the content is relatively well defined. The TA in domain **B** would, however, need to generate and use a different type of signature key in order to generate the certificate; this adds complexity to the scheme. However, the user in domain **B** would be expected to make use of the user certificates within domain **A** in order to enable secure communication, and thus it is not a significant additional burden.

CPs and CPSs. As we saw from our discussion in the previous section, the differences in the ways policies are managed in the two systems is one of the main hurdles to integration. The lack of prior work on CP/CPS equivalents in the identity-based domain would appear to complicate matters here.

Two candidate approaches for dealing with this problem can be identified.

- The users in domain **B** could be required to parse the certificate policies from domain **A** directly.
- The TA within domain **B** could parse and identify suitable CPs from domain **A**. The TA could then make available a list of the CPs which it deems acceptable for use within domain **B**.

The second option provides a much cleaner approach. Not only does it reduce the complexity of interactions at the user level, but it more accurately reflects the existing balance of policy checking in an identity-based environment. The TA is able to regulate the issuing of private keys on a more ad hoc basis, with some schemes requiring policy identifiers to be included within the public key to specify the policy that must be enforced when using that key pair [7]. This contrasts with the way in which clients in a certificate-based domain are expected to download the CPs of relevance and make local decisions before making use of a public key for encryption.

Revocation. Possibly the largest hurdle to interaction is the lack of explicit revocation within an identity-based domain. A 'pure' identity-based domain does not have explicit key revocation, but is more likely to rely on key re-issuance. If a user in domain **B** wishes to use a key for a user in domain **A**, then they are likely to need to validate the current certificate status. There are the three ways in which we believe this could be achieved.

- The CA could issue new certificates at the same rate as the TA would issue identity-based public keys for its users.
- The TA could act as a filter for user requests, where the TA performs the revocation interrogation on behalf of its clients.
- The user could be required to interrogate and interpret CRLs or OCSP servers directly.

The first two solutions are cleaner for users in domain **B**. The first solution is, however, likely to be too computationally intensive, as certificate distribution in a traditional PKI often incurs a large overhead. The second solution could result in the TA becoming a resource bottleneck, but fits closer to the identity-based model, where the TA is given the power to control key access within a secure environment. The third solution is simplest from an architectural perspective, but it increases the complexity of certificate processing at the user level. In practice, the application level considerations are clearly going to impact on this design decision.

4 Extending the Analysis

In this section, we briefly review some potential alternatives to the certificate-based solution presented above, along with additional technologies which could impact on an inter-domain design.

4.1 Potential Alternative Solutions

A high level outline of some possible alternative designs for inter-domain inter-operability is as follows.

- The design of the scheme presented in Section 3 requires the sender of an encrypted message to do the majority of the additional work. It would be possible to reverse this burden. For example, an identity-based user could act as though the recipient is "local" to the identity-based domain, and force the certificate-based user to retrieve the necessary identity-based private key from the TA in domain **B**.
- A trusted intermediary could be set up to act as a server that decrypts, then re-encrypts, the flow of messages. This would have no major impact in the identity-based domain, as the TA can already read anything that is sent to its clients (again, unless a scheme such as CL-PKC is used — see below). However, this might contravene the security policy in the certificate-based domain, where key or plaintext escrow is only carried out in exceptional circumstances.
- In the case of signatures, alternative solutions could be built that make use of the flow of signed messages. Much of the complexity introduced in the previous sections is designed to deal with the issue of retrieval of public encryption keys that is necessary before encryption can take place. However, because a signature has to be generated before any verification can take place, the verification key can be bound to the document or message being signed. This can greatly simplify the key retrieval protocol.

4.2 Additional Technologies

In this section we briefly outline some additional technologies which could impact on any cross-infrastructure design.

Certificateless Public Key Cryptography. One of the limitations of identity-based cryptography is the fact that the TA has the ability to escrow all private keys used in the system. Al-Riyami and Paterson developed Certificateless Public Key Cryptography (CL-PKC) [1,2] in order to overcome this problem (see also a variety of related work, including the notion of 'self-certified public keys', due to Girault [10], and Gentry's 'certificate-based encryption' [8]). The notion of CL-PKC gives the benefits of an identity-based public key mechanism, where the key pair is derived from publicly identifiable information. However, the private key is created in a joint process between the TA and the user, where the TA only knows a partial share of the resulting completed private key. This circumvents the key escrow problem.

Using a CL-PKC mechanism within the identity-based environment would address some of the problems we discuss in the previous sections, most notably as follows.

- It would allow the TA in domain **B** to give a signing key to the CA in domain **A**. This would simplify the parsing of the cross-certificates for the users in domain **B**.
- It would mean that private keys within domain **B** are more likely to fit the policies of use within domain **A**, given that most traditional PKIs completely reject the escrow of signing keys, and only carry out escrow of decryption keys in strictly controlled circumstances.

Certificate Chaining. Certificate chaining is a fundamentally important technique where multiple CAs are employed in certificate-based environments. Related work in the identity-based literature aims at generating tiered hierarchies of identity-based keys [9]. However, we believe that most keys within an identity-based domain are likely to be issued directly by the TA in a flat hierarchy within the domain. This would appear to make the best use of the control of private key issue inherent to identity-based cryptography.

The obvious question that results is to ask whether one could imagine a chain which consists of a mixture of certificates signed by regular CAs and keys/certificates generated by TAs? [2] If this were to occur it could potentially be both a benefit and drawback to the way in which such a chain would be processed.

- It would be of benefit if almost all of the keys in domain **B** are "grounded" at the TA. This would simplify the chain reduction algorithms when a client in a domain **A** is assessing a key from the identity-based domain.
- It would present a problem for users in domain **B** who are required to assess a certificate chain, as they are unlikely to be used to processing chains in an identity-based environment.

We note that both these assertions depend on how identity-based mechanisms are used in practice. It will thus be interesting to return to this question when more real-world examples exist.

5 Lessons Learnt and Future Work

In this section we highlight the early lessons we have learnt from this ongoing research. We also identify the avenues that we need to explore further in order to gain a better understanding of how the two infrastructures differ, and how that would impact on any potential interoperation.

- What are the main differences between the two infrastructure types which we have assessed to date?
  - We believe that the main differences lie in how the policy setting and validation is performed. Ensuring that the CP/CPS can be adequately expressed in an identity-based infrastructure will present a major challenge. Conversely, it is important to ensure that the CP/CPS of the certificate-based domain is adhered to by the TA when issuing keys that will be used in inter-domain communication. This work could require the development of a form of policy matching algorithm for the two domains.
  - Where and how the policy matching takes place can impact upon the implementation of that policy. In a certificate-based environment the policy is verified by the user before making use of the key. By contrast, in an

[2] We note that, in the scenario explored in Section 3, the cross-certificate from domain **B** to domain **A** and the user certificate signed by the CA already constitute a chain.

identity-based environment the policy is often verified by the TA immediately prior to the release of the private key. Taking these differences into account can be important when designing a security infrastructure.
  - It is important to understand the difference between what is being certified in a certificate-based environment, and what is built into an identifier in an identity-based environment (i.e. does the identity/identifier content correspond to the content of an X.509 certificate?).
- What additional work do we need for our analysis?
  - A clearer assessment and understanding of the differences in policy handling in light of actual application security requirements is needed. This can only come from genuine practical experience in rolling out ID-based infrastructures.
  - Although we only briefly discussed the potential use of attribute certificates in cross-certification, we believe that further research in this area is needed. One of the proposed strengths of identity-based cryptography is its ability to build authorisation policies and implement them directly into the key management infrastructure (see, for example, the trials carried out with the NHS in the UK [7]). Comparing how such use of identity-based cryptography might fit with attribute certificates would seem to be a logical next step.
- What shape should any further analysis take? Our next step will be to develop some example scenarios to provide us with a more detailed understanding of how the outstanding issues might be resolved.

Based on these early results, we believe that the difficulties we highlight above, combined with the technical solutions required, point us towards future use of a form of hybrid architecture, if interoperation is to be viable.

6 Conclusions

In this paper we have analysed a potential means of interoperation between a traditional certificate-based infrastructure and identity-based one.

Our main conclusion is that the existing technical solutions are far from ideal for the users in identity-based environments. The two main reasons for this are: the solutions either force the identity-based client to make use of certificates (bringing with it all the associated problems); or they require the use of trusted intermediates (e.g. trusted decryption servers, Delegated Path Validation-like services). Both of these solutions move us away from the benefits of identity-based cryptography.

In addition, it would appear that building mechanisms to reduce the impact of interoperation at the user level, forces us down a similar route to the proposals for path discovery algorithms in certificate-based environment. For example, it is likely that services would need to be set up to convert CPs in the certificate-domain to policy-centred identifier-based keys in the identity-based domain.

However, we close our discussion by highlighting the fact that interoperation between any security infrastructures can pose major headaches, and the difficulties we have highlighted in this paper are common problems.

Acknowledgements

The authors would like to thank Prof. David Chadwick and various anonymous referees for their contribution in helping clarify the presentation of some of the arguments within this paper.

References

1. S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In C. S. Laih, editor, *Advances in Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer-Verlag, Berlin, 2003.
2. S. S. Al-Riyami and K. G. Paterson. CBE from CL-PKE: a generic construction and efficient schemes. In S. Vaudenay, editor, *Public Key Cryptography — PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 398–415. Springer-Verlag, Berlin, 2005.
3. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.
4. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32:586–615, 2003.
5. L. Chen, K. Harrison, A. Moss, D. Soldera, and N.P. Smart. Certification of public keys within an identity based system. In A. H. Chan and V. D. Gligor, editors, *Information Security, 5th International Conference, ISC*, volume 2433 of *LNCS*, pages 322–333. Springer-Verlag, 2002.
6. S. Chokhani and W. Ford. RFC 2527: Internet X.509 public key infrastructure certificate policy and certification practices framework, March 1999.
7. Chris R. Dalton. The NHS as a proving ground for cryptosystems. Technical Report HPL-2003-203, Trusted Systems Laboratory, HP Laboratories, Bristol, October 2003.
8. C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer-Verlag, 2003.
9. C. Gentry and A. Silverberg. Heirarchical ID-based cryptography. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer-Verlag, 2002.
10. M. Girault. Self-certified public keys. In D. W. Davies, editor, *Advances in Cryptology — EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer-Verlag, 1992.
11. A. Menezes, P. C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.
12. K. G. Paterson and G. Price. A comparison between traditional public key infrastructures and identity-based cryptography. *Information Security Technical Report*, 8:57–72, 2003.
13. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.
14. D. K. Smetters and G. Durfee. Domain-Based Administration of Identity-Based Cryptosystems for Secure Email and IPSEC. In *Proceedings 12th USENIX Security Symposium*, pages 215–229, 2003.

XKMS Working Group Interoperability Status Report

Guillermo Álvaro[1], Stephen Farrell[1], Tommy Lindberg[2], Roland Lockhart[3], and Yunhao Zhang[4]

[1] Trinity College Dublin, Ireland
galvarorey@gmail.com
[2] Markup Security, Dublin, Ireland
[3] Entrust Inc., Ottawa, Canada
[4] SQLData, Virginia, USA

Abstract. The XML Key Management Specification (XKMS) is a World Wide Web Consortium (W3C) Candidate Recommendation. Current work on XKMS aims to demonstrate interoperability in order to progress the specification along the W3C standards track. This paper describes the state of that work and discusses some of the issues which have arisen during the course of the work.

1 XKMS

The XML Key Management Specification, XKMS[1][2], is a World Wide Web Consortium (W3C[3]) specification designed for retrieving and registering public keys. It is suitable for use in conjunction with the standard for XML Signatures[4] and its companion standard for XML Encryption[5] as well as in other application contexts (e.g., email).

XKMS consists of two different parts: the XML Key Information Service Specification (X-KISS) and the XML Key Registration Service Specification (X-KRSS). X-KISS defines a protocol to support the delegation by an application to a service of the detailed processing of key information associated with an XML signature, XML encryption, or other usage of the XML Signature <ds:KeyInfo> element. X-KRSS defines a protocol for the registration of a public key by a key pair holder, with the intent that the key subsequently be usable in conjunction with X-KISS or a Public Key Infrastructure (PKI) such as X.509[6] PKIX[7].

XKMS provides XML-friendly public key management, location and validation services which are roughly equivalent to a combination of the X.509/PKIX certificate management protocol (CMP[8]) together with the current work-in-progress simple certificate validation protocol (SCVP[9]).

We will now briefly describe XKMS, focusing particularly on those aspects of the specification necessary to understand the issues which have arisen during testing.

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 86–99, 2005.

1.1 X-KISS

Reducing the complexity of applications using XML Signature is one of the key objectives of the protocol design. X-KISS clients are relieved of the complexity of the underlying PKI used to establish trust relationships. These relationships may be based upon a different specification, such as X.509/PKIX, SPKI[10] or PGP[11].

In addition, sometimes the information provided by a signer can be insufficient for performing cryptographic verification or to be able to decide whether to trust a signature. Alternatively, the information provided by the signer may be in a format that is not supported by the client. In these cases communication with an X-KISS service can be useful as a way to get that "missing" information.

Examples where the key information could be insufficient for the client include:

- The key may be specified by a name only.
- The key may be encoded in an X.509 certificate that the client cannot parse.
- In the case of an encryption operation, the client may not know the public key of the recipient (e.g., just having a name).

X-KISS works via two different services: Locate and Validate.

Locate resolves a <ds:Keyinfo> element but does not require the service to make an assertion concerning the validity of the data in the <ds:Keyinfo> element. Validate does all that the Locate does, but in addition, the client obtains an assertion (at that time, according to that responder) specifying the status of the binding between the public key and other data, for example a name or a set of extended attributes. Furthermore the service represents that each of the other data elements returned are bound to the same public key.

Both Locate and Validate are implemented using request/response pairs of messages, derived from MessageAbstractType abstract type, which includes all types of messages. These message pairs are *LocateRequest* and *LocateResult* for Locate and *ValidateRequest* and *ValidateResult* for Validate.

1.2 X-KRSS

X-KRSS handles the registration and subsequent management of public key information. An X-KRSS service may bind information such as a name, an identifier or other attributes, to a public key, on reception of a client request. The key may be generated by the client or by the service on request. The Registration protocol may also be used for subsequent management operations including recovery of the private key and reissue or revocation of the key binding. The protocol provides ways of authenticating the requester and the possession of a private key. Additionally it provides a means of communicating the private key to the client in the case that the private key is generated by the registration service.

The operations constituting X-KRSS are:

- Register: Information is bound to a public key through a key binding. Generation of the key pair may be performed by either the client or the Registration service. The messages used are *RegisterRequest* and *RegisterResult*.
- Reissue: A previously registered key binding is updated. It is similar to the initial registration of a key and the principal reason a client would make a Reissue request is to cause the registration service to generate new credentials in the underlying PKI, e.g., X.509 Certificates. The messages used are *ReissueRequest* and *ReissueResult*.
- Revoke: A previously registered key binding may be revoked. A revocation request need only contain sufficient information to identify the key binding to be revoked and the authority for the revocation request. The messages used are *RevokeRequest* and *RevokeResult*.
- Recover: The private key associated with a key binding is recovered. The private key must have been previously escrowed with the recovery service, for example by means of the X-KRSS registration of a server generated key. The messages used are *RecoverRequest* and *RecoverResult*.

1.3 Bindings

XKMS specifies SOAP and HTTP protocol bindings[2] together with relevant security characteristics.

XKMS implementors are required to support SOAP 1.2 [12][13]. Bindings for both SOAP 1.1[14] and plain HTTP protocols are optional.

1.4 Processing Modes

XKMS supports different processing modes: synchronous and asynchronous.

- In synchronous processing the service immediately returns a message containing the substantive response to the request.
- In asynchronous processing the service returns a message to the effect that the request is not yet satisfied. A subsequent request/response (using a *PendingRequest*) pair is needed to complete the protocol.

Asynchronous processing may be used to allow administrator intervention during the processing of a request. For example an administrator might be required to verify and approve all X-KRSS Registration requests before they are processed.

A *StatusRequest* (and *StatusResult*) operation can be used to check the current status of an operation in asynchronous processing.

XKMS requests may also employ a two phase request protocol which is used to protect against denial of service attacks, as it allows the service to perform a lightweight authentication of the source of an XKMS request. The two phases of this protocol are as follows:

– Phase 1: The service responds to an initial request presenting a nonce.
– Phase 2: The requester sends the original request including the nonce.

The two-phase protocol may be combined with asynchronous processing. Such a scenario might consist of three round trips as follows:

– Initial request (phase 1)
– Initial request (phase 2)
– Pending request

1.5 Compound Messages

XKMS also supports compound requests and responses. A compound request permits multiple XKMS requests to be made at the same time. It consists of an outer request and one or more inner requests (X-KISS and/or X-KRSS). The semantics of making a set of requests as a compound request are "exactly the same as if each individual request in the set had been made separately and simultaneously"[1].

The response to a compound request is a compound response. A compound response consists of an outer response and zero or more inner responses. If the operation is successful the compound response should contain an inner response response element corresponding to each inner request element of the compound request.

The compound request/response pair elements are *CompoundRequest* and *CompoundResult*.

An XKMS service may support the use of the two phase protocol on the outer request of a compound message, but not in an inner request. Asynchronous processing may be used on the outer message as a whole or on individual inner requests or both.

1.6 Implementations

The current objectives of the XKMS Working Group [15] is to demonstrate interoperability. At the time of writing, interoperability tests had being conducted using seven client toolkit implementations and four servers (table 1).

An XKMS Implementation. In this section we briefly review one of the client implementations that took part in interoperability testing - the one developed at Trinity College, Dublin.

The client is basically a library - which is planned to be released shortly - and several programs that use the library. The development of the client API was done at two levels: A low level client written in C (with a length of about 4000 lines of code) and above that a higher level client library written in C++ (about 1000 lines of code long).

The library can deal with XKMS issues, and takes advantage of the benefits of the specification. For example, it can be used with X.509 certificates, even though the code doesn't know anything about X.509.

Table 1. XKMS Implementations

Implementation	Client	Server	Type
Trinity College Dublin	√		intended open source
Markup Security	√	√	possibly open source
Entrust Inc.	√	√	commercial
Oracle Corporation	√		commercial
Apache Software Foundation	√	√	intended open source
DataPower Technology, Inc.	√		commercial
SQLData Systems	√	√	commercial

Low Level Implementation. The low level API uses XMLSec [16], a C library based on LibXML2 [17] that supports the major XML security standards (XML Signature, XML Encryption, Canonical XML and Exclusive Canonical XML). As XKMS is deeply related to these standards, the XMLSec library proved very suitable for our implementation.

Functions defined at this level include:

- Ctx handling, used to deal with the creation and destruction of the processing context,
- Tree functions, used to add nodes and attributes into the tree structure of the message,
- Writing functions, used to generate request messages from the information in the processing context,
- Reading functions, used to parse response messages and get the information into the processing context.

It is entirely possible to construct an XKMS communication using only these functions. To achieve this the context structure must to be populated with the information for the request message - including the processing mode (synchronous, asynchronous, two phase), the SOAP binding used, etc.

Once the context is populated with the desired information, the function *xkmsProcCtxRequestWrite* can be called and the XKMS request message will be created, ready to be sent.

Once a response is received from the XKMS service, the function *xkmsProcCtxProcess* can be invoked to get all the information from the response message into the context. This information can be retrieved later directly accessing the context structure fields.

High Level Implementation. The low level API doesn't by itself provide a ready to use, easily configurable, solution for developing clients. In order to provide a better solution for developers of XKMS clients, a high level API in C++, that uses the low level C code, was also created.

The main new concept with this API is the use of message objects. There is a "Message" class that has some related functions and the different types of

XKMS protocol messages (Locate, Validate, Register, Compound, Pending, etc.) derive from it. The design of these classes was simple - given the XKMS schema and specification.

The functions offered by these classes modify the processing context using methods directly related to the message structures themselves. For example, there is a function (*xkmsAddKeyUsage*) to include a <KeyUsage> element that states the use (Encryption, Signature, Exchange) of the key, and so on.

Other information required by this high level API can be stored in a "configuration file", which can contain:

- the SOAP binding used,
- the Service access point,
- whether request messages will be signed or not,
- processing modes (asynchronous / two phase) and related information (e.g., where to direct pending notifications)
- whether the client expects the service to return the signature value of the request, (to check response authenticity),
- response limit (the number of bindings that the client is willing to accept).

Another file contains the relevant keys used by the client, like the one for signing the message - if it is going to be signed - or the trusted keys/certificates for signature verification.

Building programs using this API should be very easy:

- Initialize the libraries and the key manager that contains the keys and certificates,
- Initialize the desired message class,
- Fill the context with the information from the configuration file,
- Add the specific information that is going to be sent in the request,
- Communicate with the service,
- Store the relevant information from the context into files,
- Destroy message,
- Finalize libraries.

A Program Example. Below there is a short example of use of the high level API. The following fragments of configuration file and code demonstrate registering a server-generated key synchronously.

The configuration file might include the following:

```
\# FORMAT
\# Message Format
\# options: plain | soap11 | soap12
FORMAT =        soap12

\# SERVICE
\# Service where the requests are directed
\# example: SERVICE = http://foo/xkms
```

```
SERVICE =      http://www.example.com/xmks

\# SIGNATURE
\# Signature Flag
\# = YES, to sign the requests
\# = NO, not to sign the requests (set by default)
SIGNATURE = YES

\# ASYNCHRONOUS, TWO PHASE
\# Processing modes
\# = YES, if prepared to accept that mode
\# = NO, if processing mode not desired (set by default)
ASYNCHRONOUS =  NO
TWO PHASE =     NO
```

The program would include the following C++ code fragment:

```
xkmsInitialize();
mngr = xkmsCreateKeysMngr();
xkmsRegister message(mngr);
message.xkmsFillCtx("configuration-file");

message.xkmsAddRespondWith(xkmsRespondWithMaskPrivateKey);
message.keyBinding.xkmsAddKeyUsage(xkmsKeyUsageMaskSignature);

message.authentication.xkmsAddSharedSecret(NULL);
message.privateKey.xkmsAddPrivateKeySecret(NULL);

message.keyBinding.xkmsAddUseKeyWith("urn:ietf:rfc:2633",
                                     "me@example.com");

message.xkmsCommunicate();

message.privateKey.xkmsSavePrivateKey("keys/receivedkey.pem",
                                      pwd, codification);

message.~xkmsRegister();
xkmsFinalize();
```

Testing this Implementation. This client implementation took part in the interoperability tests. A C++ test program, using the library, was created for each of the tests (all of them, required and optional were supported).

Every test - required and optional - was successfully tested against the XKMS service of Markup Security, which also supported all of them. Additionally, the required tests were successfully tested against the XKMS service of SQLData Systems. Finally, some X-KISS tests involving Validate operations were successfully tested against the XKMS service of Entrust Inc., which offered a Validate service.

2 XKMS Interoperability

2.1 The Problem

To prove interoperability between XKMS implementations each feature of the specification has to be implemented and the W3C prefers two implementations of each. However, the W3C doesn't specify how to accomplish this. Ideally there would be several interoperable implementations covering all features. However, this is not the case since most current implementations feature some parts of the specification but not others, etc.

It is also important to remember that the purpose of the interoperability phase is to test the specification itself, not the implementations - there have been occasions when it has been hard to distinguish between testing the spec and testing the implementations.

Note that many classical software-engineering test procedures could not be used in this context, since most such approaches involve testing a single implementation against a test specification in order to test whether that implementation passes or fails. Here however, we are aiming to provide evidence that a specification supports interoperability. This is basically a different problem, as tests need to be run between independently developed clients and services.

One major issue to be tackled was how to be sure that every feature of the specification was covered by some test. We will now describe how the XKMS working group has handled this.

2.2 The Approach Taken

After studying what had happened in some other W3C working groups (in particular with respect to SOAP), the final approach taken to demonstrate interoperability had three different steps:

1. *Extract assertions from the specification.* Initially, all the assertions in the specification which contained keywords such as MUST or SHOULD were taken out.
2. *Try to cover each assertion at least once using a set of test scenarios.* Each test scenario is effectively an "executable" transaction and therefore may cover multiple assertions, though we try to minimise the number of assertions covered by each test scenario.
3. *Test clients against servers, reporting the results in an online form.* Each of the test scenarios has a different table allowing each implementation to report success or failure.

The assertions and the test scenarios are contained in the Test Suite document[18]. A password protected online form[19] enables developers to report on their implementations, and synopses of the form are posted to the XKMS working group mailing list periodically ([20]). The report form re-uses a script based tool which was previously used by the W3C as a questionnaire processor.

2.3 Difficulties and Special Cases

As with any interoperability demonstration, not all was smooth-sailing. We can divide the problems we found into test design issues, special difficulties caused by the peculiarities of the specification, changes in the specification and the way the results have to be interpreted.

Design of the Tests. An early issue was to agree on common parameters and conditions that would simplify testing. In particular, a common key set was defined in the test suite document so every server would know about it and so that tests could refer to existing keys. (This key set has had to be extended/regenerated a number of times as testing progressed which could have, but luckily didn't, cause difficulties for implementors.)

Covering each assertion with a good test scenario was a major issue. Separate tests were initially defined for each separate X-KISS or X-KRSS transaction or operation. This allowed for basic testing of features such as the types of messages, the SOAP bindings, the processing modes, etc.

Unfortunately, not all of the assertions found in the specification can be easily tested, and some of them are not testable at all. For example, some assertions explain the expected behaviour on receiving an incorrect message. To test that a service shouldn't for example accept an empty string as an identifier, a client emitting such a message would be needed, and that could be difficult to arrange. It was decided that it was impractical to test many such assertions.

There are also some inherently untestable assertions, for example: paragraph 160 of the XKMS specification says that "A Location service SHOULD attempt to provide only information which is trustworthy to the best of its knowledge but does not provide any assurance that it will do so. Information obtained from a Locate service SHOULD NOT be relied upon unless it is validated." We clearly cannot test this, and so it would have been better given our approach to generating assertions, had "should" been used there, instead of "SHOULD".

Even though we did not develop specific tests for these assertions, it was decided to leave them in the Test Suite indicating that the working group explicitly decided not to test them.

A problem related to X-KRSS is that some of its operations result in a state change in the server's database. For example, a key registered in a service supporting also X-KISS should be returned in response to a subsequent Locate operation. However, as it is not a requirement to support both services it wasn't a good idea to create a test that combined both operations since they might make a later conformance or regression test harder for an implementation. (This is an example of how testing the specification and testing the implementations may subtly differ!) In this case, some X-KRSS tests combined registration of a key binding with further X-KRSS operations (like a revocation of the key binding), so the change in the server's database was implicitly tested.

Peculiarities of XKMS that Made Testing Difficult. Like many protocols, XKMS is quite open and flexible, with many optional elements and with decisions

left to the implementations. As an example of this, there are (properly) few constraints on what an X-KISS service should return on receiving a request. There is flexibility for the implementor on how to match the information sent by the requester to the information that the service has. For example, if a service receives a request with elements referring to two different keys, the returned response is implementation dependent - there is no specification as to whether the response ought reflect the union or the intersection of the different keys. It is even possible that the same instance of a service could take different approaches influenced by the identity of the requester, from where the request is being made or about whom.

Obviously, not always knowing the expected results makes testing more difficult so some additional conditions have to be imposed, not for the specification itself, but for the tests.

Another problem was how to test a specific behaviour of an implementation where the openness of the specification allows other behaviours as well. For example, a registration request could take an entire day to be processed, if some human intervention is required. In that case we are dealing with an asynchronous operation in which a Status Request can be made, which can result in a "Pending" result code. As the specification doesn't define *when* this completion has to take place, it is a perfectly valid behaviour for a server to complete it immediately, which would make it impossible to get the "Pending" result code. A solution to this particular case was agreed by having the server implementation trigger (delayed) completion of the asynchronous request upon receipt of the StatusRequest. Special behaviours like this related to the tests had also to be included in the Test Suite specification.

Yet another particularity of the specification was how to deal with the existence of elements with no limit in the number of occurrences (maxOccurs="unbounded" in the schema). For interoperability it was agreed to interpret this as meaning that any relevant test message had to have at least two instances of the element in question. But this doesn't completely solve the question as we are not checking whether a service could handle messages with a huge number of occurrences. A solution to this is planned - essentially defining reasonable finite values for each unbounded occurrence in the next version of the base XKMS specification.

Compound messages are another peculiarity of XKMS that make interoperability difficult, as they may contain inner X-KISS and X-KRSS messages at the same time. This means that we need to combine the tests for the individual operations, which could lead to a combinatorial explosion in the number of tests. Also, as the semantics of making a set of requests as a compound request are (claimed to be!) exactly the same as if each of them had been made separately and simultaneously, special cases should be considered. For example: registering, retrieving and revoking the same key in the same outer compound message. Again, as the openness of the specification may not specify the expected behaviour in some of these cases, care must be taken to define the conditions and reach an agreement, at least for the interoperability demonstration purposes.

Changes in the Specification and the Schema. Some of the issues that have arisen during testing have pointed to differences between the specification text and the part of the schema they are describing. For example, it used be stated that it is possible to include inner Pending Requests in the outer Compound Request, whereas the XKMS schema didn't allow that. Generally implementors have followed the schema but each time this has occured we had to discuss whether a change should be made to the schema or to the specification.

There are also cases where the working group decided that schema changes were required. For example, it was decided that the name of an element and a type would have to be changed (RSAKeyValue and RSAKeyValueType to RSAKeyPair and RSAKeyPairType respectively) and (more subtly) that QNames would be replaced with URIs using an open enumeration technique.

The problem here was when and how to apply those changes. One possibility was to change the schema when reaching Proposed Recommendation status. However, this would prevent testing the changes on the schema until that point. (Essentially this is due to the W3C process for promoting the specification to the next level.)

Once the schema was modified, we then had to decide whether a new namespace is required or if the changes would be made into the current location (since the namespace is also a URL!).

Finally, changing the schema when some tests have already been made, raises another question: Should all the tests be made again using the new schema? As the change in the schema may affect the implementations it seems that it would be necessary to re-test. The problem is how to reflect the tests in the report site, should previous tests be deleted or should the new ones be added? The approach we have taken is to ask all implementors to do one last execution of all tests at the end of the process.

A small modification was done to the schema after the interoperability testing period: we added new result minor codes to indicate reasons for failure such as that certain elements were required (ProofOfPossessionRequired), not supported (OptionalElementNotSupported, TimeInstantNotSupported) or out of range (TimeInstantOutOfRange). These changes had no impact on the testing, but aim to avoid the need for out of band communication between parties (caused by the openness of the specification).

Interpreting Results. Having an agreement on the tests and assuming that the specification is going to be properly covered doesn't guarantee that all pitfalls of the specification are going to be discovered. Firstly, because there is not a great level of detail about reasons for failure in response messages, errors between implementations generally have to be diagnosed through direct contact between implementors.

Also, reporting the success or failure of the tests against the services in the report site was not an easily measurable action, except for completely successful tests. Moreover, as some implementations support only parts of the specification, summary results obtained from the report site mean less than first appears.

Ultimately, for each scenario, the overall success or failure of the test has to be determined by hand.

2.4 Current Results

This section documents the current state of testing in the middle and at the end of the interoperability testing period.

Table 2 summarises the test results at the middle of the interoperability phase.

Table 2. XKMS testing status, 28 October 2004

Tests [18]	Total # Tests	Success	Partial
XKISS-[T1-T5] (Messages)	5	5	0
XKISS-[T6-T8] (Protocols)	3	2	1
XKISS-[T9-T12] (Compound)	4	0	3
XKISS-[T13-T14] (Bindings)	2	2	0
XKRSS-[T1-T5] (Messages)	5	0	0

Table 3 below summarises the latest test results.

Table 3. XKMS testing status, 25 January 2005

Tests [18]	Total # Tests	Success	Partial
XKISS-[T1-T5] (Messages)	5	5	0
XKISS-[T6-T8] (Protocols)	3	3	0
XKISS-[T9-T12] (Compound)	4	4	0
XKISS-[T13-T14] (Bindings)	2	2	0
XKISS-[T15-T18] (Additional features)	4	4	0
XKRSS-[T1-T6] (Messages)	5	5	0
XKRSS-[T7-T9] (Protocols)	3	3	0
XKISS-[T10-T13] (Compound)	4	4	0
XKISS-T14 (Additional features)	1	1	0
Compound-T1 (Compound)	1	1	0
Optional-[T1-T3] (Additional features)	3	3	0

For compulsory tests, a status of success was assigned only for those tests for which at least two servers successfully passed the test with at least two clients (the XKMS working group's agreed success criterion). Partial success status was assigned where one server has worked with at least two clients.

The XKMS working group's agreed success criterion for optional tests was that at least one server had successfully passed the test with at least two clients.

Note that the testing protocol only required implementors to declare their results, there was no independent oversight of the results. This was the simplest way to proceed given that some of the implementations are commercial.

Counting success as one "point" and "partial" as a half, the above implies that testing (of X-KISS) was about 78 per-cent complete at the middle of this phase. However, more tests were added later.

Testing is a hundred per-cent complete now. A summary of interoperability can be found at [21].

At the end of the testing period, one service (Markup Security) supported all the tests (even the optional ones), and another service (SQLData) supported all the required tests. The ASF-XKMS service supported the X-KISS tests and Entrust's service supported the X-KISS tests involving Validate operations.

3 Conclusions

Advancing along the W3C's Recommendation track is quite a convoluted process which requires fulfilling various criteria including interoperability testing. Every specification has its peculiarities which can require a new testing scheme unlike any done by previous W3C (or other) working groups.

There are also difficulties related not only to the carrying out of the tests but also to the design of the tests themselves, as it is not practical to demonstrate interoperability for every single optional field in the entire specification. There is also serious additional complexity in testing compound messages which may not have been apparent when this feature was added to the specification.

However, the working group believes it has defined a reasonable set of tests which cover all the most interesting use cases for XKMS.

Good communication between working group members has been vital in making possible the work done to date, as has the significant amount of effort invested by both commercial and (putative) open source implementors. Without that level of effort, it is definitely the case that a significant number of bugs in the XKMS specification would have gone unnoticed. Even though none of the problems found has been very serious from a security or functional point of view, a number of them would have prevented interoperability.

Finally, it would of course have been better had the authors of the XKMS specification taken more account of testing at an earlier stage - in particular had they considered whether MUST/SHOULD type assertions could be tested or not, (though of course the working group had not decided to generate tests that way at that early stage - so here we are really asking for clairvoyance!).

Acknowledgements

The authors wish to acknowledge the XKMS Working Group members for their work, especially to Jose Kahan (W3C team member) and Shivaram Mysore (co-chair with SF).

GA's work on XKMS was supported by Microsoft Research.

References

1. P. Hallam-Baker and S. H. Mysore. *XML Key Management Specification (XKMS 2.0), W3C Proposed Recommendation*, May 2005. http://www.w3.org/TR/2005/PR-xkms2-20050502/.
2. P. Hallam-Baker and S. H. Mysore. *XML Key Management Specification (XKMS 2.0) Bindings, W3C Proposed Recommendation*, May 2005. http://www.w3.org/TR/2005/PR-xkms2-bindings-20050502/.
3. World Wide Web Consortium. *W3C*. http://www.w3.org/.
4. D. Eastlake, D. Solo, M. Bartel, J. Boyer, B. Fox, and E. Simon. *XML-Signature Syntax and Processing, W3C Recommendation*, February 2002. .
5. D. Eastlake, J. Reagle, T. Imamura, B. Dillaway, and E. Simon. *XML Encryption Syntax and Processing, W3C Recommendation*, December 2002. http://www.w3.org/TR/xmlenc-core/.
6. International Telecommunications Union-Telecommunication. *ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*, June 1997.
7. R. Housley, W. Polk, W. Ford, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, IETF RFC 3280*, April 2002. http://www.ietf.org/rfc/rfc3280.txt.
8. C. Adams and S. Farrell. *Internet X.509 Public Key Infrastructure Certificate Management Protocols, IETF RFC 2510*, March 1999. http://www.ietf.org/rfc/rfc2510.txt.
9. T. Freeman, R. Housley, A. Malpani, D. Cooper, and T. Polk. *Simple Certificate Validation Protocol (SCVP)*, February 2005. http://www.ietf.org/internet-drafts/draft-ietf-pkix-scvp-18.txt.
10. C. Ellison. *SPKI Requirements, IETF RFC 2692*, September 1999. http://www.ietf.org/rfc/rfc2692.txt.
11. J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. *OpenPGP Message Format, IETF RFC 2440*, November 1998. http://www.ietf.org/rfc/rfc2440.txt.
12. M. Gudgin et al. *SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation*, June 2003. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.
13. M. Gudgin et al. *SOAP Version 1.2 Part 2: Adjuncts, W3C Recommendation*, June 2003. http://www.w3.org/TR/2003/REC-soap12-part2-20030624/.
14. D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S Thatte, and D. Winer. *Simple Object Access Protocol (SOAP) 1.1, W3C Note*, May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.
15. XKMS Working Group. Xml key management working group. http://www.w3.org/2001/XKMS/.
16. A. Sanin. *XML Security Library*. http://www.aleksey.com/xmlsec/.
17. D. Veillard. *Libxml2 - The XML C parser and toolkit of Gnome*. http://xmlsoft.org/.
18. G. Alvaro. Xkms assertions and test collection. http://www.w3.org/2001/XKMS/Drafts/test-suite/CR-XKMS-test-suite.html.
19. XKMS Working Group. Xkms cr test-suite report. http://www.w3.org/2002/09/wbs/1/XKMS-WG-CR-TEST-SUITE/.
20. XKMS Working Group. Final results of questionnaire xkms cr test-suite report. http://www.w3.org/2001/XKMS/Drafts/test-suite/results-snapshot.html.
21. XKMS Working Group. Xkms candidate recommendation implementation report. http://www.w3.org/2001/XKMS/Drafts/test-suite/CR-XKMS-Summary.html.

An Innovative Policy-Based Cross Certification Methodology for Public Key Infrastructures

Valentina Casola[1], Antonino Mazzeo[2], Nicola Mazzocca[2]
and Massimiliano Rak[1]

[1] Seconda Universita' di Napoli
Dipartimento di Ingegneria dell'Informazione
Aversa (CE), Italy
{valentina.casola, massimiliano.rak}@unina2.it
[2] Universita' degli Studi di Napoli, Federico II
Dipartimento di Informatica e Sistemistica
Naples, Italy
{mazzeo, n.mazzocca}@unina.it

Abstract. Cross Certification among CAs is a very huge problem which is actually manually performed by security experts and organizational people, trying to understand if two CAs could cooperate. The evaluation process is based on the evaluation of the Certificate policies which are usually expressed in a not formalized (and native language) way. In this paper we propose a methodology to automatically evaluate and compare security policies for Cross Certification. The methodology consists in the formalization of a policy template and in the building of a reference evaluation model. The proposed approach can be applied on several models of Cross Certification.

1 Introduction

The security of complex infrastructures depends on many aspects, both technical and organizational and they need to be properly addressed, in a unifying way, by a security policy after the analysis of the resources to protect and of all the threats and risks involved.

Security policies are made of a "set of statements on what is, and what is not, allowed"; they can be expressed and formalized in many different ways and, consequently, the mechanisms to enforce them, may be procedural, technical or physical. Incorrect policies and incorrect mechanisms represent the main security threats for a system and their ambiguity is often the primary reason for which a security expert is not able to completely trust a system declared "secure". Really, when system administrators use the term "secure", they simply intend that they have adopted some special kind of security mechanism; security mechanisms could vary the amount of provided security, they are able to meet requirements and specification but may be not sufficient to get trust in the system.

Adoption of security policies in Public Key Infrastructure is a common approach. Usually each Certification Authority (CA) expresses the set of adopted

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 100–117, 2005.

rules for the common usage of its certificates. A common problem is how to help different CAs to reach an agreement in order to *cross certify* their users, i.e. each accepts the certificates of the other, assuring trust. As the state of the art, the main solution focuses on manual evaluation of the proposed policies and mutual agreement from experts of the two CAs.

Within this context, the main research issue that we face in this paper is related to a quantitative evaluation of the system security and, specifically, to certificate policy comparison for Cross Certification. We aim at developing a methodology to automate or semi-automate the process of cross-certification.

The approach we propose is to build a **Reference Evaluation Model** to evaluate and compare different security policies, quantifying their "security level". The model will define how to express in a rigorous way the security policy (formalization), how to evaluate a formalized policy, and what is its security level.

In order to face the problem of heterogeneous origin of the policies, the proposed approach is flexible and each step can be adapted to specific problem requirements. We propose one or more solutions both for policy formalization and evaluation techniques.

The reminder of this paper is structured as follows: Section 2 will introduce the problem of PKI cross certification and security policy evaluation. Section 3 introduces the **Reference Evaluation Model**, exploiting for each component of the model how to build correct and usable solutions. By examples we will show some solutions for policy formalization and evaluation techniques. Section 4 shows how to apply the model in several cross certification models. Finally Section 5 contains some conclusions.

2 PKI Cross Certification

The procedures and protocols that Certification Authorities (CAs) and Registration Authorities (RAs) use to manage certificates are usually very complex and have critical security requirements. When the procedures are well defined and controlled, the security and trustability associated with a certificate is high. All procedures and protocols are described in a Certificate Policy, so we can gather information on the security level by analyzing the description of all policy provisions. A Certificate Policy may be used by a certificate user to help him in deciding whether a certificate, and the binding information therein, are sufficiently trustworthy for a particular application; a certificate policy must be used by different CAs when they need to extend their trusted domains (Cross Certification) and interoperate [9,12,15,18].

At the state of the art RFC3647 [21]is the main reference in building certificate policies, but not all the national laws and legal CAs follow the standard; this is a very huge problem, especially during the comparison process, when two different Certification Authorities (CAs) need to extend trust to each other.

The action of extending trust is usually referred to as Cross Certification and it usually involves the cooperation of both technical and organizational people

from the CAs that need to trust each other to completely understand if they are able to interoperate in terms of technological implications and guaranteeing the same security level to their users.

This problem is actually faced as an "organizational problem": security experts, from both technological and organizational fields, manually evaluate the different provisions implemented by the two CAs and, by manually comparing each provision, they finally decide to cross-certify them or not. The diverse perspectives of policymakers, including legislators, industry representatives, and business associations, have resulted in divergent approaches to facing key issues within different jurisdictions; for this reason, up to few years ago, there were very different models of certificate policies, too.

Cross Certification is not simple at all, since it involves not only technological aspects (partially resolved by the standards X.509, RFC2459, RFC3647 PKCS family, and so on [20,21]) but, above all, organizational and liabilities-related aspects. To simplify the work, each CA builds a table with all its critical security provisions on the rows and, on the columns, they respectively put the provision instances implemented by the two CAs; only after an accurate evaluation of the table, they decide to cross certify or not. The building and evaluation of this table is not simple, as well, because it is not obvious that the two CAs consider critical the same provisions and it is not obvious that technical and organizational people agree on the evaluation of the single provisions.

The target of our research is the definition of an automatic way to combine separate systems into larger connected "networks of trust". Actually, to do this, each component within a network of PKIs reliably trusts digital certificates generated by the other components when explicit cross certification models are used [18,19,22]:

Hierarchical model: In which a root CA exists, and it guarantees down to the level of the leaf-CAs.

Bridge Certification Authority Architecture: In which there is an explicit agreement between two different CAs, completely independent of other agreements.

Mesh architecture: CAs make agreements with each other, building a *mesh* of trusted CAs.

Each model has strengths and weaknesses and a clearly superior method has yet to emerge; they all have a common point: the policy mapping, which actually consists of a "manual" comparison of all provisions.

3 The Methodology

The state of the art security evaluation for secure infrastructures is manually performed by technical and organizational experts. However, security evaluation must face uncertainties derived from different perspectives, different verbal judgments, different ways to express and formalize security statements and lack of information.

Our innovative methodology proposes to characterize a policy and to define metrics with different security levels against which we are able to evaluate and compare policies. The comparison has to give clear information about policy weakness that could be used to help security administrator to eventually enforce better rules.

Our methodology takes into account the security criteria of the evaluators on one side and the features of the evaluated policy on the other side. The methodology core is the **Reference Evaluation Model (REM)**, that is a general model, adopted for security evaluation.

The proposed methodology builds on the following steps:

1. choose the cross certification model,
2. build a Reference Evaluation Model,
3. define the cross certification rules.

The REM building is the main step of the proposed approach, in particular, the REM should face the following problems:

- In order to (automatically) compare different security policies, policies should be described in a rigorous, controllable way. This implies that informal or semiformal descriptions should be translated in some way.
- Even if a formal description of the policy is available, how to quantify the system security? The REM should define evaluation metrics and techniques.
- Usually (but not always) a CA expresses security through a set of "security levels" which are related to different classes of certificates [11]; such levels are represented by a growing scale of security policy instances. The REM should help in defining this levels, and in assigning a given level to a policy.

Due to the above considerations, we define the "Reference Evaluation Model" (REM) as the following 3-pla:

$$REM =< Formalization, Technique, ReferenceLevels >$$

where:

Formalization represents the formal (semi-formal) representation of the policy. Examples of formalization are expressed by a tag language (an XML schema or DTD) or by a mathematical representation (like a matrix of a given dimension). The chosen formalization will affect final evaluation, and will be built by taking into account the adopted PKI architecture for the evaluation (see section 2).

Technique represents the evaluation technique that can be applied to compare policies; the evaluation technique strictly depends on the policy formal representation. Evaluation techniques should respect some criteria, that grant their correctness.

Reference Levels Reference Levels are instances of policies, which represent different security levels. If the formalization is an XML document, levels are valid XML documents which represent different policies with a different

security level; if the formalization is a matrix, reference levels are matrix instances, where each element assumes a valid value and which represent different policies with a different security level. Evaluation technique should assure that the numerical evaluation of growing policy security levels results in growing values. This REM component is optional, because not always the evaluation will be expressed in terms of security levels. Section 4 will investigate the problem.

The REM gives a powerful means to evaluate and compare policies. REM evaluation will give useful indications for establishing how to perform cross certification between different CAs. In the following sections we will present the steps of the methodology to build valid and usable REMs, adopting different formalization and evaluation techniques. In Section 4 we will explain how to effectively use the proposed approach in order to perform cross-certification.

3.1 Policy Formalization

As already mentioned, the formalization is the first necessary step for the introduction of our methodology, so in this Section, we first illustrate some criteria to classify security policy types and then we will face the problem of policy formalization

Types of Security Policies. Policies can be expressed and formalized in many different ways, they could be informal (consider for example "some good practices to choose a password") or highly mathematical. We could identify a type of policy by the way in which it is expressed; for example we could classify the following types of policies:

Formal policies are usually expressed by mathematical or machine-parsable statements and languages.
Semi formal policies are partially expressed by machine-parsable languages.
Informal policies are usually expressed in a very informal language, with statements often ambiguous and or expressed in a free textual form.

Notice that formal policies are typically expressed by technical staff-members who need to express in an unambiguous way technical procedures, while organizational members who often need to express practical and behavioral aspects of the organization of a secure site typically prefer informal policies. Both technical and organizational aspects are very critical for security but often members of one part do not understand the criticality of the other ones. The more a policy is formalized, the more the evaluation process is easy when performed by an automatic machine; on the other side the evaluation process becomes very difficult for a non-technical member who needs to read the security statements.

The classification is not exhaustive, it intends to make the reader more sensitive to the policy formalization problem in terms of what to express in a security policy and in which formalism to express it [1,16,17].

Recently, some efforts to formalize PKI certificate policies and the related security provisions have been produced; the formalizations proposed in [5,6][13,14] help system and security administrators in the policy life cycle management.

Really, all policy frameworks present wide limits; they certainly represent a valid means to develop a textual policy, but they do not resolve ambiguity problems, they are not sufficiently structured to be used as a valid mean to evaluate and compare policies.

An Example of Formalization. Probably, as mentioned before, the most detailed and relevant suggestion for the formal presentation of certificate policies was described in the Internet RFC3647. It is not a standard but it is actually widely used by all the Internet Community and for all these reasons we have decided to choose its main provisions and structure for the first steps of our formalization.

We primarily underline that the framework is structured as a hierarchical tree.

Textual provisions were refined in a more fine-grain and a grammar to automatically compare them was proposed in [5]. We will adopt this approach to define a policy tree formalization.

All the macro-provisions are very complex objects categorized for different topics that need to be addressed in a Certificate Policy. According to RFC3647, the first level provisions include:

- Introduction
- General Provisions
- Identification and Authentication
- Operational Requirements
- Physical, Procedural and Personnel Security
- Technical Security Control
- Certificate and CRL Profiles
- Specification Administration

Second level provisions try to describe all the details about all macro-provisions and they express objects that are still complex but bring a more bounded security information; for example the Technical Security Control provision includes: Key Pair Generation, Private Key Protection, Other Aspect of Key Pair Management, Activation Data, Computer Security Controls, Life Cycle Technical Control, Network Security Control, Cryptographic module engineering controls.

The provisions defined in the first two steps are very complex objects and this is the most important reason for ambiguity; to solve the ambiguities the proposed model supports a hierarchical structure which consists of several couples (element-type, value) representing topics and sub-topics, where the "value" itself is a complex object. A wide range of new data-structures has been defined to represent the values, and finally a grammar has been created based on such data-structures to formalize a certificate policy. The used data-structures are new atomic or enumerative types and total order relations among their values

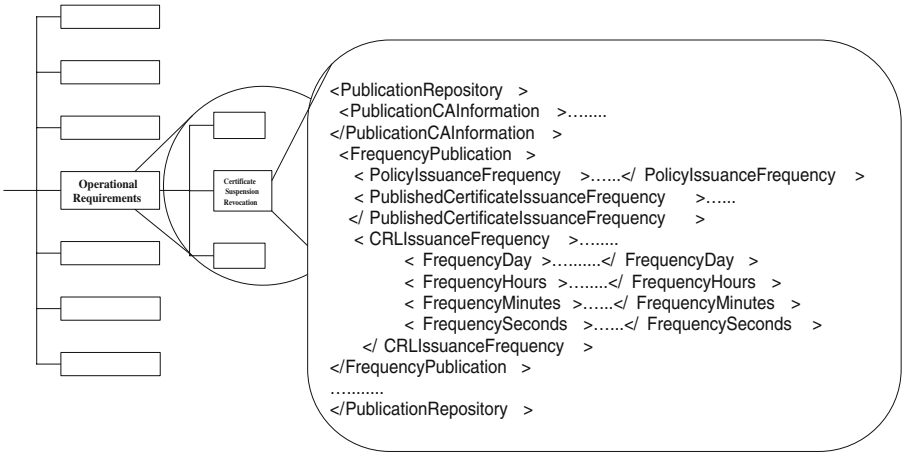

Fig. 1. Decision Graphics Example

may be defined, so as to solve the ambiguity problem; we will associate a Local Security Level to each provision instance. For most critical topics we were able to build such a structure that could be automatically processed by a numeric algorithm.

The proposed structure is a hierarchical tree (Figure 1) that can be represented by an XML document; tree nodes identify complex security provisions, leaves identify simple security provisions. Examples of such formalization are available in [6].

This representation is very useful for security experts and PKI experts, who know in detail the CA policy manuals, but they are not useful in terms of evaluations. Furthermore alternative representations can be easily derived; for example XML documents can be represented as trees and the set of leaves of the XML tree can be represented as a vector. In the following we will use both these syntactical representations for policy evaluation.

3.2 Evaluation Techniques

A formalized policy instance expresses in a rigorous way, who, how and where security principles will be applied. This will help in comparison of security choices, but it is not enough. REM includes the definition of the technique adopted to compare and evaluate the policy; we call this component the *REM Evaluation technique*.

Different evaluation techniques characterize a policy in different ways, for example with a numerical value, a fuzzy number or a verbal judgement representing its security level.

Even if final evaluation interpretation depends on the objectives of the study, adopted techniques must respect some principles which grant significance to the value expressed:

1. given two policies `A` and `B`, if all security clauses of `A` can be evaluated *more secure* than the corresponding clause in `B` then policy `A` must be evaluated better than `B`. This rule grants coherence of the evaluation technique.
2. given an existing policy with `n` growing security levels, evaluation of such levels, according to the chosen technique, should result in growing security levels. This means that the evaluation technique and the real model (previously defined by security experts) agree about the security level of the proposed policy instances.

In the following we introduce two different evaluation techniques: the first one is based on fuzzy theory to represent and evaluate policies [7] and the most innovative contribution is on the use of fuzzy numbers to analyze and evaluate security through policies. The second technique is based on an innovative definition of metric policy spaces by which we could represent policies. By applying an Euclidean distance criterium applied to the metric policy space we can compare policies by measuring their distance.

For brevity sake we could not give details about how we have built those techniques, but, in this paper we will give clear information on how they are able to represent and evaluate policies.

The Fuzzy Technique. In this section we will introduce how a fuzzy technique could be adopted to represent and evaluate the security associated to a certificate policy.
The main characteristic of this technique could be summarized as follows:

- All provisions of the policy tree are translated into a fuzzy judgement which expresses the Local Security Level of each provision [7];
- A fuzzy judgement can be represented by a pair (p,s), where p is the ordinal position of the label in the chosen scale of judgment and s is the number of labels considered by the scale i.e. the number of Local Security Levels for that provision.
- The pairs are translated into fuzzy numbers with triangular shapes. Their shape is characterized by these characteristic points:

$$x_L = \frac{p-2}{s-1}; x_M = \frac{p-1}{s-1}; x_R = \frac{p}{s-1} \tag{1}$$

  The distance between x_L and x_R determines the base of the triangular shape and it expresses a measure of the judgement uncertainty; x_M determines its vertex orthogonal to the base, it expresses a measure of a traditional judgement (not fuzzy crisp value).
- A policy is characterized by the aggregation of fuzzy judgements on structured provisions through the OFNWA (Ordered Fuzzy Number Weighted Averaging) aggregation technique [2,3]; the result of judgements aggregation is a global judgment of the policy that we interpret as the Global Security Level of the policy (GSL for shorts).

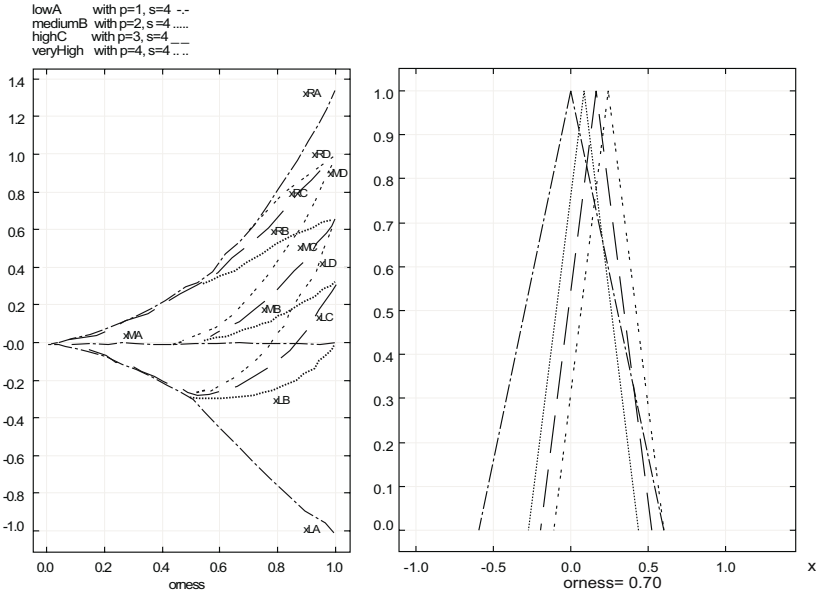

Fig. 2. Decision Graphics Example

- The aggregation result is a fuzzy number, too. It is represented by a triangular membership function, where x_M expresses the Global Security Level of the policy under evaluation while the distance between x_L and x_R gives a measure of the evaluation uncertainty.
- Judgements aggregation takes in count the tree structure of the policy and not only the value assumed by its provision-leaves; it is possible to evaluate intermediate nodes, too, and furthermore, it is possible to give different importance and/or aggregation weights to intermediate nodes according to the evaluator needs and experience.
- There is another important parameter that lets the policy evaluator to decide his severity, the *orness* parameter.

For brevity sake, further details are provided in [7,8]; in the remainder of this section we will show an example of policy representation through Decision graphics [4,8]. An example of these graphics is shown in figure 2.

As shown, there are two types of graphics:

1. orness variable graphics (left side);
2. x variable graphics (right side).

These graphics are obtained by processing four different instances of a built evaluation graph for a given security policy. The selected instances differ only in one provision.

On the left-hand side of figure 2 the orness value is represented on the x axis, while the aggregation result is represented on the y axis. Since each triangular fuzzy number can be completely described by the support $[x_L; x_R]$ and the prototypal value x_M, the evaluator can analyze how these characteristic points change by changing the orness value in the interval `0` (where the evaluator's severity is maximum) and `1` (where the evaluator's severity is minimum).

These characteristic points are represented by the three curves named x_L, x_R, x_M. In particular in the example of the figure we can observe four different sets of these three curves, each one representing a different policy instance.

By analyzing these graphics, the evaluator can get information concerning the impact of the uncertainty of the initial judgements on the global security level of a selected policy measuring the distance between the curves x_L and x_M and the curves x_M and x_R. Increasing these distances, fuzziness of the evaluation grows.

In the figure we can observe that these distances grow by changing the orness; moreover we can observe that when orness is lower than `0.5` the final judgement is not affected: when the orness is low, the severity of the evaluator is high, so even if one clause is better, the worst evaluation affects the judgement. Note that at orness 0, the evaluation is zero crisp; this usually happens when there is almost one "non stipulated" provision in the policy, which is treated as the worst judgement in the policy (named zero crisp) .

The right-hand side of figure 2 shows the *x variable graphics*. By choosing an orness value in the interval [0,1], the evaluator can obtain the triangular fuzzy numbers representative of the selected policies instances evaluations for each selected orness value. These graphics help the evaluator to better understand the aggregation continuum provided with OFNWA operators.

The Metrical Space Technique. The main characteristic of the metrical space technique could be summarized as follows:

- After the formalization, each provision is represented by an enumerative data-type; the policy space is defined as the vectorial product of all provisions K_i . P = K1 x K2 x . x Kn where $n = 1.m$
- The policy space is homogeneous thanks to threshold functions (F-functions) which allow us to associate a Local Security Level to each provision.
- The policy space is represented by a matrix whose rows represent the single provisions K_i. For example, if the LSL associated to a provision is l_3, the vector corresponding to its row in the matrix is: (1,1,1,0).
- The evaluation process takes into account just the provisions of the policy which represent the leaves of the policy tree structure.
- The policy space is represented by a *n x 4* matrix (where n is the total number of provisions and 4 is the number of Local Security Levels (LSL for short) admissible for each provision).
- The distance criterium for the definition of the metric space is the Euclidean distance among matrices, defined as:

  $d(A, B) = \sqrt{(\sigma(A - B, A - B))}$
  where $\sigma(A - B, A - B) = Tr((A - B)(A - B)^T)$

provision name				
Local Registration Authorities (LRAs)	1	1	1	1
Repositories	1	1	1	0
Policy applicability	1	1	0	0
Notification of certificate issuance and revocation	1	1	1	0
Time between certificate request and issuance	1	1	1	0

Fig. 3. An example of Policy provisions

In figure 3 an example of provision representation is reported.

To show that the defined distance really represents the distance between policies, we will give two examples; the policy P is compared with two different policies X and Y; they are all stronger than P but with different GSLs. Each policy in the example has just 10 provisions, this is just a simplification which does not affect the validity of the method; in the real cases we usually have fifty or more provision leaves.

$$\mathrm{P} = \begin{pmatrix} 1\,1\,0\,0 \\ 1\,1\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,1\,0\,0 \\ 1\,0\,0\,0 \end{pmatrix} \quad \mathrm{X} = \begin{pmatrix} 1\,1\,1\,0 \\ 1\,1\,1\,0 \\ 1\,1\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \\ 1\,1\,0\,0 \\ 1\,0\,0\,0 \\ 1\,1\,0\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,0 \end{pmatrix} \quad \mathrm{Y} = \begin{pmatrix} 1\,1\,1\,1 \\ 1\,1\,1\,1 \\ 1\,1\,1\,0 \\ 1\,1\,0\,0 \\ 1\,1\,1\,0 \\ 1\,1\,1\,1 \\ 1\,1\,1\,1 \\ 1\,1\,0\,0 \\ 1\,1\,0\,0 \\ 1\,1\,1\,1 \end{pmatrix}$$

Example 1: X is a policy that appears stronger than P, just looking at the levels of the single provisions; we first calculate the trace:
Tr((X-P)(X-P)^T) = 6
The distance between X and P is: $d = 2,45$
That mirrors the fact that X is just a little stronger then P.

Example 2: Y is a policy that appear stronger than X and much stronger than policy P, while P is the same as that of the example 1; the trace is:
Tr((Y-P)(Y-P)^T) =19
The distance between Y and P is: $d = 4,36$
This result mirrors the evident difference between the two cases.

3.3 The Reference Levels

The last component of the REM is the set of reference security levels that could be used as a reference scale for the numerical evaluation of security. When references are not available, the REM is used for direct comparison among two or more policies.

To properly choose the references, we can proceed in two different ways:

- when possible, if n different policy instances are available and they certainly correspond to n different security levels, then we could use those ones as reference levels;
- when they are not available, we need to define an appropriate set of policy instances.

At this point we need to represent reference levels according to the chosen technique of the REM. For example we could choose as reference levels the four policies of the Government of Canada which really correspond to four security levels [23]. If we refer to the fuzzy technique the graphical representation of the four Canadian policies could be reported as illustrated in figure 4. Details on how to use these graphics in the comparison process are given in [8].

If we refer to the metrical space technique we first evaluate the distances among the references (denoted as GofCi) and the origin of the metric space (denoted as $\emptyset$), then define the metric function which gives the resulting level as follows.

The numeric values for the references are:

d_{10} = d(GofC1,$\emptyset$) = 7, 07
d_{20} = d(GofC2,$\emptyset$)= 11, 18
d_{30} = d(GofC3,$\emptyset$)= 12
d_{40} = d(GofC4,$\emptyset$)= 12, 65

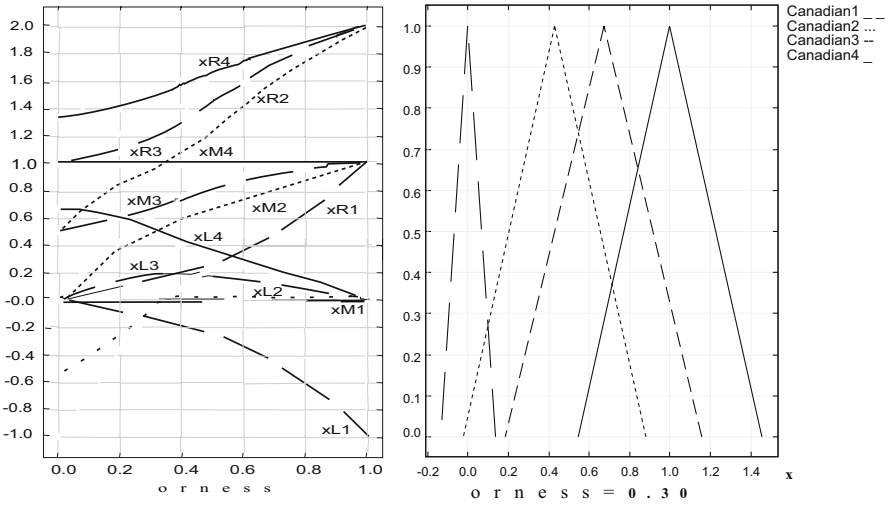

Fig. 4. Decision Graphics for Security Policy Levels

The proposed *security metric function* to evaluate the Global Security Level (GSL) of a generic policy P_x is:

$$L_{Px} = \begin{cases} L_0 \; iff d_{x0} \leq d_{10} \\ L_1 \; iff d_{10} < d_{x0} < d_{20} \\ L_2 \; iff d_{20} < d_{x0} < d_{30} \\ L_3 \; iff d_{30} < d_{x0} < d_{40} \\ L_4 \; iff d_{40} \leq d_{x0} \end{cases}$$

where L_{Px} is the GSL of P_x

4 Methodology Applicability

As previously described, the main cross certification problem is about how a new CA should be trusted enough to be added to an existing PKI model to inter-operate with other CAs. All the existing models imply that the new CA policy has to be evaluated, in order to establish its security level. We explicitly note that, when the model changes the evaluation criteria, the elements which should be evaluated and the reference levels could change.

In this section we will use the following acronyms:

Trusted Group (TG) a set of CAs, which trust each other thanks to one of the models introduced in Section 2.

Trusted Group levels a set of reference security levels common to the TG.

New Certification Authority (NCA) the Certification Authority which aims at being added to an existing TG.

New Certification Authority Policy (NCAP) The policy proposed by the NCA.

Adoption of the REM helps in all these approaches, giving a tool which helps in automating the evaluation process. In the following we will apply the evaluation techniques through the REM in the three models for cross certification described in section 2.

4.1 Hierarchical Model

When a NCA should be added to a hierarchical TG model, its policy needs to be evaluated against its father node in the hierarchy. Evaluation consists in assigning a security level evaluated against the father node reference security levels.

In this case the formalization step of the REM building phase will take place using the father-node policy as the policy template. For example if the formalization approach chosen is the proposed XML tree, it will represent the formalization of the father-node policy. The choice of the REM technique can be carried on taking into account that the rules of the father-node CA *must* be respected. So, flexible judgements, like the ones supported by the fuzzy evaluation technique, are probably not useful and the result of the evaluation technique should be a single number.

Security reference levels are defined by the father-node CA too.

Once the REM is built, the NCAP evaluation is made up of the following steps:

- the NCAP is formalized according to the father-node policy template;
- the NCAP is evaluated with the chosen REM technique and a Global Security Level is assigned to it according to the adopted technique (see Section 3.3).

The result of the methodology is the numerical Global Security Level of the proposed policy. NCA certificates will be accepted at the agreed security level.

Example. As already said, for this type of model the metrical space technique is more useful; in this case, according to the definition of metric function given in Section 3.3, we need to evaluate $d_{NCAP,\emptyset}$ and compare this distance against the references.

4.2 Bridge Certification Model

When cross certification takes place through a Bridge Certification Authority (BCA), the evaluation process is different from the previous approach. In this case the BCA defines the rules of the comparison, but not necessarily some reference security levels. Both CAs propose their policy with their eventual security levels and the BCA will define what are the two CAs Level correspondences according to its own policy template and formalization.

In this case the formalization step of the REM building phase will take place using the BCA policy template. The bridge authority chooses what are the security features each policy must or should respect and both the CA policies are modelled according to the BCA formalization. Possible cross certifications among policy levels of each CA could be expressed as a correspondence table which expresses security equivalence between two security levels of the cooperating CAs. Note that in the cross certification process certificate acceptance is bidirectional, it is never accepted that a certificate of CA1 at level i is accepted by CA2 at level j, and not viceversa.

Example. We want to cross certify two significant certificate policies:

1. EuroPKI Italian Certificate Policy [10];
2. Manuale operativo per il servizio di certificazione di chiavi pubbliche per la rete unitaria della pubblica amministrazione (registered name of Centro Tecnico, an Italian CA with legal validity, CT for short) [24].

The BCA does not have any absolute reference but could introduce its own; in this example we establish that the reference levels are all equidistant and the GSL of a policy of level L_i is defined by the matrix with all "1" in the corresponding LSL (l_i). By applying the metric function we obtain that the CT

policy is of level L_3; in fact:

d(CT,$\emptyset$)= $\sqrt{149}$=12,21 and $d_{30} < d_{CT,\emptyset} < d_{40} \Rightarrow L_{CT} = L_3$.

The EuroPKI is of level L_2, in fact:

d(EuroPKI,$\emptyset$)= $\sqrt{102}$=10,10 and $d_{20} < d_{EuroPKI,\emptyset} < d_{30} \Rightarrow L_{EuroPKI} = L_2$

.

Note that the $d_{X,0}$ values are not the ones evaluated in Section 3.3; they were, in fact, the real references from the Government of Canada while, in this example, we have chosen different, and *ad hoc*, references.

Furthermore, the two CAs could not cooperate in this way. We could think of applying the fuzzy technique too to give further details of their differences.

4.3 Mesh Model

The Mesh cross certificate model rests on peer-to-peer agreement between CA couples of the TG. When a new CA asks for acceptance in a new TG, it has to agree with one or more CAs of the group.

In this case there is no external authority that can be used as reference for policy formalization.

Being a peer-to-peer agreement, the CAs have the same role, it is impossible to build the formalization on the basis of only one of the two parts so, in this case, each CA proceeds by building its own REM and evaluation phase. The final results could result in different evaluations.

The adopted evaluation technique should manage flexible and vague judgements and should point out the security level differences, in order to help both the CAs to update their policy and come to an arrangement. In this case, the Fuzzy technique is the best choice.

In this case we have to collect a large set of mutual judgements, which will help to point out how to carry on the final agreement about accepted security levels that will be expressed through a final table similar to the previous one.

This kind of peer-to-peer agreement needs to be always carried on with the help of human experts, the proposed methodology will give support to the decision process, however, it does not express a definitive resolution.

Example. Considering the case of two CAs, CA_A and CA_B, we need to build two REMs and perform mutual evaluations. For brevity sake we will report only one comparison, from the point of view of the CA_B which has 4 different security levels. Using the fuzzy technique, the comparison could be graphically performed as shown in figure 5 where the target policy of CA_A could cooperate with an L2 policy of CA_B by a severe evaluator (orness < 0.3) and it could cooperate with an L3 policy if the evaluator is more indulgent. Analysis of these results are available in [8].

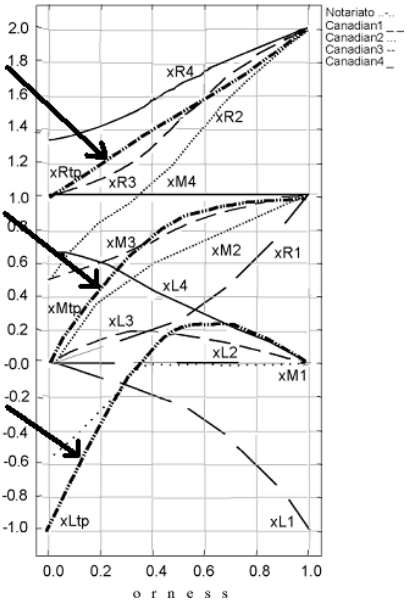

Fig. 5. Evaluation of a policy against Reference Levels

5 Conclusions and Future Work

Cross Certification among CAs is a very huge problem which is actually manually performed by security experts and organizational people, trying to understand if two CAs could cooperate. The evaluation process is based on the evaluation of the Certificate policies which are usually expressed in a not formalized (and native language) way. In this paper we have proposed a methodology to automatically evaluate and compare security policies for Cross Certification. The methodology consists in the formalization of a policy template and in the building of a reference evaluation model, the REM. The core of the REM is the evaluation technique with which we could represent formalized policies and evaluate them against some reference security levels. The application of the methodology in different models of Cross Certification is a numerical value representative of the Global Security Level offered by a CA or just a comparison between two policies when a set of references is not available. We are actually working on the implementation of an automatic evaluator system to apply the methodology in un-trusted domains .

Acknowledgments

This work was partially supported by "Centro Regionale di Competenze" and by the Project: " Simulazione delle prestazioni di architetture per la VoIP operanti in sicurezza, L.R. 5 2002 " Regione Campania.

References

1. Brewer D., Nash M., The Chinese Wall Security Policy, Proceedings of the 1989 IEEE Symposium on Security and Privacy, pp.206-214 (May 1989).
2. Canfora G., Troiano L., "An Extensive Comparison between OWA and OFNWA Aggregation," VIII Sigef Congress, Naples - Italy.2001.
3. Canfora G. and Troiano L., The Importance of Dealing with Uncertainty in the Evaluation of Software Engineering Methods and Tools. SEKE 2002. ACM Press, Ischia, Italy. 691-698 (2002)
4. Canfora G., Cerulo L., Preziosi R., Troiano L., A tool for Decision Support implementing OFNWA approach: a case study. SEKE 2003 .
5. Casola V., Mazzeo A., Mazzocca N., Vittorini V., Policy Formalization to combine separate systems into larger connected networks of trust -Proceedings of NetCon'2002 Conference, Paris, France. 2002.
6. Casola V., Mazzeo A., Mazzocca N., Vittorini V., Policy based interoperability in distributed security infrastructures -Proceedings of 10th ISPE International conference on concurrent engineering: research and applications. Madeira, Spain. 2003.
7. Casola V., Preziosi R., Rak M. , Troiano L. 2004. Security Level Evaluation: Policy and Fuzzy Technique. In IEEE Proceedings of International Conference on Information Technology: Coding and Computing (ITCC 2004), vol. 2, pp. 752-756, Las Vegas, ISBN 0-7695-2108-8.
8. Casola V., Preziosi R., Rak M., Troiano L., A Reference Model for Security Level Evaluation: Policy and Fuzzy Techniques, in JUCS - Journal of Universal Computer Science - edited by Ajith Abraham, Oklahoma State University, USA and L.C. Jain, University of South Australia, January 2005
9. Curry I., Trusted Public-Key Infrastructures, Version 1.2,Entrust Technologies www.entrust.com. 2000.
10. EuroPKI, 2000. Certificate Policy VERSION 1.1 (DRAFT 4), OID: 1.3.6.1.4.1. 5255.1.1.1.
11. M.S. Baum, W. Ford. Secure Electronic Commerce. *Ed. Prentice Hall, 1997.*
12. Grill S. An Approach to Formally Compare and Query Certification Practice Statements, Published on Informatik GI Workshop , Berlin 2000
13. Huitema C., Mendes S., A new approach to the X.509 framework: allowing a global authentication infrastructure without a global trust model, Proceedings of the 1995 Symposium on Network and Distributed System Security (SNDSS'95) 1995.
14. Klobucar T., Jerman-Blazic B., A Formalization and evaluation of certificate policies, Computer Communication 22(1999), 1104-1110
15. Kokolakis S.A., Kiountouzis E.A., Achieving Interoperability in a multiple-security-policies environment , Computer & Security. Vol 19, no. 3 pp 267-281, Elsevier Science 2000.
16. Jajodia S., Samarati P., and Subrahmanian V. S., "A Logical Language for Expressing Authorizations," Published in the proceedings of IEEE Symposium on Security and Privacy, Oakland, USA, 1997.
17. Kagal L., Finin T., Joshi A.,2003. A Policy Language for a Pervasive Computing Environment, IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003)
18. NIST 2001, Report of Federal Bridge Certification Authority Initiative and Demonstration.
19. Polk W. , Hastings N. 2000. Bridge Certification Authorities: Connecting B2B Public Key Infrastructures.

20. RFC2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile. 1999.
21. RFC 3647: Chokhani, S. & Ford, W., 1999. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework.
22. Turnbull J. "Cross-Certification and PKI Policy Networking" Version 1.1, Entrust Technologies www.entrust.com. 2000.
23. Digital Signature and Confidentiality, Certificate Policies for the Government of Canada Public Key Infrastructure, version 3.02 , 1999.
24. Centro Tecnico per la Rete Unitaria, Sezione Sicurezza, 2001. Manuale operativo per il servizio di certificazione di chiavi pubbliche per la rete unitaria della pubblica amministrazione. Versione 1.1

Modeling Public Key Infrastructures in the Real World

John Marchesini and Sean Smith

BindView Corporation and Department of Computer Science, Dartmouth College
john.marchesini@bindview.com
sws@cs.dartmouth.edu

Abstract. PKIs are complex distributed systems that are responsible for giving users enough information to make reasonable trust judgments about one another. Since the currencies of PKI are trust and certificates, users who make trust decisions (often called *relying parties*) must do so using only some initial trust beliefs about the PKI and some pile of certificates (and other assertions) they received from the PKI. Given a certificate, a relying party needs to conclude that the keyholder described by the certificate actually possesses the properties described by the certificate. In this paper, we present a calculus that allows relying parties to make such trust judgements. Our calculus extends Maurer's deterministic model, and is focused on real world issues such as time, revocation, delegation, and heterogeneous certificate formats. We then demonstrate how our calculus can be used to reason about numerous situations that arise in practice.

1 Introduction

PKIs are complex distributed systems that are responsible for giving users enough information to make reasonable trust judgments about one another. While there are a number of metrics we can use to reason about PKIs, one measure stands out: we say a PKI is *correct* if it allows Alice to conclude about Bob what she should, and disallows her from concluding things she should not. PKI designers need tools which can accurately evaluate the correctness of their designs and clearly illustrate what types of trust judgments their systems enable. The literature contains a number of approaches for applying formal methods to the PKI problem (e.g., [11,13,15,20,23]). The modeling work of Ueli Maurer [20] stands out, as it is simple, flexible, and is used to reason about PKI.

We are primarily concerned with designing, building, and deploying PKI systems which allow relying parties to make reasonable trust judgments. We have applied Maurer's calculus to model some of the systems we have seen in the wild as well as systems we have built in the lab. However, the real world is messy. Repeatedly, we find that the calculus cannot model some of the concepts we see in practice. For example:

- Usually, what matters about a public key is not some innate "authenticity" of it, but whether the keyholder has the properties to which the certificate attests.
- Certificates carry more than names; they carry extensions, use policies, attributes, etc. Some types of certificates (e.g., X.509 Attribute Certificates [8]) bind a key to a set of properties, and other types (e.g., SDSI/SPKI [7]) do not require names at all. In many real-world PKI applications, a globally unique name is not even the relevant parameter [5,6].

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 118–134, 2005.

- Certificates and beliefs expire and/or get revoked. Some systems use short certificate lifespans as a security advantage. Systems which use multiple certificates to describe an entity can have lifespan mismatches. For example, an Attribute Certificate that contains the courses Alice is enrolled in this term may expire well before her Identity Certificate.
- Some systems allow users to delegate some or all of their authority to other users.
- Some systems use a combination of multiple certificate types. The Grid community's MyProxy [22] system uses X.509 certificates in conjunction with short-lived Proxy Certificates [24,26] for authentication and dynamic delegation. Greenpass uses an X.509 certificate in conjunction with a SDSI/SPKI certificate to express delegation.
- Many federated PKI systems (such as the Federal Bridge Certification Authority, the Higher Education Bridge Certification Authority, and SAFE) involve multiple entities issuing multiple statements about the trustworthiness of multiple users.

In this paper, rather than start with a calculus and attempt to make all of the PKIs we see fit into the calculus, we start with the things we have seen, and rework Maurer's calculus to allow us to reason about all of them. We begin by reviewing Maurer's calculus in Section 2. Then, we extend the calculus in Section 3 in order to make a more powerful tool for evaluating PKIs. Section 4 uses the extended calculus to reason about a number of real-world PKI scenarios, and relates this work to ideas in *trust management.* Section 5 concludes.

2 Maurer's Calculus

In 1996, Ueli Maurer's seminal "Modelling a Public-Key Infrastructure" [20] presented a deterministic model for PKI. In this model, a relying party Alice can use a certificate issued by Certification Authority (CA) X for user Bob if and only if Alice knows the public key for X and believes that it is *authentic*, and Alice *trusts* X to be honest and to correctly authenticate the owner of a public key before signing it. To determine whether Alice can deduce these facts, the calculus contains four types of *statements* and two *inference rules*. Alice can use her *initial view* (her axioms) and the rules to derive new statements. A *valid statement* is one contained in Alice's *derived view.*

The calculus introduced two concepts which are worth clarifying. First, a *recommendation*, transfers trust. Similar to a certificate, a recommendation grants the power to issue certificates and/or further recommendations. For example, if entity X has issued a recommendation to entity Y, then X is stating that it believes Y is trustworthy enough to issue certificates and further recommendations. Second, a *trust level parameter* limits the length of recommendations and certificate chains. For instance, if Alice trusts X at level 3, then she will accept certificate chains with a maximum length of 3.

Maurer also presents a useful graphical notation for the calculus, but given space constraints, we do not reproduce the details of Maurer's definitions here.

Maurer's deterministic model is appealing because it is simple and flexible. However, when we apply the model to the types of systems we deal with in practice, we discover limits of its applicability.

Authenticity. Maurer's "Authenticity of public keys" is the wrong concept. In practice, we find that a relying party does not care about some innate "authenticity" of a public key, but rather about the binding between the public key and the information in the certificate. Often, the portion of the certificate information that defines the subject's name is not what Alice cares about—she may instead care about key usage policies, constraints, or other extensions.

Time. In real-world PKIs, certificates expire, beliefs expire, and certificates get revoked. Without any concept of time, Maurer's model makes it impossible for relying parties to take such events into consideration when making trust decisions.

Delegation. Sometimes, Bob would like to give another party the right to claim some of the attributes in his certificate. For instance, Bob may want to let Charlie claim to be Bob, so that Charlie can act as Bob. Diane may want to issue a certificate to Frank which indicates he is one of her teaching assistants. Maurer's recommendations are all-or-nothing, meaning that if Alice has issued a recommendation to Bob, then Alice is claiming Bob is trustworthy for the same set of operations that Alice is trustworthy for. In practice, Alice may want to limit what properties she gives to Bob.

Verification. Maurer claims that certificates and recommendations are *allegedly* issued by an entity, because verification is outside the scope of the calculus. However, verification—including the various contending approaches to checking revocation and expiration—is an important (and messy) part of real world PKI [4,10,12,19,21]. For example, assume that a relying party Alice has the following initial view:

$$\mathit{View}_A = \{\mathit{Aut}_{A,X},\ \mathit{Trust}_{A,X,1},\ \mathit{Cert}_{X,Y}\}\ .$$

Even if $\mathit{Cert}_{X,Y}$ is invalid for some reason (e.g., revocation, expiration, usage violation), Alice can still derive the authenticity of Y's public key:

$$\mathit{Aut}_{A,X},\ \mathit{Trust}_{A,X,1},\ \mathit{Cert}_{X,Y} \vdash \mathit{Aut}_{A,Y}$$

Thus Alice draws an incorrect conclusion.

3 A Model for the Real World

Our revised model is rooted in Maurer's deterministic model, but extends it in order to deal with the complexity of real-world PKIs. From a high level, our extensions involve several elements.

1. We generalize Maurer's *Authenticity of public keys* to capture the notion of the authenticity of the binding between a public key and the certificate information.
2. We add the concept of time to Maurer's calculus so that we can model expiration and revocation.
3. We replace Maurer's *Recommendation* with a *Trust Transfer* which allows an entity A to give entity B the right to claim some or all of A's certificate information. This replacement allows us to remove the non-intuitive trust level parameter from the calculus, and to explicitly handle the various forms of trust transfer that occur in real-world PKI.

4. We introduce the notion of *validity templates* which are used to capture format-specific definitions of a statement's validity.
5. We redefine the inference rules to utilize these extensions.

(We also change the notation to use postfix instead of subscripts for the arguments, to improve readability.)

3.1 The Model

Informally, we use two concepts to make Maurer's deterministic model time-aware. The *lifespan* of a statement s is the time interval from t_j to t_k on which s can be used in trust calculations. We denote lifespans as the interval $\mathcal{I}$ where $\mathcal{I}$ is the time interval $[t_j, t_k]$. At time $t > t_k$, we say that s has *expired* and is no longer usable in trust calculations. We say statement s is *active* at time t if and only if $t \in \mathcal{I}$, the lifespan of s. (In theory, we could also add two levels of time: the time period during which the assertion is true, and the time period during which a party may believe and use this assertion. However, we found the simpler approach sufficed.)

We use the concept of a *domain* to indicate the set of properties that a certificate issuing entity may assign to its subjects. Intuitively, the domain of an entity is what it is allowed to vouch for. For example, the Dartmouth College CA can bind names, Dartmouth-specific attributes, and other extensions to public keys. Thus, the CA's domain (denoted as the set $\mathcal{D}$) is the set of names, Dartmouth-specific attributes, and extensions it can bind to public keys. The Dartmouth CA cannot bind Department of Defense (DoD)-specific attributes to public keys because it is not authorized to vouch for the DoD—i.e., the DoD-specific attributes are not in $\mathcal{D}$.

With these three concepts, we can formally define our model with the following two definitions.

Definition 1. In our model, statements and their representations are one of the following forms:

- **Authenticity of binding.** *Aut*$(A, X, \mathcal{P}, \mathcal{I})$ denotes A's belief that, during the interval $\mathcal{I}$, entity X (i.e., the entity holding the private key K_X) has the properties defined by the set $\mathcal{P}$.

  The symbol is an edge from A to X labeled with $\mathcal{P}, \mathcal{I}$: $A \xrightarrow{\mathcal{P},\mathcal{I}} X$.
- **Trust.** *Trust*$(A, X, \mathcal{D}, \mathcal{I})$ denotes A's belief that, during the interval $\mathcal{I}$, entity X is trustworthy for issuing certificates over domain $\mathcal{D}$.

  The symbol is a dashed edge from A to X labeled $\mathcal{D}, \mathcal{I}$: $A \overset{\mathcal{D},\mathcal{I}}{\dashrightarrow} X$.
- **Certificates.** *Cert*$(X, Y, \mathcal{P}, \mathcal{I})$ denotes the fact that X has issued a certificate to Y which, during the interval $\mathcal{I}$, binds Y's public key to the set of properties $\mathcal{P}$.

  The symbol is an edge from X to Y labeled with $\mathcal{P}, \mathcal{I}$: $X \xrightarrow{\mathcal{P},\mathcal{I}} Y$.
- **Trust Transfers.** *Tran*$(X, Y, \mathcal{P}, \mathcal{I})$ denotes that A holds a trust transfer issued by X which, during the interval $\mathcal{I}$, binds Y's public key to the set of properties $\mathcal{P}$.

  The symbol is a dashed edge from X to Y labeled with $\mathcal{P}, \mathcal{I}$: $X \overset{\mathcal{P},\mathcal{I}}{\dashrightarrow} Y$.

- **Certificate Validity Templates.** *Valid*$\langle A, C, t\rangle$ denotes A's belief that certificate C is valid at evaluation time t according to the definition of validity appropriate for C's format. Minimally, the issuer's signature over C must be verified and C must be active.
- **Transfer Validity Templates.** *Valid*$\langle A, T, t\rangle$ denotes A's belief that trust transfer T is valid at evaluation time t according to the definition of validity appropriate for T's format. Minimally, the issuer's signature over T must be verified and T must be active.

As with Maurer's model, some of the symbols are identical because of the similarity in meaning. Authenticity can be thought of as a certificate signed by one's own private key; trust can be thought of as a recommendation signed by one's own private key.

We introduce the notion of *template* because different PKI approaches have different (and non-trivial) ways of expressing validity of certificates and transfers. For example, validity for X.509 identity certificates may be determined by expiration dates and the absence of the certificate on a currently valid *certificate revocation list (CRL)*; validity of transfer in an X.509 identity certificate may be determined by basic constraints and usage bits in a certificate held by the source party.

Alice's initial view is denoted *View*$_A$, as in Maurer's deterministic model. Under our model, if Alice wishes to verify that Bob had some property p at time t, she must be able to derive the statement *Aut*$(A, B, \mathcal{P}, \mathcal{I})$ where $t \in \mathcal{I}$ and $p \in \mathcal{P}$. In many cases, the evaluation time t is the current time, meaning that Alice wants to verify that Bob currently has some property p. It should be noted, however, that the model still functions if the evaluation time t is some time in the past or the future. Such scenarios will be examined more closely in Section 4.

Definition 2. In our model, a statement is valid if and only if it is either contained in *View*$_A$ or if it can be derived from *View*$_A$ by applications of the following inference rules:

$$\forall X, Y, t \in \{\mathcal{I}_0 \cap \mathcal{I}_1\}, \mathcal{Q} \subseteq \mathcal{D}: \tag{1}$$

$$\textit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \textit{Trust}(A, X, \mathcal{D}, \mathcal{I}_1),\ \textit{Valid}\langle A, \textit{Cert}(X, Y, \mathcal{Q}, \mathcal{I}_2), t\rangle \vdash \textit{Aut}(A, Y, \mathcal{Q}, \mathcal{I}_2)$$

$$\forall X, Y, t \in \{\mathcal{I}_0 \cap \mathcal{I}_1\}, \mathcal{Q} \subseteq \mathcal{D}: \tag{2}$$

$$\textit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \textit{Trust}(A, X, \mathcal{D}, \mathcal{I}_1),\ \textit{Valid}\langle A, \textit{Tran}(X, Y, \mathcal{Q}, \mathcal{I}_2), t\rangle \vdash \textit{Trust}(A, Y, \mathcal{Q}, \mathcal{I}_2)$$

As with Maurer's deterministic model, for a finite set S of statements, $\overline{S}$ denotes the closure of S under applications of the inference rules (1) and (2), i.e., the set of statements derivable from S. The *evaluation time* t is the time that Alice is attempting to reason about. Alice's *derived view at evaluation time* t is the set of statements derivable from her initial view at evaluation time t. Alice's derived view is defined by the function $\overline{\textit{View}_A(t)}$ where $\overline{\textit{View}_A} : t \longrightarrow \overline{S}$. Under the model, a statement s is *valid at evaluation time* t if and only if $s \in \overline{\textit{View}_A(t)}$, and invalid otherwise.

3.2 Semantic Sugar

We explain the intuition for the definitions of the model and highlight some of the semantic difference between our model and Maurer's.

Maurer's notion of "authenticity" establishes that entity X holds the private key corresponding to the public key in X's certificate. We extend authenticity to establish that some entity X not only holds the private key corresponding to the public key in X's certificate, but also has the other properties $\mathcal{P}$, such as attributes, roles, key attributes, extended key usage, etc.

With "level," Maurer limits trust vertically (i.e., how deep trust may propagate). With "domain," we limit trust horizontally (i.e., how wide the trust may span). A trusted entity should only be allowed to vouch (either via a certificates or trust transfer statement) for properties that it is authorized to speak for. Entities may be allowed to vouch for a specific domain for a number of reasons. In many cases, the assignment of a domain to a trusted entity is done out-of-band of the PKI, and the users trust it a priori. For example, users almost always trust their CA to vouch for the organization's population. In our calculus, this fact is represented by the inclusion of the CA's authenticity and trust statements in every user's initial view. In other cases, the assignment of a domain to a trusted entity is done implicitly. Delegation scenarios are an example of this type of binding. Typically, if Alice trusts Bob to delegate some of his privileges to another entity, Alice would require Bob to have had the privilege in the first place. In the model, this is represented by Bob's trust statement having a subset of the properties in his authenticity statement, i.e. for $\mathcal{Q} \subseteq \mathcal{P}$:

$$\mathit{View}_A = \{\mathit{Aut}(A, B, \mathcal{P}, \mathcal{I}),\ \mathit{Trust}(A, B, \mathcal{Q}, \mathcal{I})\}\ .$$

Generalizing Maurer's "recommendation," our trust transfer statement can be used to model different types of transactions such as when a CA certifies a subordinate CA, or when Alice delegates some or all of her properties to Bob. Moreover, a trust transfer may be an explicit statement (such as a certificate) or an implicit statement (e.g., by activating a certificate extension such as the X.509 "basicConstraints" extension).

As discussed in Section 2, Maurer's calculus does not include checking for certificate validity as part of the calculus. Our model deals with this is through validity templates: a meta-statements whose validity checking algorithm depends on the argument type. Templates allow us to reason about different certificate formats without having to handle every format's specifics. For example, assume that $\mathit{Valid}\langle A, C, t\rangle$ is being evaluated, and C is an X.509 Identity Certificate. In order for $\mathit{Valid}\langle A, C, t\rangle$ to be true, the template instantiation should check that C's signature verifies, that C has not expired, that C has not been revoked (e.g., by having in one's belief set a properly signed, active copy of the Certificate Revocation List (CRL) to which C points), that C's key attributes allow the requested operation, that the certificate chain length has not been exceeded, etc. If C were an X.509 Attribute Certificate, $\mathit{Valid}\langle A, C, t\rangle$ may also check that C was signed by an attribute authority. Evaluating trust transfer statements is similar.

Note that the level parameter of Maurer's deterministic model has been omitted in our model. The use of validity templates allows relying parties to directly check the properties in certificates and trust transfer statements for things like certificate chain length, delegation depth, "pathLenConstraint", etc. This way, relying parties may draw

such conclusions using the context of the certificate format rather than an artificial level parameter.

3.3 An Example

To illustrate the basic concepts of our model, we extend an example from Section 3 in Maurer's paper.

Consider the PKI depicted in Figure 1. The figure indicates that Alice (A) believes that X's public key is bound to the set of properties $\mathcal{P}$ during the interval $\mathcal{I}_0$ (depicted by the solid edge from A to X). She trusts X to issue certificates over domain $\mathcal{D}_0$ (the dashed edge). (For simplicity, we set the lifespans of the trust statements to match the authenticity and certificate statements, but this is not necessary.) Her view contains a trust transfer from X to Y (the dashed edge), and two certificates (solid edges): one from X to Y which binds Y's public key to the set of properties $\mathcal{Q}$ during the interval $\mathcal{I}_1$, and one from Y to B which binds B's public key to the set of properties $\mathcal{R}$ during the interval $\mathcal{I}_2$. The trust transfer from X to Y could be an explicit statement issued by X indicating that it trusts Y to issue certificates; in practice, it is more likely to be expressed implicitly in the certificate issued from X to Y (e.g., by X setting the "basicConstraints" field of Y's X.509 certificate or the "delegation" flag of Y's SDSI/SPKI certificate).

In order for Alice to be able to believe Bob's (B) certificate (either the public key or the properties in $\mathcal{R}$) at evaluation time t, she needs to derive the statement $\mathit{Aut}(A, B, \mathcal{R}, \mathcal{I}_2)$. In this scenario, Alice's initial view is the following set of statements:

$$\mathit{View}_A = \left\{ \begin{array}{ll} \mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0), & \mathit{Trust}(A, X, \mathcal{D}_0, \mathcal{I}_0), \mathit{Tran}(X, Y, \mathcal{D}_1, \mathcal{I}_1), \\ \mathit{Cert}(X, Y, \mathcal{Q}, \mathcal{I}_1), & \mathit{Cert}(Y, B, \mathcal{R}, \mathcal{I}_2) \end{array} \right\} .$$

Note that Alice's view does not contain any validity templates. Since validity templates take an evaluation time as input, they are not instantiated until evaluation time t. Now, since Alice does not trust Y to issue certificates directly, she must derive her trust in Y using rule (2). Suppose that evaluation time $t \in \mathcal{I}_0$ and $\mathcal{D}_1 \subseteq \mathcal{D}_0$, we have:

$$\mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, X, \mathcal{D}_0, \mathcal{I}_0),\ \mathit{Valid}\langle A, \mathit{Tran}(X, Y, \mathcal{D}_1, \mathcal{I}_1), t\rangle \vdash \mathit{Trust}(A, Y, \mathcal{D}_1, \mathcal{I}_1) .$$

Once Alice trusts Y, she can use rule (1) to establish the authenticity of the binding expressed in Bob's certificate at evaluation time t assuming $t \in \mathcal{I}_1$, $\mathcal{Q} \subseteq \mathcal{D}_0$, and $\mathcal{R} \subseteq \mathcal{D}_1$ (in addition to our previous supposition that $t \in \mathcal{I}_0$, and $\mathcal{D}_1 \subseteq \mathcal{D}_0$):

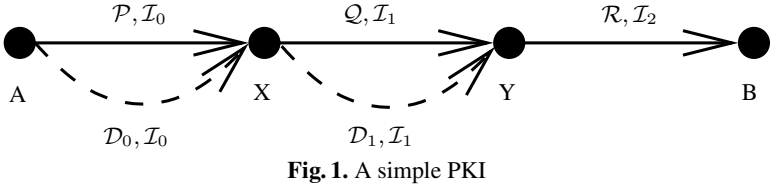

Fig. 1. A simple PKI

$$\mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, X, \mathcal{D}_0, \mathcal{I}_0),\ \mathit{Valid}\langle A, \mathit{Cert}(X, Y, \mathcal{Q}, \mathcal{I}_1), t\rangle \vdash \mathit{Aut}(A, Y, \mathcal{Q}, \mathcal{I}_1)$$

$$\mathit{Aut}(A, Y, \mathcal{Q}, \mathcal{I}_1),\ \mathit{Trust}(A, Y, \mathcal{D}_1, \mathcal{I}_1),\ \mathit{Valid}\langle A, \mathit{Cert}(Y, B, \mathcal{R}, \mathcal{I}_2), t\rangle \vdash \mathit{Aut}(A, B, \mathcal{R}, \mathcal{I}_2)\ .$$

Thus, Alice's derived view at evaluation time t is given by:

$$\overline{\mathit{View}_A(t)} = \mathit{View}_A \cup \{\mathit{Trust}(A, Y, \mathcal{D}_1, \mathcal{I}_1),\ \mathit{Aut}(A, Y, \mathcal{Q}, \mathcal{I}_1),\ \mathit{Aut}(A, B, \mathcal{R}, \mathcal{I}_2)\}\ .$$

Alice believes that the binding between Bob's public key and his certificate properties is authentic during the time interval $\mathcal{I}_2$. Alice may stop believing this fact when Bob's certificate expires or gets revoked.

4 Using This New Model

Our motivation is to give PKI designers a tool which can be used to reason about a wide range of PKI systems. In this section, we apply the model to an array of real-world situations in order to illustrate its applicability.

4.1 Modeling Multiple Certificate Families

The new model's certificate statement binds an entity's public key to some set of properties $\mathcal{P}$ for some lifespan $\mathcal{I}$. The power of the new model stems from the fact that it is agnostic with respect to the semantics of the properties in $\mathcal{P}$, and yet still builds a calculus which allows relying parties to reason about these sets.

In standard X.509 Identity Certificates [10], the property sets may include the subject's Distinguished Name, Alternative names, name constraints, her key attributes, information about where to retrieve CRLs, and any number of domain-specific policies. The property set may also include information as to whether the subject is allowed to sign other certificates (i.e., via the "basicConstraints" field).

X.509 Attribute Certificates (ACs) [8] contain a very different set of properties than X.509 Identity Certificates. ACs typically use domain-specific properties which are used by relying parties to make authorization decisions. Some common examples of attributes include: identity, group membership, role, clearance level, etc. Other differences include the fact that an AC's subject may delegate to another party the right to claim some of the delegator's attributes.

An X.509-based Proxy Certificate (PC) [24] is similar to an X.509 Identity Certificate, except that PCs have a Proxy Certificate Information (PCI) extension and are signed by standard X.509 Identity Certificates. The PC standard allows any type of policy statement expressed in any language (such as eXtensible Access Control Markup Language) to be placed in the PCI. Thus, the set of properties for a PC could contain a large family of policy statements.

The SDSI/SPKI certificate format [7] takes an entirely different approach to certificates. The set of properties placed in a SDSI/SPKI certificate does not contain a global

name for the subject, as SDSI/SPKI uses the public key as the subject's unique identifier. (If there is any name at all, it would be part of a linked local namespace.) Further, a SDSI/SPKI certificate contains attributes much like X.509 attributes, except they are expressed as S-expressions as opposed to ASN.1. In contrast to X.509 ACs, SDSI/SPKI certificates are allowed to be chained.

Our new model enables reasoning about all of these diverse certificate formats and semantics within one calculus. The addition of properties to the calculus allows relying parties to reason about different types of information contained in the different certificate families.

4.2 Modeling Revocation

Validity templates play an important role in our model: they allow users to reason about different *types* of signed statements. As an example, consider the case when Alice needs to make a trust decision about Bob. The instantiation of the validity template used to check Bob's certificate may require that Alice check a CRL to ensure that Bob's certificate is not included in the list of revoked certificates.

Thus, Alice's first step is to make a trust decision about a signed CRL. CRLs contain a list of revoked certificates, a lifespan (noted by the "thisUpdate" and "nextUpdate" fields), and are signed by the organization's CA. Formally, we can represent a CRL as a kind of certificate which is issued by a CA X and contains a list of revoked certificates $\mathcal{L}$, and a lifespan $\mathcal{I}$: $\mathit{Cert}(X, \emptyset, \mathcal{L}, \mathcal{I})$. Since CRLs do not contain a public key, we use the empty set notation to indicate the absence of a key. In order for Alice to use the CRL at evaluation time t, she needs to deduce that it is authentic, i.e., $\mathit{Aut}(A, \emptyset, \mathcal{L}, \mathcal{I}) \in \overline{\mathit{View}_A(t)}$.

Assuming that Alice believes that the binding expressed in CA X's certificate is authentic, and that she trusts CA X to issue certificates over domain $\mathcal{D}$, her initial view would be:

$$\mathit{View}_A = \{\mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, X, \mathcal{D}, \mathcal{I}_0),\ \mathit{Cert}(X, \emptyset, \mathcal{L}, \mathcal{I}_1)\}\ .$$

If X is authorized to vouch for the revocation status of all the certificates in $\mathcal{L}$ (i.e., $\mathcal{L} \subseteq \mathcal{D}$), and all of the statements are active (i.e., $t \in \mathcal{I}_0$), then Alice can deduce the CRL's authenticity by applying rule (1):

$$\mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, X, \mathcal{D}, \mathcal{I}_0),\ \mathit{Valid}\langle A, \mathit{Cert}(X, \emptyset, \mathcal{L}, \mathcal{I}_1), t\rangle \vdash \mathit{Aut}(A, \emptyset, \mathcal{L}, \mathcal{I}_1)\ .$$

For the CRL, the instantiation of the validity template $\mathit{Valid}\langle A, \mathit{Cert}(X, \emptyset, \mathcal{L}, \mathcal{I}_1), t\rangle$ must check that $t \in \mathcal{I}_1$, and that X's signature is verifiable. If the conditions are met, we have $\mathit{Aut}(A, \emptyset, \mathcal{L}, \mathcal{I}_1) \in \overline{\mathit{View}_A(t)}$, which indicates Alice's belief that $\mathcal{L}$ accurately represents the list of revoked certificates during the interval $\mathcal{I}_1$.

Once Alice believes the CRL, she must make a trust decision about Bob's certificate: $\mathit{Cert}(X, B, \mathcal{Q}, \mathcal{I}_2)$. Assuming that CA X can vouch for Bob's certificate information (i.e., $\mathcal{Q} \subseteq \mathcal{D}$), and all of the statements are active (i.e., $t \in \mathcal{I}_0$), then Alice can deduce Bob's authenticity by applying rule (1):

$$\mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, X, \mathcal{D}, \mathcal{I}_0),\ \mathit{Valid}\langle A, \mathit{Cert}(X, B, \mathcal{Q}, \mathcal{I}_2), t\rangle \vdash \mathit{Aut}(A, B, \mathcal{Q}, \mathcal{I}_2)\ .$$

In this case, the instantiation of the validity template $\mathit{Valid}\langle A, \mathit{Cert}(X, B, \mathcal{Q}, \mathcal{I}_2), t\rangle$ is being used to establish the validity of a certificate, not a CRL. As before, the template instantiation must check that $t \in \mathcal{I}_2$, and that X's signature over Bob's certificate verifies. However, in this case, the instantiation should also check that Bob's certificate has not been revoked, i.e., $\mathit{Cert}(X, B, \mathcal{Q}, \mathcal{I}_2) \notin \mathcal{L}$ as well as any other certificate information which is relevant to the requested operation.

4.3 Authorization-Based Scenarios and Trust Management

In many modern distributed systems, access to some resource is granted based on authorization rather that authentication. Systems such as PERMIS [3] use the attributes contained in ACs to determine whether an entity should have access to a resource (other Trust Management systems such as KeyNote [1,2] and PolicyMaker [17,18] have their own certificate formats for expressing credentials). This approach simplifies the management of ACLs at the resource. For example, if Bob wants to access Alice's file, he presents his AC to Alice. Alice first decides if the AC (or set of credentials) is authentic, and if so, she examines Bob's attributes to check if he should have access (e.g., if the file is accessible to the group "developers", then Bob's attributes must state that he is a member of the group).

Maurer's deterministic model cannot handle this scenario, primarily because it cannot handle ACs. Under the deterministic model, if Alice were to deduce the authenticity of Bob's public key, she still has learned nothing about Bob (i.e., his attributes). All she has established is that the entity named Bob really has the private key corresponding to the public key found in the certificate.

There are a number of *Trust Management (TM)* languages which do handle this scenario, such as *Delegation Logic* [14] and others [16]. These TM languages can not only tell Alice that Bob has a certain set of credentials, but can also evaluate Bob's credentials and Alice's policy to determine whether Alice should allow the file access. While TM languages are typically framework-specific (i.e., KeyNote, PolicyMaker, and SDSI/SPKI have their own policy languages), there have been efforts to generalize across languages [25]. Since our model is aimed at reasoning about PKI systems, and not TM systems, such policy evaluation is outside the scope of our model's abilities. However, our model can be used to model these different certificate and credential formats (e.g., ACs, credentials, SDSI/SPKI certificates), as well as reason about the authenticity of the core trust statements.

Under our model, Bob would first present his AC to Alice (e.g., $\mathit{Cert}(X, B, \mathcal{P}, \mathcal{I})$). Assume that Alice can then derive the authenticity of the binding between Bob's public key and the properties in the certificate—i.e., $\mathit{Aut}(A, B, \mathcal{P}, \mathcal{I}) \in \overline{\mathit{View}_A(t)}$. Since Bob's certificate is an AC, Alice needs to determine if an attribute placing Bob in the "developers" group is in the set $\mathcal{P}$. If so, then Bob is allowed to access the file.

4.4 Delegation

Some systems allow users to delegate some or all of their properties to another entity. Maurer's deterministic model allows users to issue recommendations and certificates to

other entities, but this is insufficient to capture the notion of delegation. Maurer's model allows Alice to vouch for Bob, but she is limited to vouching for Bob's identity.

In our model, Alice can give some or all of her properties to Bob (possibly including identity), provided she has the properties in the first place (i.e., the she can only give Bob the properties in her domain). In the calculus, Alice would issue a certificate to Bob (i.e., $\mathit{Cert}(A, B, \mathcal{P}, \mathcal{I})$).

If a relying party Charlie has established that the binding between Alice's public key and her properties is authentic, trusts Alice to delegate, and receives a delegation from Alice to Bob, then his view will be:

$$\mathit{View}_C = \{\mathit{Aut}(C, A, \mathcal{P}, \mathcal{I}),\ \mathit{Trust}(C, A, \mathcal{P}, \mathcal{I}),\ \mathit{Cert}(A, B, \mathcal{P}, \mathcal{I})\}\ .$$

He can then derive the authenticity of the binding between Bob's public key and the delegated properties (i.e., the statement $\mathit{Aut}(A, B, \mathcal{P}, \mathcal{I})$) by applying rule (1):

$$\mathit{Aut}(C, A, \mathcal{P}, \mathcal{I}),\ \mathit{Trust}(C, A, \mathcal{P}, \mathcal{I}),\ \mathit{Valid}\langle C, \mathit{Cert}(A, B, \mathcal{P}, \mathcal{I}), t\rangle \vdash \mathit{Aut}(C, B, \mathcal{P}, \mathcal{I})\ .$$

Thus, we have $\mathit{Aut}(C, B, \mathcal{P}, \mathcal{I}) \in \overline{\mathit{View}_C(t)}$.

4.5 Modeling MyProxy

The Grid community's MyProxy credential repository [22] uses a chain of certificates for authentication. When Bob (or some process to which Bob delegates) wants to access a resource on the Grid, he generates a temporary keypair, logs on to the MyProxy server, and requests that a Proxy Certificate (PC) [24,26] be generated which contains the public portion of the temporary keypair and some subset of Bob's privileges. The new PC is then signed with the private portion of the keypair described by Bob's long term X.509 Identity Certificate, thus forming a chain of certificates.

As Figure 2 shows, entity X is the CA which issued Bob's X.509 Identity Certificate, and T is the entity which will own the temporary keypair (possibly Bob or some other delegated entity or process). Initially, Alice believes that the binding between X's public key and properties is authentic during $\mathcal{I}_0$, and she trusts X to issue certificates and trust transfers for the domain $\mathcal{D}$. X has issued Bob a certificate binding his public key to the set of properties $\mathcal{Q}$ during $\mathcal{I}_1$. X has also issued a trust transfer to Bob, so that he may use his private key to sign his PC. The trust transfer is not a separate certificate in this scenario; it is an implicit statement which X makes by issuing an identity certificate to Bob. In practice, the relying party's validation software must be modified to accept a standard identity certificate as a trust transfer statement from the CA to the

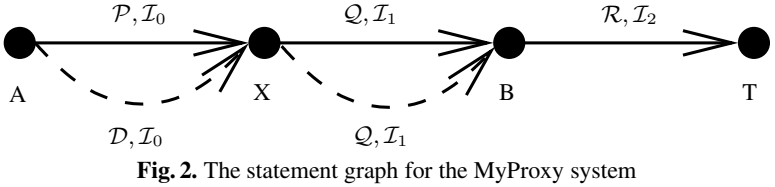

Fig. 2. The statement graph for the MyProxy system

subject. Finally, Bob has issued a certificate (the PC) to the entity possessing the temporary keypair T for some subset $\mathcal{R}$ of his properties. The PC is valid over the interval $\mathcal{I}_2$, which in practice, is on the order of eight hours. In order for Alice to accept Bob's PC, she must derive the statement $\mathit{Aut}(A, T, \mathcal{R}, \mathcal{I}_2)$. This scenario can be reduced to the example discussed in Section 3.3.

4.6 Discovering Requirements: Greenpass

The Greenpass [9] system uses delegation to give guests inside access to a campus-wide wireless network. Further, it relies on an X.509 certificate in conjunction with SDSI/SPKI certificates to express delegation. To gain some insight as to why the designers chose this hybrid approach, we can model the problem with the calculus.

Let us assume that a relying party Alice is a member of that college, which we denote C. Let us also assume that another member of C, named Bob, has invited his colleague George from the University of Wisconsin (denoted W) to come for a visit. Bob would like to give George some guest access to the network, so that he can access some resources protected by Alice. In order for Alice to grant access to George, she must make a trust decision about George. Since there is no trust relationship between C and W (i.e., they are not cross-certified or participating in the Higher Education Bridge CA), Alice cannot simply reason about George based on statements made by George's CA. Since George is Bob's guest, Bob is in a position to vouch for George.

So, initially, Alice's view consists of her authenticity and trust beliefs about her CA, a certificate issued by her CA to Bob, and a certificate issued by George's CA to George:

$$\mathit{View}_A = \{\mathit{Aut}(A, C, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, C, \mathcal{D}, \mathcal{I}_0),\ \mathit{Cert}(C, B, \mathcal{Q}, \mathcal{I}_1), \mathit{Cert}(W, G, \mathcal{R}, \mathcal{I}_2)\}\ .$$

Since Bob has a certificate issued by a CA which Alice trusts, she can deduce the authenticity of Bob's certificate information (assuming that $t \in \mathcal{I}_0$, $\mathcal{Q} \subseteq \mathcal{D}$, and Bob's certificate is valid), i.e.,

$$\mathit{Aut}(A, C, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, C, \mathcal{D}, \mathcal{I}_0),\ \mathit{Valid}\langle A, \mathit{Cert}(C, B, \mathcal{Q}, \mathcal{I}_1), t\rangle \vdash \mathit{Aut}(A, B, \mathcal{Q}, \mathcal{I}_1)\ .$$

Now, in order for Alice to grant George access to her resources, she needs to believe the binding between George's public key and the properties in his certificate, and then that the properties grant him authorization. However, since Alice does not trust W (and has no reason to), she has no reason to trust any of the properties about George expressed in his certificate (namely, in the set of properties $\mathcal{R}$).

Since George is Bob's guest, Bob is in a position to delegate some of his privileges to George. In order for Alice to believe this delegation, Alice first needs to believe that Bob is in a position to delegate, and she then needs to believe that Bob actually delegated to George. The first condition requires the CA to transfer trust to Bob (i.e.,

$\mathit{Tran}(C, B, \mathcal{Q}, \mathcal{I}_1) \in \mathit{View}_A$).[1] The second condition requires that Bob issue a certificate which delegates some of his properties to George (i.e., $\mathit{Cert}(B, G, \mathcal{S}, \mathcal{I}_2) \in \mathit{View}_A$ where $\mathcal{S} \subseteq \mathcal{Q}$).

Assuming all of the preconditions are met, and the certificates and trust transfer are valid, Alice can deduce Bob's trustworthiness, and the authenticity of the certificate issued from Bob to George, i.e.,

$$\mathit{Aut}(A, C, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, C, \mathcal{D}, \mathcal{I}_0),\ \mathit{Valid}\langle A, \mathit{Tran}(C, B, \mathcal{Q}, \mathcal{I}_1), t\rangle \vdash \mathit{Trust}(A, B, \mathcal{Q}, \mathcal{I}_1)$$

$$\mathit{Aut}(A, B, \mathcal{Q}, \mathcal{I}_1),\ \mathit{Trust}(A, B, \mathcal{Q}, \mathcal{I}_1),\ \mathit{Valid}\langle A, \mathit{Cert}(B, G, \mathcal{S}, \mathcal{I}_3), t\rangle \vdash \mathit{Aut}(A, G, \mathcal{S}, \mathcal{I}_3)\ .$$

Thus Alice can reason about George because $\mathit{Aut}(A, G, \mathcal{S}, \mathcal{I}_3) \in \overline{\mathit{View}_A(t)}$.

The last question that the system designer is faced with is: "what type of certificate format should be used for the certificate issued from Bob to George?" The first consideration is that if George already has a public key, the system should reuse it. The second consideration is that Alice is not concerned with George's identity, but rather his authorization. Finally, we need to reason about what type of certificate format would allow this type of delegation scenario, and still make $\mathit{Valid}\langle A, C, t\rangle$ evaluate to true. Proxy Certificates would not allow George to have the public key of his regular certificate (i.e., $\mathit{Cert}(W, G, \mathcal{R}, \mathcal{I}_3)$) also used in his Proxy Certificate, resulting in $\mathit{Valid}\langle A, C, t\rangle$ never being true. An X.509 Attribute Certificate would not make $\mathit{Valid}\langle A, C, t\rangle$ true unless Bob was an Attribute Authority proper. This leaves us with the choice to either make all entities attribute authorities as well, or use SDSI/SPKI certificates (which is what Greenpass implemented).

4.7 Time Travel

There may be times when a relying party would like to reason about an event that has past or one that has not happened yet (because some statements are not yet active). Maurer's model lacks of the concept of time. In the new model, we can reason about such events by manipulating the evaluation time t.

For example, assume that a relying party Alice is trying to verify a signature that Bob generated on April 21, 1984. (Note that Alice would need a mechanism such as a timestamping service in order to know that the signature existed on April 21, 1984). Further, assume that Alice believed that the CA X had an authentic binding between its public key and certificate information, and that it trusted the CA during that time period. Last, Alice would have to possess a certificate for Bob's which was valid during that period.

More formally, let $\mathcal{I}_0$ be the time period from January 1, 1984 to December 31, 1984. Let $\mathcal{I}_1$ be the time period from April 1, 1984 to April 30, 1984. Finally, let $\mathcal{Q} \subseteq \mathcal{D}$. Alice's initial view is given by:

[1] In the Greenpass prototype, this trust transfer is expressed as a SDSI/SPKI certificate issued to Bob's public key and allowing him to delegate. It could also have been implicit, by setting the "basicConstraints" field of Bob's X.509 certificate, but this would require reissuing Bob's certificate.

$$\mathit{View}_A = \{\mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, X, \mathcal{D}, \mathcal{I}_0),\ \mathit{Cert}(X, B, \mathcal{Q}, \mathcal{I}_1)\}\ .$$

Now, at evaluation time t where $t \in \{\mathcal{I}_0 \cap \mathcal{I}_1\}$ (i.e., t is some time in April, 1984), Alice can use rule (1) to derive the authenticity of the binding between Bob's public key and his certificate information:

$$\mathit{Aut}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Trust}(A, X, \mathcal{P}, \mathcal{I}_0),\ \mathit{Valid}\langle A, \mathit{Cert}(X, B, \mathcal{Q}, \mathcal{I}_1), t\rangle\ \vdash \mathit{Aut}(A, B, \mathcal{Q}, \mathcal{I}_1)\ .$$

Thus, Alice can use Bob's public key to verify the signature (she could also use any other of Bob's properties in the set $\mathcal{Q}$) because $\mathit{Aut}(A, B, \mathcal{Q}, \mathcal{I}_1) \in \overline{\mathit{View}_A(t)}$ when t is some time in April, 1984. If she were to try and derive the same statement in May of 1984, she would fail because Bob's certificate expired after April, 1984. Since the evaluation time t would be in May, 1984, we have $t \notin \{\overline{\mathcal{I}_0 \cap \mathcal{I}_1}\}$, and the validity template instantiation would fail. Thus, $\mathit{Aut}(A, B, \mathcal{Q}, \mathcal{I}_1) \notin \overline{\mathit{View}_A(t)}$.

4.8 Comparison

Table 1 shows how our model and Maurer's model handle the systems discussed in this section. Since Maurer's model relies on the use of names instead of properties, his model cannot be used to reason about certificate formats which do not use names (such as SDSI/SPKI and X.509 Attribute Certificates). With no notion of time, Maurer's model cannot handle revocation, time travel, and the MyProxy system which relies on short-lived Proxy Certificates. Finally, our concept of domain permits delegation scenarios where a subset of the delegator's privileges are given to the delegatee. This level of granularity is necessary for most real world systems, such as MyProxy and Greenpass.

Table 1. A comparison of the Maurer's model and ours

PKI System	Maurer's model	Our model	Enabling feature
Multiple formats	no	yes	properties
Revocation	no	yes	time
Authorization	no	yes	properties
Delegation	some	yes	domains
MyProxy	some	yes	time, domains
Greenpass	no	yes	properties, domains
Time travel	no	yes	time

5 Conclusions and Future Work

While our new model makes it possible to reason about a number of different types of PKIs and has been useful in practice, it is not perfect. There are a number of interesting potential future directions.

First, our model does not describe how well the properties in the certificates match the real world properties of the certificate's subject. A similar issue arises in the field of program verification. One might determine how well the program fits the specification. However, this does not answer the question "Is my specification any good?" Such approaches yield a program which is at most as correct as the specification. In determining authenticity of a binding between a set of properties and a public key, the relying party trusts the attributes at most as much as it trusts the issuer. If an issuer is careless (or malicious), and binds false properties to Bob's public key then, under the new model (and in the real world), Alice will accept false properties about Bob. As an alternative view, we might consider whether the building blocks of a particular certificate scheme are in fact sufficiently *articulate* for relying parties to make the correct decision, or consider the size of the fraction of the space where relying parties make the wrong decisions. Further investigation into this issue, perhaps including automated formal methods, is an area for future work.

Second, the inclusion of time in the new model makes it *nonmonotonic*: true statements can become false over time. Nonmonotonicity can have a fatal side effect: a relying party may deduce authenticity when it should not. Some statement may have expired or been revoked, and the relying party has not received the revocation information yet. Li and Feigenbaum [15] introduce a concept of "fresh time" which could be used either in the certificate's properties, or possibly as an explicit parameter to make the system monotonic. Using fresh times in our model is another area for future work.

Last, certification and trust transfer statements in our new model are similar to Jon Howell's "speaks-for-regarding" operator [11]. However, our statements go beyond Howell's because they are applicable to a number of certificate formats (not just SDSI/SPKI), and they allow cases where transfers of trust are expressed implicitly (e.g., via the X.509 "basicConstraints"). If a relying party Alice receives multiple certificates about Bob, and she successfully deduces their authenticity (i.e., the authenticity of the bindings they contain), then Alice may hold multiple sets of properties assigned to Bob. What kind of set operations should we allow on these sets of properties? Howell disallows the relying party to use the union operation, but allows intersection. Considering the universe of allowable set operations is another area for future work.

In sum, we briefly reviewed Maurer's calculus for reasoning about PKI systems, and illustrated its limitations. We then introduced a new model which generalized and extended Maurer's calculus to handle many real-world PKI concepts, such as the notion of authentic bindings, the consideration of certificate information, and the concept of time. Next, we used the new model to illustrate how it can be used to reason about real-world PKI systems that we have seen in the wild as well as in our lab. Finally, we discussed some of the model's limitations and directions for future work.

Acknowledgements

This work was supported in part by the Mellon Foundation, by the NSF (CCR-0209144), by Internet2/AT&T, by Sun, by Cisco, by Intel, and by the Office for Domestic Preparedness, U.S. Dept of Homeland Security (2000-DT-CX-K001). The views and conclusions do not necessarily represent those of the sponsors.

References

1. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System, version 2. IETF RFC 2704, September 1999.
2. M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The Role of Trust Management in Distributed Systems. *Secure Internet Programming*, 1603:185–210, 1999. Lecture Notes in Computer Science.
3. D. Chadwick, A. Otenko, and E. Ball. Role-Based Access Control with X.509 Attribute Certificates. *IEEE Internet Computing*, March-April 2003.
4. D. Cooper. A Model of Certificate Revocation. In *15th Annual Computer Security Applications Conference (ACSAC '99)*, pages 256–264, Phoenix, Arizona, USA, December 1999. IEEE Computer Society.
5. C. Ellison. The nature of a usable pki. *Computer Networks*, pages 823–830, 1999.
6. C. Ellison. Improvements on Conventional PKI Wisdom. In *Proceedings of the 1st Annual PKI Research Workshop*, April 2002.
7. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. IETF RFC 2693, September 1999.
8. S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. IETF RFC 3281, April 2002.
9. N. Goffee, S. Kim, S.W. Smith, P. Taylor, M. Zhao, and J. Marchesini. Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In *3rd Annual PKI Research and Development Workshop*. NIST, April 2004.
10. R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 3280, April 2002.
11. J. Howell and D. Kotz. A formal semantics for SPKI. In *Proceedings of the Sixth European Symposium on Research in Computer Security (ESORICS 2000)*, volume 1895 of *Lecture Notes in Computer Science*, pages 140–158. Springer-Verlag, October 2000.
12. P. Kocher. On Certificate Revocation and Validation. In *Proceedings of the Second International Conference on Financia l Cryptography (FC'98)*, volume 1465 of *LNCS*, pages 172–177. Springer-Verlag, 1998.
13. R. Kohlas and U. Maurer. Reasoning About Public-Key Certification: On Bindings Between Entities and Public Keys. *Journal on Selected Areas in Communications*, pages 551–560, 2000.
14. N. Li, B. Grosof, and J. Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):128–171, February 2003.
15. N. Li and Feigenbaum J. Nonmonotonicity, User Interfaces, and Risk Assessment in Certificate Revocation. In *Proceedings of the 5th International Conference on Financial Cryptography*, pages 166–177. Springer-Verlag, 2002.
16. N. Li, W. Winsborough, and J. Mitchell. Beyond Proof-of-compliance: Safety and Availability Analysis in Trust Management. In *Proceedings of 2003 IEEE Symposium on Security and Privacy,*, pages 123–139. IEEE Computer Society Press, May 2003.
17. M. Blaze and J. Feigenbaum and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.
18. M. Blaze and J. Feigenbaum and M. Strauss. Compliance-checking in the PolicyMaker Trust Management System. In *Proceedings of Second International Conference on Financial Cryptography (FC '98)*, volume 1465, pages 254–274, 1998.
19. A. Malpani, S. Galperin, M. Mayers, R. Ankney, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol. RFC2560, `http://www.ietf.org/rfc/rfc2560.txt`, June 1999.

20. U. Maurer. Modelling a Public-Key Infrastructure. In *ESORICS*. Springer-Verlag LNCS, 1996.
21. M. Naor and K. Nissim. Certificate Revocation and Certificate Update. *IEEE Journal on Selected Areas in Communications*, 18(4):561–570, April 2000.
22. J. Novotny, S. Tueke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. In *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*. IEEE Press, August 2001.
23. S.W. Smith. Outbound Authentication for Programmable Secure Coprocessors. *International Journal on Information Security*, 2004.
24. S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure Proxy Certificate Profile. `http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-10.txt`, 2003.
25. S. Weeks. Understanding Trust Management Systems. In *Proceedings of 2001 IEEE Symposium on Security and Privacy*, pages 94–105. IEEE Computer Society Press, May 2001.
26. V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. In *3rd Annual PKI Research and Development Workshop*, April 2004.

Classifying Public Key Certificates

Javier Lopez[1], Rolf Oppliger[2], and Günther Pernul[3]

[1] Computer Science Dept., University of Malaga, Malaga, Spain
jlm@lcc.uma.es
[2] eSECURITY Technologies, Gümligen, Switzerland
rolf.oppliger@esecurity.ch
[3] University of Regensburg, Germany
guenther.pernul@wiwi.uni-regensburg.de

Abstract. In spite of the fact that there are several companies that (try to) sell public key certificates, there is still no unified or standardized classification scheme that can be used to compare and put into perspective the various offerings. In this paper, we try to start filling this gap and propose a four-dimensional scheme that can be used to uniformly describe and classify public key certificates. The scheme distinguishes between (i) who owns a certificate, (ii) how the certificate owner is registered, (iii) on what medium the certificate (or the private key, respectively) is stored, and (iv) what type of functionality the certificate is intended to be used for. We think that using these or similar criteria to define and come up with unified or even standardized classes of public key certificate is useful and urgently needed in practice.

1 Introduction

It is commonly agreed that security is an important prerequisite for Internet-based electronic commerce. The term security, in turn, means different things to different people, and there are many security services one may think of. According to the OSI security architecture specified in ISO/IEC 7498-2, there are at least authentication, authorization, data confidentiality, data integrity, and non-repudiation services to distinguish [9].

Public key cryptography as originally proposed by Diffie and Hellman [7] provides an important technology to provide security services. Some of the services, such as non-repudiation services, cannot easily be provided without public key cryptography, whereas other services can be provided more efficiently with public key cryptography (as compared to secret key cryptography). This is particularly true for (entity or data origin) authentication services and the key establishment for data confidentiality and integrity services (see, for example, [14]).

With respect to its practical deployment, the Achilles heel of public key cryptography is public key certification, meaning that the authenticity of the public keys in use must be guaranteed, that is, certified by some trusted party (see, for example, Chapter 7 of [13]). This certification is usually done by *Certifi-*

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 135–143, 2005.

cation Authorities (CAs) or—more generally— *Certification Service Providers* (CSPs [4]).

In short, a CSP authenticates a public key by digitally signing it together with some identification or naming information about the key owner (and some other attributes). The result is a *digital* or *public key certificate*. A set of mutually trusting and cooperating CSPs forms a *Public Key Infrastructure* (PKI).

Public key cryptography can only be used in an efficient and effective way, if a PKI exists and is operated in some trusted way. Unfortunately, the establishment and successful commercial deployment has not really taken off so far (see, for example, [12] for a corresponding analysis).

There are many companies that (try to) act as CSPs and market public key certificates and corresponding services on a national or international level [5]. They all use policies and terminologies of their own, and it is getting increasingly difficult to tell their offerings apart and to put them into perspective.

Additionally, several standardization organizations are working on public key certificates and PKIs (e.g., ANSI, NIST, IETF, OASIS, ...). However, they work neither on a unified or even standardized terminology and set of policies, nor on interoperability issues. This is unfortunate, because it makes everything more involved from the user's perspective. An old proverb saying that every cat is black at night also applies to public key certificates, and hence it may be difficult to tell the various offerings of CSPs apart. Against this background, we believe that a unified or even standardized classification scheme is urgently needed to compare and put into perspective the offerings of the CSPs.

In this paper, we propose a classification scheme for public key certificates. The scheme distinguishes between (i) who owns a certificate, (ii) how the certificate owner is registered, (iii) on what medium the certificate (or the private key, respectively) is stored, and (iv) what type of functionality the certificate is intended to be used for. It goes without saying that the resulting four-dimensional classification scheme can be used as a starting point for further standardization activities. In fact, a classification scheme is only useful if many CSPs support it and specify their offerings according to this scheme. The rest of the paper is organized as follows. The related work is addressed in Section 2. The four classification criteria mentioned above are introduced and discussed in Section 3. A notation is proposed in Section 4, and a few major classes are overviewed and discussed in the same section. Finally, conclusions are drawn in Section 5.

2 Related Work

Each CSP has to specify a set of policies—*certificate policies* (CPs) and/or *certification practice statements* (CPSs)—to specify and nail down its offerings (see informational RFC3647 [6] for a corresponding framework). Unfortunately, most

[4] Note that the acronym CSP is sometimes also used to refer to a cryptographic service provider.

[5] http://www.openvalidation.org/en/service/calist.html

policies are written in a terminology of their own. This, in turn, makes it difficult to compare directly the various offerings of different CSPs, and to tell the sometimes subtle difference(s) between them.

To the best of our knowledge, there is no international standardization effort to define unified classes for public key certificates (in fact, the major goal of this paper is to initiate such an effort). In some countries, there are CSPs that work together in defining some unified classes of public key certificates. For example, in Switzerland, the Certification Service Providers Forum [6] has proposed five classes of public key certificates (i.e., *Bronze*, *Silver*, *Gold*, *Platinum*, and *Qualified*).

From a user's point of view, this classification is advantageous and allows them to better compare the offerings of various CSPs. Unfortunately, there are only two small CSPs that together form the Swiss CSP Forum [7], and hence the classification scheme is not widely deployed and used by all CSPs (even in Switzerland). To be really successful, a classification scheme needs to be agreed upon on an international level.

3 Classification Criteria

The initial classification scheme proposed in this paper distinguishes between the following four criteria:

Certificate owner: Who is the owner of the certificate and the corresponding private key?

Registration: How is the certificate owner registered? More specifically, how is the certificate owner identified and authenticated before the certificate is issued?

Storage medium: On what medium is the certificate (or the corresponding private key, respectively) stored?

Functionality: What type of functionality can the certificate and the corresponding private key be used for?

Note that the storage medium is not an inherent property of a public key certificate. In fact, it is possible and technically feasible to provide a certificate on different media. Nevertheless, we think that it is appropriate to take the storage medium into account, mainly because the certificate is coupled with a way to store the private key. So it is more a property of the private key (than the certificate).

There are other criteria one may think of. For example, one can distinguish whether a public key pair is generated by the user, generated by the CSP, or imported (from a potentially unknown source). For the purpose of this paper, however, we do not use key generation as a criterion for certificate classification. Instead, we argue that from the CSP's viewpoint, it does not really matter how a key pair is generated. The only thing that matters for the CSP (and for which

[6] http://www.csp-forum.ch

[7] The companies are SwissSign and SwissCERT.

the CSP can be held accountable) is that the certificate owner is registered as claimed in the CSP's policies. If the CSP offers complementary key generation services, then these services can be treated independently from the certification services (like many other trusted services, such as time-stamping services).

3.1 Certificate Owner

As its name suggests, the criterion "certificate owner" specifies who owns the certificate and the corresponding private key. We distinguish basically three possibilities:

- *Natural person:* The certificate is owned by a natural person. These are the certificates one usually has in mind when one elaborates on digital signature laws. In fact, in most legislations it is ultimately required that the certificate owner (i.e., the entity that is specified in the subject field) is a natural person. Furthermore, certificates for natural persons are used and widely deployed in the realm of secure messaging and secure authentication (i.e., single sign-on and secure firewall traversal).
- *Legal entity:* The certificate is owned by a legal entity, such as a company, an administrative entity, or a non-profit organization. From a business point of view, it is often argued that public key certificates owned by legal entities are ultimately required. In fact, many digital signature legislations have to struggle with the question whether it is possible to have legal entities issue legally-binding signatures (in addition to natural persons). There is, for example, an ongoing controversy on this topic in Germany and Switzerland. There are pros and cons on either side, and we are not going to get into this discussion in this paper.
- *Machine:* The certificate is owned by a computer system, device, or service. Examples include certificates for devices that implement the IP security (IPsec) protocol suite and certificates for Web servers that implement the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol. So far, certificates for machines have been the only certificates that have been able to be successfully deployed in the marketplace.

In some classification schemes, it is also possible to distinguish between certificates that are owned by CAs, root CAs, or software publishers. We think that the distinguishing feature in these cases is not who owns the certificate, but for what purpose is it actually to be used for. For example, a certificate owned by a (root) CA is used to issue other certificates, whereas a certificate owned by a software publisher is used to digitally sign software. In either case, the (root) CA or software publisher may be a natural person, a legal entity, or a machine. There are mainly commercial and/or legal reasons for CSPs to market software publisher certificates separately. Again, this point is not further addressed in this paper.

3.2 Registration

The criterion "registration" specifies how the certificate owner is registered, or—more specifically—who registers the certificate owner before the certificate is issued (i.e., is registration part of the service or not). There are basically three possibilities (and several sub-possibilities in the last case) to distinguish between.

- *No registration:* The certificate owner is not registered at all. The corresponding certificates are typically used for test purposes only. In fact, a certificate without registration does not make a lot of sense for all practical purposes.
- *Customer registration:* The certificate owner is registered by the customer organization. This type of registration is usually used if a CSP is providing virtual CA services. In this case, the CSP "only" offers its certificate issuing capabilities, whereas the actual work related to user registration is done by the customer organization.
- *Registration:* The certificate owner is registered by the CSP, or a registration authority (RA) working on behalf of the CSP, respectively. There are a number of possibilities to register the owner.

  - The certificate owner may be registered using some form of e-mail based identification and authentication (EBIA) as, for example, discussed in [8].
  - The certificate owner may be registered by a trusted organization acting as registration authority and using some existing identification and authentication mechanism (e.g., username and password). For example, if an organization already has a customer relationship, it can use this relationship to identify and authenticate certificate applicants. It goes without saying that the authentication information must be transmitted over some secure channel (e.g., an SSL/TLS connection).
  - The certificate owner may be registered personally by a trusted organization using strong identification and authentication mechanisms (using, for example, a photo ID).
  - The certificate owner may be registered personally by a trusted organization using some official identification and authentication document, such as a national ID card or passport.

  In either case, registration services can be provided by the CSP or delegated to partner companies acting as RAs, such as postal service providers or banks. In search of business plans to allow the successful marketing of certification services, many potential CSPs have been talking (and are still talking) to postal service providers and banks.

It goes without saying that a certification service without registration (i.e., no registration or customer registration) is substantially simpler to provide, and that the resulting business risks for the corresponding CSP can be made very small.

3.3 Storage Medium

There are several possibilities to store the certificate, or the private key, respectively. On a high level of abstraction, there are three possibilities to distinguish between.

- *Hardware:* The certificates employ special hardware devices to store private keys and corresponding certificates. Most of the time, it is assumed that all cryptographic computations with the private key are performed on the hardware device. Examples of hardware devices include smartcards and USB tokens.
- *Software:* The certificates do not employ hardware devices to store private keys and corresponding certificates. Instead, the private keys are stored in the application software that is going to use them. Most importantly, the Windows operating system can store the private keys of the registered users and make them available to all applications that can make use of them.
- *Server:* The private keys are stored on a server. They either never leave the server or are downloaded only temporarily and with special security precautions. Server-based certificates have many advantages for practical deployment, especially if one considers the mobility of users as an important criterion (see, for example, [15]).

From a security viewpoint, the use of special hardware and hardware-based certificates are certainly the preferred choice. It must also be said, however, that the security advantage of hardware certificates is frequently overemphasized, and that there are many possibilities to attack smartcards and other hardware tokens (cf. [3], [1], [11], [2], [4], [5], [10]).

Also, security is not always the main decision criterion and there are many situations in which the use of hardware-based certificates is prohibitively expensive. In these cases, the use of software-based certificates or server-based certificates provides a reasonable alternative. This is particularly true for software-based certificates that are frequently used in Windows operating systems. Using the auto-enrollment feature of the Windows 2003 PKI server, for example, certificates can be easily deployed in a corporate environment. The security of the corresponding certificates is directly coupled to the security of the user accounts for the Windows domains.

3.4 Functionality

There are basically three types of functionality a certificate and the corresponding private key can be used to provide.

- *Authentication:* The certificate can be used for authentication, meaning that its owner can use the private key to authenticate himself or herself to a (peer) entity.
- *Digital signatures:* The certificate can be used for digital signatures, meaning that its owner can use the private key to digitally sign documents (or public key certificates, respectively) and protect the authenticity and integrity of these documents accordingly.

– *Encryption:* The certificate can be used for encryption, meaning that its owner can use the private key to decrypt documents.

If a certificate and the corresponding private key are used to agree on a shared secret key (using, for example, a Diffie-Hellman key exchange [7]), then we consider encryption to be the functionality that is actually provided. Consequently, we do not consider key agreement to be a functionality of its own. Note that this is a simplification, because the secret key can also be used to protect the authenticity and integrity of a document (using, for example, a message authentication code).

4 Notation and Major Classes

Because our classification approach comprises four different criteria, it is necessary to use a four-dimensional notation to refer to public key certificates. More specifically, the term [⟨O⟩-⟨R⟩-⟨M⟩-⟨F⟩]-certificate refers to a certificate with owner ⟨O⟩, registration ⟨R⟩, storage medium ⟨M⟩, and functionality ⟨F⟩.

– Possible values for owner ⟨O⟩:
  - NP: Natural person
  - LE: Legal entity
  - M: Machine
– Possible values for registration ⟨R⟩:
  - NR: No registration
  - CR: Customer registration
  - R1: EBIA
  - R2: Authentication based on some existing identification and authentication mechanism
  - R3: Personal authentication based on a strong identification and authentication mechanism
  - R4: Personal authentication based on an official identification and authentication document, such as a national ID card or a passport
– Possible values for storage medium ⟨M⟩:
  - HW: Hardware
  - SW: Software
  - S: Server
– Possible values for functionality ⟨F⟩:
  - A: Authentication
  - E: Encryption
  - S: Digital signatures

  These values can be combined, i.e., AE refers to authentication and encryption.

The classification scheme yields $3 \cdot 6 \cdot 3 \cdot 3 = 162$ certificate classes, and the criteria can even be further refined at will. Consequently, there are many certificate classes that are not used, and that do not make a lot of sense in practice. For example, if we use very strong registration mechanisms but employ server-based certificates, then the overall security is somehow difficult to evaluate.

In practice, it is possible and likely that we see only a small fraction of all possible certificate classes being offered by CSPs. As an example, only a few of these classes are overviewed and discussed:

- [NP-R4-HW-S]: Certificates from this class are issued for natural persons. Registration is as strong as possible and the private key is stored on a hardware device. Furthermore, the certificates can be used for digital signatures. Certificates from this class play a major role in digital signature legislations and discussions about electronic ID cards (comprising public key certificates).
- [NP-CR-SW-A]: Certificates from this class are issued for natural persons that are registered by the customer organization(s). The private key is stored in software, and the certificates can be used for authentication purposes. Certificates from this class are frequently used for single sign-on and secure firewall traversal.
- [M-CR-SW-AES]: Certificates from this class are issued for machines that are registered by the customer organization(s). The private key is stored in software, and the certificates can be used for authentication, encryption, and digital signatures. Note that the certificates may actually consist of three certificates; one for authentication, one for encryption, and one for digital signatures.

It goes without saying that many other classes may be used and play an important role in practice.

5 Conclusions

In this paper we have argued that a unified (or even standardized) classification scheme is urgently needed to compare and put into perspective the various offerings of commercially operating CSPs (without having users read and compare all CPSs). This is particularly true if CSPs are to become successful. Against this background, we proposed a four-dimensional classification scheme that can be used to briefly describe and characterize the offerings of CSPs. The scheme distinguishes between (i) who owns a certificate, (ii) how the certificate owner is registered, (iii) on what medium the certificate (or the private key, respectively) is stored, and (iv) what type of functionality the certificate is intended to be used for. There is a total of 162 possible certificate classes in the scheme, and some exemplary classes are briefly mentioned. If there is concensus about the look and feel of the major classes, one may introduce abreviations to refer to them (e.g., A, B, C, ... or 1, 2, 3, ...). In either case, it will be interesting to see what classes are actually populated and supported by commercially-operating CSPs.

References

1. Anderson, R., and M. Kuhn, Tamper Resistance — A Cautionary Note, Proceedings of the 2nd USENIX Workshop on Electronic Commerce, November 1996, pp. 1–11.
2. Anderson, R., and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," *Proceedings of the 5th International Workshop on Security Protocols*, Springer-Verlag, LNCS 1361, 1997, pp. 125–136.
3. Anderson, R., "Why Cryptosystems Fail," *Communications of the ACM*, Vol. 37, No. 11, November 1994, pp.32–40.
4. Boneh, D., R. DeMillo, and R. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," *Proceedings of EUROCRYPT '97*, Springer-Verlag, LNCS 1233, 1997, pp. 37–51.
5. Biham, E., and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," *Proceedings of CRYPTO '97*, Springer-Verlag, LNCS 1294, 1997, pp. 513–525.
6. Chokhani, S., et al., Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework, RFC 3647, November 2003.
7. Diffie, W., and M.E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, IT-22(6), 1976, pp. 644–654.
8. Garfinkel, S.L., "Email-Based Identification and Authentication: An Alternative to PKI?" *IEEE Security & Privacy*, Vol. 1, No. 6, November-December 2003, pp. 20–26.
9. ISO/IEC 7498-2, Information Processing Systems—Open Systems Interconnection Reference Model—Part 2: Security Architecture, 1989.
10. Kocher, P., J. Jaffe, and B. Jun, "Differential Power Analysis," *Proceedings of CRYPTO '99*, Springer-Verlag,LNCS 1666, 1999, pp. 388–397.
11. Kocher, P., "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," *Proceedings of CRYPTO '96*, Springer-Verlag, LNCS 1109, 1996, pp. 104–113.
12. Lopez, J., R. Oppliger, and G. Pernul, "Why have public key infrastructures failed so far?," work in progress.
13. Oppliger, R., *Security Technologies for the World Wide Web, Second Edition*, Artech House Publishers, Norwood, MA, 2003.
14. Oppliger, R., *Contemporary Cryptography*, Artech House Publishers, Norwood, MA, 2005.
15. Oppliger, R., "Server-based Signatures: A Different Approach," work in progress.

Identity Based Ring Signature: Why, How and What Next

Sherman S.M. Chow*, Richard W.C. Lui, Lucas C.K. Hui, and S.M. Yiu

Department of Computer Science, The University of Hong Kong,
Pokfulam, Hong Kong
{smchow, wclui, hui, smyiu}@cs.hku.hk

Abstract. This paper gives a solid and inspiring survey of ID-based ring signatures from a number of perspectives. It is well known that ID-based cryptosystems provide some advantages that traditional public key infrastructure (PKI) cannot achieve. What advantages do ID-based ring signature schemes possess that PKI-based schemes do not? Many ID-based ring signature schemes have been proposed. What is the design philosophy behind existing ID-based ring signature schemes? This paper summarizes the study of ID-based ring signature schemes in the literature, investigates their relationships with other existing cryptographic schemes, describes the extension of ID-based ring signature schemes and the related supporting protocol, reviews the state-of-the-art and discusses a number of interesting open problems.

Keywords: Identity based cryptography, ring signature, spontaneous anonymous group signature, PKI, bilinear pairings.

1 Introduction

Identity-Based Cryptography. In 1984, Shamir introduced the notion of *identity-based (ID-based) cryptography* [38] to solve the certificate management problem (or the public key distribution problem). The distinguishing property of ID-based cryptography is that a user's public key can be any binary string, such as an email address, that can identify the user. Then a trusted party called a *private key generator* (PKG) generates the corresponding private key on the user's demand, with the help of the PKG's master secret key. Since the public key can be easily derived, PKG does not need to maintain a list of issued certificates. Each user only needs to store the PKG's system parameters instead of a database of certificates of other users, hence ID-based cryptography is supposed to provide a more convenient alternative to the traditional public key infrastructure (PKI).

Shamir suggested a concrete ID-based signature scheme; however, ID-based encryption scheme (IBE) was left as an open question. There have been several constructions of IBE afterwards, but none of these proposals are fully satisfactory until the work of Boneh and Franklin in 2001 [4].

* Corresponding author.

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 144–161, 2005.

They proposed the first practical IBE scheme by utilizing bilinear pairings. Afterwards, bilinear pairings have been used extensively in the design of ID-based schemes (e.g. [2,3,8,11,12,13,14,15,22,23,24,26,31,33,39,44,45,46,47,48]) and other cryptographic schemes (e.g. [5,6,30,50]).

Ring Signature. Anonymity is becoming a major concern in many multi-user electronic commerce applications. Group-oriented signature schemes [10] enable an entity of a group to produce a signature on behalf of the group. There are two major paradigms in anonymous group-oriented signature schemes: group signature and ring signature. In a group signature scheme, the group is predefined and there is a group manager that can revoke this anonymity. The ring signature scheme provides a similar feature. It does not support an anonymity revocation mechanism, but no setup stage is needed to produce and distribute a group secret explicitly. Hence it enables any individual to spontaneously conscript arbitrarily $n-1$ entities and generate a publicly verifiable 1-out-of-n signature on behalf of the whole group, yet the actual signer remains anonymous.

ID-Based Ring Signature. ID-based cryptography and ring signature schemes have rapidly emerged in recent years. Their combination: ID-based ring signature, has been well-studied as well. The state-of-the-art achieved a constant number of pairing computations [14,31] and also a constant size signature [31] already. Besides, various research work on its extensions [12,13,14,22,24] and applications [25,40] has already appeared. It may be an appropriate time to have a summary of the existing work now.

The rest of the paper is organized as follows. The next section presents a short survey of ring signature schemes. The need for ID-based ring signature are discussed in Section 3. All previous ID-based ring signature schemes are reviewed in Section 4, together with investigations of the design philosophy behind them. In Section 5, we review some extensions of ID-based ring signatures, which includes the versions for a threshold number of signers (t-out-of-n) and 1-out-of-n-groups extensions, and also ring signcryption that achieve confidentiality at the same time. An example of supporting protocol of ID-based ring signature schemes is reviewed in Section 6. Finally, Section 7 concludes the paper by stating the state-of-the-art and some possible future research directions.

2 Related Work

2.1 Traditional PKI-Based Ring Signature

The ring signature scheme was first formalized by Rivest, Shamir and Tauman in [36], which is a solution based on trapdoor one-way permutations (e.g. RSA [35] and Rabin [34]). Actually, this concept has been discussed by Cramer, Damgård and Schoenmakers [16] in 1994 as a witness hiding proofs of a partial knowledge interactive protocol. When instantiated with the Fiat-Shamir heuristic [19], it gives a ring signature scheme. We call such class of scheme a CDS-type ring

signature. Discrete logarithm key was considered in [1]. Furthermore, [1] provided a construction applicable for several flavours of public-keys (e.g. integer factoring based and discrete logarithm based). Subsequent work on ring signatures re-investigated the problem from different perspectives. A ring signature scheme based on the Nyberg-Rueppel signature scheme [32] was proposed in [20]. Instead of unconditional anonymity, Liu *et al.* considered a "middle" level of anonymity between group signatures and ring signatures and proposed the notion of *linkable* ring signature [29]. Such a scheme can let anyone to determine if two ring signatures are signed using the same private key.

In threshold ring signature schemes, any group of t entities spontaneously conscript arbitrarily $n - t$ entities to generate a publicly verifiable t-out-of-n signature on behalf of the whole group, yet the actual signers remain anonymous. Bresson *et al.* [7] extended the ring signature scheme into a threshold ring signature scheme using the concept of partitioning. Later, Wong *et al.* [43] proposed another threshold ring signature using a tandem construction method. There are some threshold ring signature schemes with special properties. Tsang *et al.* [41] introduced individual-linkability to threshold ring signatures, which enables anyone to determine if two ring signatures are signed with the help of the same signer; and provided a better solution than the trivial extensions of [29] by concatenating the threshold number of linkable ring signatures. Chan *et al.* [9] constructed a CDS-type [16] t-out-of-n blind threshold ring signature, such that the signers do not know what exactly they are signing and cannot link which invocation of signing algorithm corresponds to which unblinded signature; Liu *et al.* [28] incorporated separability [1] into a threshold ring signature scheme, which enables the use of various flavours of public keys in a single threshold ring signature. The authors have illustrated their generic construction by using RSA [35] and Schnorr signature [37].

There are a number of pairing-based ring signature schemes. Inspired by the aggregate signature, a ring signature scheme was proposed in [5]. A technique similar to that of [5] was used to derive a new ring signature scheme in [44]. In [50], a ring signature scheme was derived from a short signature scheme. Proxy ring signature was considered in [2] and [49], and threshold ring signature from pairings was proposed in [30] and [42].

Ring signatures are called as spontaneous anonymous group signatures in some of the literature, such as [9], [27], [29] and [42].

Due to space limitations, we refer our readers to work like [14] for the framework and the security notions of an ID-based ring signature, and the cryptographic primitive used in most ID-based ring signature: bilinear pairings.

2.2 ID-Based Signature

Before the discussion of ID-based ring signatures, we should get some idea of how a basic ID-based signature scheme is devised. There are a number of ID-based signature schemes [3,8,23,33]. Here we review two of them [8,23], which are related to our discussion on two existing ID-based ring signature schemes. We firstly describe the setup procedure and the private key extraction process of the

PKG, which is common to all of the ID-based (ring) signature schemes described in this paper. We refer the interested reader to [3] for a solid treatment on the security of these two schemes and many other ID-based signature schemes.

Setup and Extract Algorithm of ID-Based Paradigm
`Setup`: The PKG randomly chooses $s \in_R \mathbb{Z}_q^*$, keeps it as the master secret key and computes the corresponding public key $P_{pub} = sP$. Let $H_1(\cdot)$ be a cryptographic hash function where $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$. This hash function is for hashing any arbitrary identity into a value representing the user's public key. In addition, we require another cryptographic hash function $H_2(\cdot)$ for the ID-based ring signature schemes, where $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The system parameters are: $params = \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}(\cdot,\cdot), q, P, P_{pub}, H_1(\cdot)\, H_2(\cdot)\}$.

`Extract`: The user with identity $ID \in \{0,1\}^*$ submits ID to the PKG. The PKG sets the user's public key Q_{ID} to be $H_1(ID)$, computes the user's private key S_{ID} as sQ_{ID}, where $s \in \mathbb{Z}_q^*$ is the master secret key of the PKG. Then the PKG sends the private key to the user over a secure channel.

Sign and Verify Algorithm of Two ID-Based Signature Schemes
Let m be the message to be signed and ID be the identity of the signer.

Hess's Scheme [23]:	
`Sign`:	`Verify`:
1. Select an element $A \in_R \mathbb{G}_1$.	On receiving a message m and a
2. Compute $r = \hat{e}(A, P)$.	signature $\sigma = (c, V)$, the verifier
3. Compute $c = H_2(m\|\|r)$.	accepts the signature if and only if
4. Compute $V = cS_{ID} + A$.	$c = H_2(m\|\|\hat{e}(P,V) \cdot \hat{e}(H_1(ID), -P_{pub})^c)$.
5. Output the signature as (c, V).	
Cha and Cheon's Scheme [8]:	
`Sign`:	`Verify`:
1. Select an element $r \in_R \mathbb{Z}_q^*$.	On receiving a message m and a
2. Compute $U = rQ_{ID}$.	signature $\sigma = (U, V)$, the verifier
3. Compute $c = H_2(m\|\|U)$.	computes $c = H_2(m\|\|U)$ and
4. Compute $V = (r + c)S_{ID}$.	accepts the signature if and only if
5. Output the signature as (U, V).	$\hat{e}(P,V) = \hat{e}(P_{pub}, U + cQ_{ID})$.

3 Motivations

3.1 Arguments in Favour of the PKG's Roles

Before evaluating the benefit brought by the ID-based paradigm, we first consider a fundamental question: are ID-based ring signature schemes really ring signature schemes?

When the notion of ring signature was introduced [36], the key properties of ring signatures include: no group managers, no setup procedures, and no coordination. For existing ID-based ring signature schemes, the PKG has to be completely trustworthy due to the inherent key escrow property, which can be regarded as a kind of group manager. So ID-based ring signatures are not considered as *completely spontaneous* by some researchers, e.g. it is regarded as *partial spontaneous group signature* in the study of spontaneous anonymous group cryptography performed by Liu [27].

We hold a different point of view. For PKI-based ring signature, the existence of a certificate authority (CA) is assumed, and each possible signer of a ring signature is assumed to have obtained a digital certificate from the CA already. The involvement of a CA and a PKG is only for setting up the parameters of the whole system, but not for the setting up of the signers group. In this regard, CAs in PKI also assume the role of "group manager" if PKGs in an ID-based paradigm do. From the trusted level perspective, a CA can forge a user's certificate and impersonate the user to communicate and impersonate the user by using the private key associated with the forged certificate. However, the user can accuse the dishonest CA by showing another different but also valid certificate. Indeed, for ID-based scenarios, we can also integrate a similar mechanism, which gives users the power to provide a proof of treachery when impersonation occurs [11].

From the above discussion, we see no strong reason to reject ID-based ring signature schemes from being ring signature schemes.

3.2 Cost Related to Certificates

In a traditional public key infrastructure (PKI), the public key is usually a "random" string that is unrelated to the identity of the user, so there is a need for a trusted-by-all CA to assure the relationship between the public key and the user. Therefore, any verifier of a signature must obtain a copy of the user's certificate and check the validity of the certificate before checking the validity of the signature. In ring signatures, not only the verifier must verify all the public keys of the group. The signer must do so as well to preserve his or her anonymity (consider the extreme case that all certificates used are indeed invalid except the signer's one). The communication and the validation of a large number of public keys greatly affects the efficiency of the scheme. Using ID-based ring signatures avoids the necessity of certificates and the authentication of the public keys.

We make a remark that for an ID-based ring signature scheme to maintain its advantage over the PKI-based counterpart, it should not involve computationally intensive operations in which the number of executions grows linearly with the number of possible signers. Chow *et al.*'s scheme (to be reviewed in a later section) satisfies this requirement.

3.3 Spontaneity

Real spontaneity is not always possible for ring signatures under traditional PKIs. As argued previously, the signer cannot spontaneously conscript users

who have not registered for a certificate, or the verifier can figure out the real signer more easily.

ID-based ring signatures solved this problem: the public key of each user can be easily computed from a string corresponding to his or her identity, associating an implicit public key to each person in the world.

Actually the real spontaneity of an ID-based ring signature relies on the assumption that the PKG do not reveal any information about who has requested his or her private key and who has not. In [39], a separable and anonymous ID-based key issuing protocol was proposed such that any eavesdropper cannot learn the identity associated with the private key being issued even though the key is not transmitted via a secure channel. We review this protocol in the Section 6.

4 Design Philosophy

In this section, we review four major existing ID-based ring signature schemes. Conceptually, the signing algorithm involves three phases before outputting the final ring signature, which are *initialization*, *generating the ring sequence for non-participating signers* and *closing the ring*. Instead of merely giving the construction details of each scheme, we try to discuss the key idea behind their design. Notice that we tried to unify the notations in different schemes' descriptions.

4.1 Notations

Let $L = \{ID_1, ID_2, \cdots, ID_n\}$ be the set of all identities of n users and m be the message to be signed. Also let k be the index of the actual signer (i.e. his or her public key is $Q_{ID_k} = H_1(ID_k)$). Note that k should be chosen at random each time to preserve the signer's anonymity.

4.2 Zhang and Kim 's Scheme

Design Philosophy. Zhang and Kim's scheme [48] shares similar structures as that of Abe *et al.*'s discrete logarithm based scheme [1], in which the public-private key pair is in the form of $(y = g^x \bmod p, x)$, where p is a prime and g is the generator of $\mathbb{Z}_p^*$: a group of prime order q. We first give a special case of Abe's construction, where all the signers share the same generator g. Let H be a cryptographic hash function mapping $\{0,1\}^*$ to $\mathbb{Z}_q$.

Abe *et al.*-`Sign`:

- *Initialization:* Randomly choose an element a from $\mathbb{Z}_q$ and compute $c_{k+1} = H(L||m||g^a)$.
- *Generating the ring sequence for non-participating signers:* For $i = k+1, \cdots, n-1, 0, \cdots, k-1$ (i.e. modular arithmetic is considered), compute $c_{i+1} = H(L||m||g^{r_i} {y_i}^{c_i} \bmod p)$, where r_i is randomly chosen from $\mathbb{Z}_q$.

- *Closing the ring:* Compute $r_k = a - c_k x_k \bmod q$
  (which is equivalent to solving $g^a = g^{r_k} {y_k}^{c_k} \bmod p$ for r_k).
- *Output the signature:* Output the signature $\sigma = \{c_0, r_0, r_1, \cdots, r_{n-1}\}$.

Abe *et al.*-`Verify`:

1. For $i = 0, 1, \cdots, n-1$, compute $c_{i+1} = H(L||m||g^{r_i} {y_i}^{c_i} \bmod p)$.
2. Accept if $c_n = c_0$, reject otherwise.

The above construction is a realization of Abe *et al.*'s generic construction from three-move honest verifier zero-knowledge proofs. *Initialization* generates the first-move commitment g^a from randomness a and the generation of challenge c_{k+1} from message m and commitment. For the *ring sequence generation*, the verification conditions are simulated for each possible signer. In *closing the ring*, the secret key x_k and the randomness a are used to generate the response to the challenge c_{k+1}.

In Zhang and Kim's scheme, we can see a similar structure that can be regarded as a "bilinear pairing version" of the above construction. $\hat{e}(P, P)$, a generator of $\mathbb{G}_2$, is chosen to be the generator of the scheme. A random value A is chosen from $\mathbb{G}_1$ and the commitment is $\hat{e}(A, P) \in \mathbb{G}_2$, which can be regarded as generated from the chosen generator as $\hat{e}(A, P) = \hat{e}(P, P)^a$, where $A = aP$. For the *ring sequence generation*, we can see the term $g^{r_i} {y_i}^{c_i}$ manifested in the form of $\hat{e}(R_i, P)\hat{e}(H_1(ID_i), P_{pub})^{c_i}$. The first component is the randomness introduced to make each commitment look random and the second component is computed from the user's public key and also the commitment for the previous signer. In *closing the ring*, the secret key S_{ID_k} and the randomness A are used to generate the response to the challenge c_{k+1}, which satisfy the simulation for each possible signer in the phase of *ring sequence generation*.

As noted by Zhang and Kim [48], their scheme reduces to the ID-based signature by Hess [23] (which is reviewed in Section 2.2) when there is only one signer. Indeed, their scheme can be obtained by applying Abe *et al.*'s generic construction of ring signature on the three-move based signature by Hess [23].

Construction

`Sign`:

- *Initialization:* Randomly choose an element A from $\mathbb{G}_1$ and compute $c_{k+1} = H_2(L||m||\hat{e}(A, P))$.
- *Generating the ring sequence for non-participating signers:* For $i = k+1, \cdots, n-1, 0, \cdots, k-1$ (i.e. modular arithmetic is considered), randomly choose element R_i from $\mathbb{G}_1$, and then compute

$$c_{i+1} = H_2(L||m||\hat{e}(R_i, P)\hat{e}(c_i H_1(ID_i), P_{pub})).$$

- *Closing the ring:* Compute $R_k = A - c_k S_{ID_k}$
  (which is equivalent to solving $\hat{e}(A, P) = \hat{e}(R_k, P)\hat{e}(c_k H_1(ID_k), P_{pub})$ for R_k).
- *Output the signature:* Output the signature $\sigma = \{c_0, R_0, R_1, \cdots, R_{n-1}\}$.

`Verify`: For $i = 0, 1, \cdots, n-1$, compute $c_{i+1} = H_2(L||m||\hat{e}(R_i, P)\hat{e}(c_i H_1(ID_i), P_{pub}))$; accept if $c_n = c_0$, reject otherwise.

4.3 Lin and Wu 's Scheme

Design Philosophy. Lin and Wu 's scheme [26] is quite similar to Zhang and Kim's [48], instead of including the value of pairing as the input of hash function, they simply make them available in the signature, while the hash value of the group of signers together with the message is included in the generation of the ring sequence. Such modifications are driven by the need to improve the verification's efficiency. Since each randomness term R_i (and also the term $c_i H_1(ID_i)$) are not the input of the hash function, they can be aggregated together for verification. By the bilinearity of pairing, verification only requires two pairing operations while Zhang and Kim's scheme [48] requires $2n$ of them.

Note that there are some minor inconsistencies in the original description of the scheme in [26], as pointed out by [2]. This error can be fixed by using an extra hash function $H_3 : \mathbb{G}_2 \rightarrow \mathbb{Z}_q$. We remark that the proposed scheme in [2] is very similar to the corrected version of [26]. Both [26] and [2] do not have a formal security analysis, so their constructions are not covered in this survey.

4.4 Herranz and Sáez 's Scheme

Design Philosophy. The schemes described so far at this point employ a "ring structure": the challenge term c_i from the hash function is used as the input of the hash function to generate the next challenge term c_{i+1}. The ring sequence is generated until it reaches the starting point c_k, then the ring can be closed by making the random term used to generate c_{k+1} to be in the same form as the other terms. To do so, we need to solve this equation (as illustrated in the above schemes), and this can only be done with the help of the secret key of user ID_k since the randomness introduced in the starting point has been committed into the value of c_k.

Herranz and Sáez 's scheme [22] does not use this ring structure. Instead, all challenge terms for non-participating signers are generated independently. To close the ring, the signer uses the secret key to cancel out all other terms in the verification equation. Compared with the above schemes, such a design allows the operations in signing and verification to be parallelized.

Actually, this design is related to the "attack" in a work later than Herranz and Sáez's proposal, where the problem of batch verifying ID-based signatures was investigated [45]. They showed that existing ID-based signature schemes prior to their work cannot support batch verifications of multiple signatures by showing attacks on aggregate verification mechanisms. The failure of aggregate verification comes from the gap between the possibility of using only a single private key to "sign" n messages and the desired requirement of using at least n private keys to sign n messages.

We point out that the scheme below can be regarded as an extension of the signature scheme by Hess [23] described in Section 2.2. As explained in [45], since the verification process of the scheme involves the computation of hash value and the comparison of this result with the value included in the signature, aggregate verification of n signatures sound impossible as the secure hash function for signature schemes does not have homomorphic property. To allow the aggregate verification of n signatures, the equation in the verification should be modified to $\hat{e}(P, V) = r \cdot \hat{e}(P_{pub}, H_2(m||r)Q_{ID})$.

It is easy to see the completeness of the aggregate verification as it is quite obvious that n valid signatures $(\{(r_1, V_1), (r_2, V_2), \cdots, (r_n, V_n)\})$ from n different signers $(\{ID_1, ID_2, \cdots, ID_n\})$ can pass the aggregated verification algorithm $\hat{e}(P, \sum V_i) = \prod r_i \cdot \hat{e}(P_{pub}, \sum H_2(m_i||r_i)Q_{ID_i})$. How about the soundness? Indeed, using a single private key instead of n private keys can make this aggregated verification pass, as shown in the ID-based ring signature scheme below.

Note that these concepts about the design of ID-based ring signature were not discussed in [22] (and also [45]).

Construction
`Sign`:

- *Initialization:* Randomly choose an element A from $\mathbb{G}_1$.
- *Generating the ring sequence for non-participating signers:* For $i = k + 1, \cdots, n-1, 0, \cdots, k-1$ (i.e. modular arithmetic is considered), randomly choose element R_i from $\mathbb{G}_1$, compute $r_i = \hat{e}(R_i, P)$ and $c_i = H_2(L||m||r_i)$.
- *Closing the ring:* Firstly compute $U = \sum_{i \in \{0,\cdots,n-1\} \setminus \{k\}} \{c_i H_1(ID_i)\}$, then $r_k = \hat{e}(A, P)\hat{e}(-P_{pub}, U)$ and $c_k = H_2(L||m||r_k)$. Finally compute $V = c_k S_{ID_k} + A + \sum_{i \in \{0,\cdots,n-1\} \setminus \{k\}} \{R_i\}$.
- *Output the signature:* Output the signature $\sigma = \{V, r_0, r_1, \cdots, r_{n-1}\}$.

`Verify`: Compute $c_i = H_2(L||m||r_i)$ for $i = 0, 1, \cdots, n-1$; accept the signature if $\hat{e}(P, V) = \prod_{i=0}^{n-1} \{r_i\} \hat{e}(P_{pub}, \sum_{i \in \{0,\cdots,n-1\}} \{c_i H_1(ID_i)\})$, reject otherwise.

4.5 Chow *et al.*'s Scheme

Design Philosophy. Chow *et al.*'s scheme [14], although sharing a similar design, further improved Herranz and Sáez 's scheme [22]. Instead of directly exploiting the failure of an ID-based signature scheme to support efficient batch verification, the scheme includes its own modification to the implicit underlying signature scheme (Cha and Cheon's scheme [8]) to further improve the scheme's efficiency. To the best of the authors' knowledge, it is the most efficient (ID-based or non-ID-based) ring signature scheme from bilinear pairings, which requires only two pairings computations in verification and zero of them in signing.

Construction
`Sign`:

1. Choose $U_i \in_R \mathbb{G}_1$ and $c_i = H_2(m||L||U_i)$ $\forall i \in \{1, 2, \cdots, n\} \backslash \{k\}$.
2. Choose $r'_k \in_R \mathbb{Z}_q^*$, compute $U_k = r'_k Q_{ID_k} - \sum_{i \neq k} \{U_i + c_i Q_{ID_i}\}$.
3. Compute $c_k = H_2(m||L||U_k)$ and $V = (c_k + r'_k) S_{ID_k}$.
4. Output the signature for m and L as $\sigma = \{U_1, U_2, \cdots, U_n, V\}$.

`Verify`: Compute $c_i = H_2(m||L||U_i)$ for $i = 0, 1, \cdots, n-1$; accept the signature if $\hat{e}(P, V) = \hat{e}(P_{pub}, \sum_{i=1}^{n} (U_i + c_i Q_{ID_i}))$, reject otherwise.

4.6 Comparison

Chow *et al.*'s scheme is the most efficient one among the schemes described. To sign a message involving n signers, it only takes $2n-2$ point additions on $\mathbb{G}_1$ and $n+1$ point scalar multiplications on $\mathbb{G}_1$. For the verification of this signature, the verifier needs to perform only $n+1$ point additions on $\mathbb{G}_1$ n point scalar multiplications on $\mathbb{G}_1$, and 2 pairing operations. Neither $\mathbb{G}_2$ or $\mathbb{Z}_q^*$ multiplication nor `MapToPoint` hashing from the BLS short signature [6] is required. As Herranz and Sáez 's scheme [22], the operations in signing and verification of Chow *et al.*'s scheme can also be parallelized. Due to the length constraint, we refer the reader to [14] for a complete summary on the efficiency comparison of the schemes described in this section.

For security, all of the provably secure schemes above existentially unforgeable in the random oracle model, assuming the intractabilities of the computational Diffie-Hellman problem. We remark that the full proof of the Zhang and Kim's scheme [48] was given in a separate paper [21].

5 Extensions

ID-based ring signature is a 1-out-of-n-individuals signature. This section reviews two concepts which can be seen as its extensions, which are t-out-of-n threshold ring signature and 1-out-of-n-groups ring signature. After that, we outline two extensions [12,24] that integrate the idea of ID-based ring signature and ID-based signcryption (e.g. [15,46]).

5.1 Threshold

Design Philosophy. Consider Cha and Cheon's ID-based signature scheme [8], if the challenge value c is not the output of the hash value of $(m||U)$ but chosen arbitrary by the signer, it is easy to generates a "valid-looking" signature by setting $U = xP - cQ_{ID}$ and $V = xP$, where $x \in \mathbb{Z}_q^*$. By a similar reasoning, if $n - t$ challenge terms are chosen arbitrarily and t other challenge terms are determined by the previous $n - t$ challenge terms, it is possible to devise a t-out-of-n threshold signature. The scheme in [13] employed this idea by using the

interpolation of $n-t$ arbitrary challenges to generate the remaining t challenges. A similar technique has been applied in other threshold ring signature schemes such as [28] and [30].

Construction
`Sign`: Let L be the set of all identities of the n users. Without loss of generality, we assume users indexed by $\{1, 2, \cdots, t\}$ are the participating signers while users indexed by $\{t+1, t+2, \cdots, n\}$ are the non-participating signers. The participating signers carry out the following steps to give an ID-based threshold ring signature.

1. An arbitrary entity (which is trusted to keep the identities of the participating signers confidential) "prepares the signature on behalf of" other entities in the group by performing the following computations: For $i \in \{t+1, \cdots, n\}$, chooses x_i and $c_i \in_R \mathbb{Z}_q^*$ and computes $U_i = x_i P - c_i P_{pub}$ and $V_i = x_i Q_{ID_i}$.
2. For $j \in \{1, \cdots, t\}$, each signer ID_j chooses $r_j \in_R \mathbb{Z}_q^*$ and computes $U_j = r_j P$.
3. Anyone in the group of t participating signers who got the knowledge of $\bigcup_{k=1}^{n}\{U_k\}$ computes $c_0 = H_2(L||t||m||U_1||U_2||\cdots||U_n)$ and constructs a polynomial f of degree $n-t$ over $\mathbb{Z}_q$ such that $f(0) = c_0$ and $f(i) = c_i$ for $t+1 \leq i \leq n$.
4. For $j \in \{1, \cdots, t\}$, each signer ID_j computes $c_j = f(j)$ and $V_j = r_j Q_{ID_j} + c_j S_{ID_j}$.
5. Anyone in the group of t participating signers who got the knowledge of $\bigcup_{k=1}^{n}\{V_k\}$ computes $V = \sum_{k=1}^{n} V_k$.
6. Output the signature for m and L as $\sigma = \{U_1, U_2, \cdots U, V, f\}$.

`Verify`: A verifier checks whether a signature $\sigma = \{\bigcup_{k=1}^{n}\{U_k\}, V, f\}$ for the message m is given by at least t signers from the set of users L as follows.

1. Check if the degree of polynomial f is $n-t$ and $H_2(L||t||m||U_1||\cdots||U_n)$ is the constant term of f. Proceed if both conditions are true, reject otherwise.
2. For $k \in \{1, \cdots, n\}$, compute $c_k = f(k)$.
3. Check whether $\hat{e}(P, V) = \prod_{k=1}^{n} \hat{e}(Q_{ID_k}, U_k + c_k P_{pub})$. If the equality holds, return $\top$. Otherwise, return $\bot$.

Extra Features. It is quite often that different users join different PKGs in reality. In [46], the notion of trusted authorities compatibility (or in our terms, PKG compatibility) is introduced in the ID-based signcryption (e.g. [15,46]) scenario. This notion was considered in the above threshold ring signature scheme. For ID-based threshold ring signature schemes, spontaneity will be affected if the intended group of signers joined different PKGs. Under the assumption that different PKGs have chosen the same security level and the same elliptic curve equipped with bilinear pairings, the above scheme can be

easily extended to support this requirement by changing the equality to be checked in the verification algorithm to $\hat{e}(P,V) = \prod_{k=1}^{n} \hat{e}(Q_{ID_k}, U_k + c_k P_{pub_k})$, where P_{pub_k} is the public key of the PKG of the k-th user.

For a threshold ring signature scheme that does not support robustness, the misbehavior of any participating signer cannot be detected, and the final signature generated by the group of signers will be invalid even if there is only one misbehaving signer. In the above scheme, the partial signature $\sigma_j = \{c_j, U_j, V_j\}$ generated by the signer ID_j can be verified easily by checking whether $\hat{e}(P, V_j) = \hat{e}(Q_{ID_j}, U_j + c_j P_{pub})$ holds.

5.2 1-Out-of-n-Groups

Both of the work of Herranz and Sáez [22] and Chow *et al.* [14] considered this extension. Hereafter we review the extended scheme from Chow *et al.* as it is more efficient.

Design Philosophy. With the concept of "aggregated public key", it is easier to explain how an ID-based 1-out-of-n-users ring signature scheme can be extended to an ID-based 1-out-of-n-groups ring signature scheme. In most existing ID-based schemes, n users' public keys can be added together by simple point addition, while the corresponding private key can also be obtained by simple point addition. Hence if we use these aggregated public keys (from n users' keys) instead of a normal public key (from only one user's key), it is possible for us to have a 1-out-of-n-groups ring signature scheme. Keep in mind that trivial solution may not render a secure scheme. In the following scheme, commitments made by each participating signer are aggregated together to generate a single challenge for all the participating signers; otherwise, it may be possible for a single signer to cancel out the terms related to the other signers, in the way some of the ID-based ring signatures are devised [14,22].

Construction. Define the access structure $\mathcal{U}$ as $\{\mathcal{U}_1, \mathcal{U}_2, \cdots \mathcal{U}_d\}$ (where $\mathcal{U}_i$ denotes a set of signers) and all the members of a particular set in $\mathcal{U}$ (says $\mathcal{U}_k$, where $1 \leq k \leq d$) participate in the signing. The signature can convince anyone that all the members of a certain group in $\mathcal{U}$ have cooperated to give the signature, but it does not show which group is signing.

`Sign`: Let $\mathcal{U}_k = \{ID_1, ID_2, \cdots, ID_{n_k}\}$ be the set of all identities of n_k users. They choose an access structure $\mathcal{U}$ defined as $\{\mathcal{U}_1, \mathcal{U}_2, \cdots \mathcal{U}_d\}$ where $\mathcal{U}_k \in \mathcal{U}$. The ID-based ring signature for the access structure $\mathcal{U}$ can be generated as follows.

1. Compute $Y_i = \sum_{ID_j \in \mathcal{U}_i} (Q_{ID_j})$, $\forall i \in \{1, 2, \cdots, d\}$.
2. Choose $U_i \in_R \mathbb{G}_1 *$, and $i = H_2(m||\mathcal{U}||U_i)$ $\forall i \in \{1, 2, \cdots, d\} \backslash \{k\}$.
3. Each signer $ID_{s_k} \in \mathcal{U}_k$ chooses $r'_{s_k} \in_R \mathbb{Z}_q^*$ and computes $U_{s_k} = r'_{s_k} Q_{ID_{s_k}}$, $\forall k \in \{1, 2, \cdots, n_k\}$.
4. Any particular signer who has knowledge of $\bigcup_{s_k=1}^{n_k} \{U_{s_k}\}$ computes $U_k = \sum_{s_k=1}^{n_k} (U_{s_k}) - \sum_{i \neq k} \{U_i + c_i Y_i\}$ and $c_k = H_2(m||\mathcal{U}||U_k)$.

5. Each signer $ID_{s_k} \in \mathcal{U}_k$ computes $V_{s_k} = (c_k + {r'}_{s_k}) S_{ID_{s_k}}$.
6. Output the signature for m and $\mathcal{U}$ as $\sigma = \{U_1, U_2, \cdots, U_d, V = \sum_{ID_{s_k} \in \mathcal{U}_k} (V_{s_k})\}$.

`Verify`: A verifier can check the validity of a signature $\sigma = \{\bigcup_{i=1}^{d} \{U_i\}, V\}$ for the message m and the access structure $\mathcal{U}$ as follows.

1. Compute $c_i = H_2(m||\mathcal{U}||U_i)\ \forall i \in \{1, 2, \cdots, d\}$.
2. Check whether $\hat{e}(P, V) = \hat{e}\{P_{pub}, \sum_{i=1}^{d} [U_i + c_i \sum_{ID_j \in \mathcal{U}_i} (Q_{ID_j})]\}$.
3. Accept the signature if it is true, reject otherwise.

5.3 Signcryption

To achieve the signer-ambiguity and the message confidentiality at the same time, ID-based ring signcryption were proposed [12,24]. The idea behind their schemes is that an "encrypted" value is used as one of the inputs of the hash function, which makes only the designated decryptor can verify the signature. Basically, both [12,24]'s construction is the integration of ID-based ring signature and ID-based encryption [4]. ([12] employs Zhang and Kim's scheme [48] while [24] employs Herranz and Sáez 's one [22].) Due to the space constraint, please refer to [12,24] for the detailed steps of combinations.

6 Supporting Protocol

If an adversary can gain knowledge on which "identities" have requested the corresponding private keys, then the anonymity provided by an ID-based ring signature is greatly affected if a signer gives out a ring signature without checking whether "other signers" have requested for their private key. An anonymous ID-based key-issuing protocol can help in solving this problem. Here we review the separable and anonymous ID-based key issuing protocol proposed in [39].

`Setup`: Before the execution of the protocol, the user ID requesting for the private key chose a password pw during off-line authentication. The tuple (ID, pw) is stored in PKG's database of "pending key".

`Extract`:

1. ID selects a random number $r \in_R \mathbb{Z}_q^*$.
2. $ID \rightarrow PKG$: $\{Q = rH_1(ID), T = r^{-1} H_1(pw)\}$.
3. PKG checks the validity of the request by checking if $\hat{e}(H_1(ID), H_1(pw)) = \hat{e}(Q, T)$ holds for a certain tuple in the database (instead of storing ID and pw, PKG can only store the value of $\hat{e}(H_1(ID), H_1(pw))$). If so, continues.
4. $PKG \rightarrow ID$: $\{S = sQ\}$.
5. User ID: verifies the blinded private key by the equality $\hat{e}(S, P) = \hat{e}(Q, P_{pub})$. If it holds, unblinds the encrypted key and obtains $sH_1(ID)$.

`Housekeeping`: After the execution of the above protocol, the user can delete pw after obtained the private key. The PKG can also remove the tuple (ID, pw) from the database, so the database is only holding the tuples corresponding to "private key to be issued". It will not grow to the gigantic size of the certificate repository of traditional certificate based system.

A Brief Security Analysis. Although the blinding process (from the scalar multiplication by r) cannot serve as a semantically secure encryption against adaptive chosen ciphertext attack, the "encryption key" r is used once only. So even in the case some partial information has leaked, it cannot help in another invocation of the protocol. On the other hand, it is not possible for user to request for any private key which does not correspond to his or her identity by the validity check of PKG in Step 3 of the protocol.

7 State-of-the-Art and Open Problems

In [17], a constant-size ring signature was derived from the proposed anonymous identification scheme. Their scheme uses a public key which is prime, so an extension supporting ID-based keys seems to be non-trivial. Recently, the first ID-based ring signature scheme with constant-size signatures has been proposed in [31]. Both of [17] and [31] achieve constant-size signatures with the help of a cryptographic primitive known as an accumulator. Basically, an accumulator can accumulate a set $\mathbb{X}$ of values into a unique and small value z such that a proof that x has been accumulated within z can only be made for elements $x \in \mathbb{X}$ (refer to [18] and [31] for a more formal definition of the accumulator).

As [31] is an accumulator-based construction, the scheme's structure is quite different from that of previous ID-based ring signature schemes [14,22,26,48], including the design principles behind it (basically it consists of two proofs of knowledge, one is about the private key and the other is about the witness that the corresponding public key is accumulated in the accumulator), the form of the private key (different from the description in Section 2.2), etc. So its description is omitted from this survey for the sake of uniformity. We refer the interested reader to [47,31] for a review of the scheme's construction and its security. The scheme is also quite efficient in the sense that only three pairing operations are needed for verification (with pre-computation of the result) while the most efficient scheme [14] needs two of them. However, this constant-size scheme requires a rather strong cryptographic assumption: the q-strong Diffie-Hellman assumption (refer to [31] for details). To support a maximum of q users in the ring signature, the user needs to obtain (once for all signatures) a rather large system parameter which includes $(q+1)$ $\mathbb{G}_1$ elements. One possible research direction is to break this requirement.

Other possible research directions include incorporating the special properties of some PKI-based ring signature schemes into ID-based ring signature schemes. One example is separability. For t-out-of-n case, Chow *et al.*'s threshold ring signature scheme [13] has a certain level of separability, as the signature can

include users who joined different PKGs (yet these PKGs should use the same parameters for the curve). For 1-out-of-n case, as Abe *et al.*'s generic construction of separable ring signature scheme [1] can be instantiated by any three-move based schemes and trapdoor-one-way based schemes, Zhang and Kim's scheme [48] can be easily extended to a separable version. However, a similar technique cannot be applied to more efficient constructions like Herranz and Sáez 's one [22] or Chow *et al.*'s one [14]. The reason behind is that their constructions require group operations to be performed directly on the public key of each user and the public parameters of each user's PKG.

Another one is linkability. It appears that the existing technique of adding linkability to ring signature [29,41] cannot be applied trivially on the existing ID-based ring signature; as the technique relies on the intractability of the decisional Diffie-Hellman problem (which is easy for groups equipped with bilinear pairings). It is also interesting to find other applications of ring signature by exploiting the advantages of ID-based ring signature schemes.

Acknowledgement

This research is supported in part by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-01/99), two grants from the Research Grants Council of the HKSAR, China (Project No. HKU/7144/03E and HKU/7136/04E), and two grants from the Innovation and Technology Commission of the HKSAR, China (Project No. ITS/170/01 and UIM/145).

References

1. Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n Signatures from a Variety of Keys. *Advances in Cryptology - AsiaCrypt 2002, LNCS 2501*, pp. 415–432.
2. Amit K Awasthi and Sunder Lal. ID-based Ring Signature and Proxy Ring Signature Schemes from Bilinear Pairings. Cryptology ePrint Archive, 2004/184.
3. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security Proofs for Identity-based Identification and Signature Schemes. *Advances in Cryptology - EuroCrypt 2004, LNCS 3027*, pp. 268–286.
4. Dan Boneh and Matt Franklin. Identity-based Encryption from the Weil Pairing. *Advances in Cryptology - Crypto 2001, LNCS 2139*, pp. 213–229.
5. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. *Advances in Cryptology - EuroCrypt 2003, LNCS 2656*, pp. 416–432.
6. Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Advances in Cryptology - AsiaCrypt 2001, LNCS 2248*, pp. 514–532.
7. Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold Ring Signatures and Applications to Ad-hoc Groups. *Advances in Cryptology - Crypto 2002, LNCS 2442*, pp. 465–480.
8. Jae Choon Cha and Jung Hee Cheon. An Identity-based Signature from Gap Diffie-Hellman Groups . *Public Key Cryptography - PKC 2003, LNCS 2567*, pp. 18-30.

9. Tony K. Chan, Karyin Fung, Joseph K. Liu, and Victor K. Wei. Blind Spontaneous Anonymous Group Signatures for Ad Hoc Groups. In *Security in Ad-hoc and Sensor Networks, First European Workshop, ESAS 2004, LNCS 3313*.
10. David Chaum and Eugène van Heyst. Group Signatures. *Advances in Cryptology - EuroCrypt '91, LNCS 547*, pp. 257–265.
11. Xiaofeng Chen, Fangguo Zhang, and Kwangjo Kim. A New ID-based Group Signature Scheme from Bilinear Pairings. Cryptology ePrint Archive, 2003/116.
12. Tianjie Cao, Dongdai Lin and Rui Xue. ID-based Ring Authenticated Encryption. *Advanced Information Networking and Applications - AINA 2005*, pp. 591-596.
13. Sherman S.M. Chow, Lucas C.K. Hui, and S.M. Yiu. Identity Based Threshold Ring Signature. *International Conference on Information Security and Cryptology - ICISC 2004, LNCS 3506*, pp. 218–232.
14. Sherman S.M. Chow, S.M. Yiu, and Lucas C.K. Hui. Efficient Identity Based Ring Signature. *Applied Cryptography and Network Security - ACNS 2005, LNCS 3531*.
15. Sherman S.M. Chow, S.M. Yiu, Lucas C.K. Hui, and K.P. Chow. Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. *International Conference on Information Security and Cryptology - ICISC 2003, LNCS 2971*, pp. 352-369.
16. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. *Advances in Cryptology - Crypto '94, LNCS 839*, pp. 174–187.
17. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous Identification in Ad Hoc Groups. *Advances in Cryptology - EuroCrypt 2004 LNCS 3027*, pp. 609–626.
18. Nelly Fazio and Antonio Nicolosi. Cryptographic Accumulators: Definitions, Constructions and Applications, 2003. Manuscript.
19. Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *Advances in Cryptology - Crypto '86, LNCS 263*, pp. 186–194.
20. Chong Zhi Gao, Zheng an Yao, and Lei Li. A Ring Signature Scheme Based on the Nyberg-Rueppel Signature Scheme. *Applied Cryptography and Network Security - ACNS 2003, LNCS 2846*, pp. 169–175.
21. Javier Herranz. A Formal Proof of Security of Zhang and Kim's ID-based Ring Signature Scheme. *International Workshop on Security In Information Systems, WOSIS 2004, in conjunction with ICEIS*, pp. 63–72.
22. Javier Herranz and Germán Sáez. New Identity-based Ring Signature Schemes. *International Conference on Information and Communications Security - ICICS 2004, LNCS 3269*, pp. 27–39.
23. Florian Hess. Efficient Identity Based Signature Schemes based on Pairings. *Selected Areas in Cryptography - SAC 2002, LNCS 2595*, pp. 310–324.
24. Xinyi Huang, Willy Susilo, Yi Mu and Futai Zhang. Identity-based Ring Signcryption Schemes: Cryptographic Primitives for Preserving Privacy and Authenticity in The Ubiquitous World. *Advanced Information Networking and Applications - AINA 2005*, pp. 649-654.
25. Fabien Laguillaumie and Damien Vergnaud. Multi-designated Verifiers Signatures. *International Conference on Information and Communications Security - ICICS 2004, LNCS 3269*, pp. 495–507.
26. Chih-Yin Lin and Tzong-Chen Wu. An Identity-based Ring Signature Scheme from Bilinear Pairings. *Advanced Information Networking and Applications - AINA 2004*, pp. 182-185. Also appear in Cryptology ePrint Archive, 2003/117.

27. Joseph K. Liu. *Spontaneous Anonymous Group Cryptography*. PhD thesis, The Chinese University of Hong Kong, 2004.
28. Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. A Separable Threshold Ring Signature Scheme. *International Conference on Information Security and Cryptology - ICISC 2003, LNCS 2971*, pp. 12–26.
29. Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups (Extended Abstract). In *Australasian Conference on Information Security and Privacy - ACISP 2004, LNCS 3108*, pp. 325–335.
30. Joseph K. Liu and Duncan S. Wong. On the Security Models of (Threshold) Ring Signature Schemes. *International Conference on Information Security and Cryptology - ICISC 2004, LNCS 3506*.
31. Lan Nguyen. Accumulators from Bilinear Pairings and Applications. *Topics in Cryptology - CT-RSA 2005, LNCS 3376*, pp. 275–292. Revised full version available at Cryptology ePrint Archive, 2005/123.
32. Kaisa Nyberg and Rainer A. Rueppel. Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem. *Advances in Cryptology - EuroCrypt '94, LNCS 950*, pp. 182–193.
33. Kenneth G. Paterson. ID-based Signatures from Pairings on Elliptic Curves. Cryptology ePrint Archive, 2002/004.
34. M. Rabin. Digitalized Signatures as Intractable as Factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979.
35. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 26(1):96–99, January 1983.
36. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. *Advances in Cryptology - AsiaCrypt 2001, LNCS 2248*, pp. 552–565.
37. Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 4(3):161–174, 1991.
38. Adi Shamir. Identity-based Cryptosystems and Signature Schemes. *Advances in Cryptology - Crypto 1984, LNCS 196*, pp. 47–53.
39. Ai-fen Sui, Sherman S.M. Chow, Lucas C.K. Hui, S.M. Yiu, K.P. Chow, W.W. Tsang, C.F. Chong, K.H. Pun, and H.W. Chan. Separable and Anonymous Identity-based Key Issuing. *Security in Networks and Distributed Systems - SNDS 2005, in conjunction with International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE Computer Society.
40. Willy Susilo and Yi Mu. Non-Interactive Deniable Ring Authentication. *International Conference on Information Security and Cryptology - ICISC 2003, LNCS 2971*, pp. 386–401.
41. Patrick P. Tsang, Victor K. Wei, Tony K. Chan, Man Ho Au, Joseph K. Liu, and Duncan S. Wong. Separable Linkable Threshold Ring Signatures. *Progress in Cryptology - IndoCrypt 2004, LNCS 3348*, pp. 384–398.
42. Victor K. Wei. A Bilinear Spontaneous Anonymous Threshold Signature for Ad Hoc Groups. Cryptology ePrint Archive, 2004/039.
43. Duncan S. Wong, Karyin Fung, Joseph K. Liu, and Victor K. Wei. On the RS-Code Construction of Ring Signature Schemes and a Threshold Setting of RST. *International Conference on Information and Communications Security - ICICS 2004, LNCS 2836*, pp. 34–46.

44. Jing Xu, Zhenfeng Zhang, and Dengguo Feng. A Ring Signature Scheme Using Bilinear Pairings. *Workshop on Information Security Applications - WISA 2004, LNCS 3325*, pp. 163–172.
45. Hyo Jin Yoon, Jung Hee Cheon, and Yongdae Kim. Batch Verifications with ID-based Signatures. *International Conference on Information Security and Cryptology - ICISC 2004, LNCS 3506*
46. Tsz Hon Yuen and Victor K. Wei. Fast and Proven Secure Blind Identity-based Signcryption from Pairings. *Topics in Cryptology - CT-RSA 2005, LNCS 3376*, pp. 305-322.
47. Fangguo Zhang and Xiaofeng Chen. Cryptanalysis and Improvement of an ID-based Ad-hoc Anonymous Identification Scheme at CT-RSA 05. Cryptology ePrint Archive, 2005/103.
48. Fangguo Zhang and Kwangjo Kim. ID-based Blind Signature and Ring Signature from Pairings. *Advances in Cryptology - AsiaCrypt 2002, LNCS 2501*, pp. 533–547.
49. Fangguo Zhang, Reihaneh Safavi-Naini, and Chih-Yin Lin. New Proxy Signature, Proxy Blind Signature and Proxy Ring Signature Schemes from Bilinear Pairings. Cryptology ePrint Archive, 2003/104.
50. Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. *Public Key Cryptography - PKC 2004, LNCS 2947*, pp. 277–290.

Development of a Flexible PERMIS Authorisation Module for Shibboleth and Apache Server

Wensheng Xu, David W. Chadwick, and Sassa Otenko

Computing Laboratory, University of Kent, Canterbury, England, CT2 7NZ
{w.xu, d.w.chadwick, o.otenko}@kent.ac.uk

Abstract. This paper describes the development of a flexible Role Based Access Control (RBAC) authorisation module – the Shibboleth and Apache Authorisation Module (SAAM) which is based on the PERMIS privilege management infrastructure. It explains how the module can work with the Apache web server, with or without Shibboleth. We argue that this can effectively improve the level of trust and flexibility of access control for the Shibboleth architecture and the Apache web server, as well as provide a finer grained level of control over web resources.

1 Introduction

Shibboleth [1] is a cross-institutional authentication and authorisation architecture for single sign on and access control over web resources. It is specified by the Internet2 middleware architecture committee and many universities in the USA and Europe have started to build experimental services based on it. Shibboleth can allow distributed users belonging to different institutions to share web resources conveniently and safely while respecting the users' privacy. What makes the Shibboleth architecture especially attractive is that authentication of a user is carried out by the home site (i.e. where the user originates from) whilst authorisation for a user to access specific web resources is carried out by the resource website. Such separation of authentication and authorisation functions eases the creation and management of federations of resource providers and users.

Shibboleth defines a protocol for carrying authentication information and user attributes from the user's home site to the resource site. The resource site can then use the user attributes to make the access control decision about the user's request. A user only needs to be authenticated once by the home site in order to visit other Shibboleth protected resource sites in the federation, as the resulting authentication token is recognised by any member of the federation. In addition to this, protection of the user's privacy can be achieved, since the user is able to restrict what information about him will be released to the resource providers from the user's home site.

Shibboleth's functionality is achieved by a simple trust relationship between the resource site and the user's home site. To put it simply, the resource site trusts the origin site to authenticate the user and to provide the correct set of attributes for the user, and the home site trusts the resource site to give access to users with the correct set of attributes. If a finer grained trust relationship is required to allow for distributed management of user attributes and dynamic delegation of authority, then a more

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 162 – 179, 2005.

sophisticated authorisation infrastructure than that provided by Shibboleth is required. For example, if the resource site trusts specific managers/authorities to allocate specific attributes to different groups of users, this cannot be conveyed via Shibboleth since there is a single attribute authority (AA) at each home site. Furthermore, the security of the source of the user attributes at the home site might be of concern to the resource site, both because of how the attributes are stored, and because of the user's dynamic pseudonymity[1]. Finally, the access control decision making based on these attributes is simplistic in its functionality, and the management of the access controls is mixed together with web server administration at the resource site. Therefore the flexibility of setting the access control policy is adversely affected.

These limitations in Shibboleth can be alleviated by integrating a policy controlled Privilege Management Infrastructure (PMI) into it. PMIs are described in the 2001 edition of X.509 [3]. PERMIS [2] is an implementation of an X.509 PMI, and uses the Role Based Access Control (RBAC) [10] paradigm. PERMIS is built in accordance with the ISO 10181-3 standard [15] and incorporates a sophisticated policy controlled application independent RBAC decision engine, or policy decision point (PDP), in its software suite. Roles are stored in X.509 attribute certificates (ACs), and since these are digitally signed for integrity protection, it can support the distributed management of roles between multiple AAs. Other experimental RBAC implementations have been developed, for example by Ferraiolo et al [12] and Sandhu et al [13, 14], but PERMIS is the first one to use X.509 ACs to store roles. PERMIS has already been successfully applied in several applications, and more recently has been integrated with the Globus Toolkit [4]. By developing and integrating a RBAC authorisation module into Shibboleth – the PERMIS SAAM (Shibboleth-Apache Authorisation Module) – a highly improved authorisation capability can be achieved for distributed web resource access control.

The rest of this paper is organised as follows. Section 2 describes Shibboleth, and lists its main features and limitations. Section 3 analyses how Apache [7] authentication and authorisation works, how Shibboleth interacts with this, and the approach that needs to be taken to replace Shibboleth authorisation by PERMIS authorisation. Section 4 presents the PERMIS SAAM system structure and the Apache directives that control it. Section 5 describes the interactions between the PERMIS SAAM and Shibboleth. Section 6 describes how the PERMIS SAAM can be integrated with the Apache web server without Shibboleth. Finally Section 7 gives the conclusions.

2 Main Features and Limitations of Shibboleth

As a middleware layer Shibboleth uses SAMLv1.1 [5] for encoding some of its messages. When a user contacts a Shibboleth-protected resource site (target site) with the browser, requesting access to a particular URL, the user is required by the Shibboleth

[1] Dynamic pseudonymity, provided by Shibboleth, allows the user to have a different pseudonym each time she contacts the resource site. Whilst this provides better user privacy (the user cannot be profiled by the resource site), it reduces the strength of the association between the user and her attributes. Furthermore, if multiple attribute authorities (AAs) issue attributes to the user, it will be difficult to facilitate that all of them dynamically re-issue the attributes each time the user's identity changes.

Indexical Reference Establisher[2] (SHIRE) to go to a WAYF (Where Are You From) site to pick his/her home site (origin site), and their browser is redirected to their home site's authentication server where the user is invited to log in. After the user is authenticated by the origin site, the browser is redirected back to the target site along with a handle package which includes an assertion that this user has been successfully authenticated by a particular means (e.g. username/password, Kerberos or digital signature), a unique handle generated by the Handle Service for the user (the user's pseudonym), and the Attribute Authority (AA) location at the origin site where the user's attributes may be obtained from. Then the Shibboleth Attribute Requester (SHAR) at the resource site returns the handle to the AA of the origin site and gets a set of attributes of the user from the AA. The messages between the target site and origin site are encoded in SAML and are embedded as a browser cookie, so the user observes only redirections between the sites.

The user attributes are then passed to the Shibboleth authorisation function - ShibAuthz, which will make an access control decision based on these attributes. The SHIRE, SHAR and ShibAuthz are all included in the Shibboleth Apache module called *mod_shib*[3]. The web server will then give a response back to the user browser based on the decision result. The whole Shibboleth authentication and authorisation process is shown in Fig. 1.

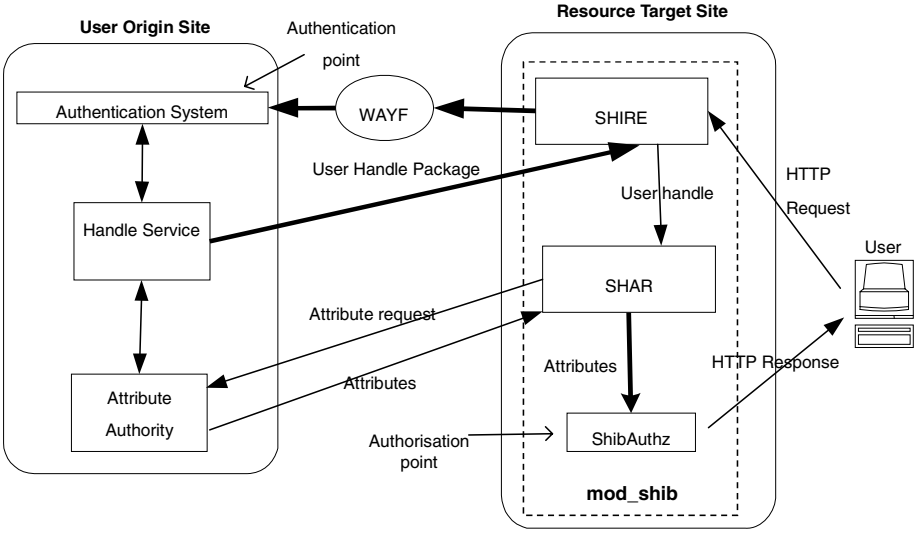

Fig. 1. The Shibboleth Authentication and Authorisation Process

[2] In this article we refer to Shibboleth version 1.2 and its related documentation. At the time of writing Shibboleth architecture undergoes significant changes, whereby some of the components of the system will be regrouped and renamed.

[3] This is correct from the functional perspective. In the Shibboleth implementation however, the SHAR's actual function is implemented by an independent module which communicates with mod_shib by internal socket communications.

Because user authentication and authorisation are taking place at different locations, namely at the origin site and the target site respectively, Shibboleth allows for a different pseudonym (the handle) for the user's identity to be invented by the origin site every time. Both the origin site and the user can have control over the release of the user's attributes, so the user's privacy can be well protected. On the other hand, because authentication and authorisation are performed by different sites and the user's name is not provided to the target site for privacy reasons, the target site's access control is only based on the user's attributes without the need to know who issued them, whether they are still valid, or whether they are even the correct attributes for the particular user, so the safety of the target site heavily relies on trusting the origin site to return the correct attributes.

The messages carrying these attributes are digitally signed by the SAML authority at the origin site, so the security of these messages is ensured, but the security of the source of the attributes is not guaranteed. In many sites a back end LDAP [11] server is the authoritative source for both authentication and attribute information. These attributes in the LDAP server are not digitally signed, so it is relatively easy for these attributes to be tampered with compared to for example digitally signed X.509 attribute certificates (c.f tampering with passwords compared to tampering with X.509 public key certificates). Furthermore Shibboleth doesn't cater for multiple attribute authorities at the home site. There is only one AA that creates the cryptographically protected SAML tokens. The AA must query the attribute repository (e.g. LDAP server) to collect the user's attributes that are typically stored there as plain text. Even though the repository can be managed by multiple administrators, we would like to argue that this may not be secure enough, as it is difficult to ensure that an administrator does not exceed their authority. Because the security of the attributes in the repository is essential to the whole Shibboleth system, origin sites typically have a single administrator centrally managing the attributes of the users. This reduces the flexibility of the attribute assignments and inhibits the distributed/devolved management of them.

Another limitation of the Shibboleth infrastructure is that it provides only a basic access control decision making capability. The authorisation decision made by the target site is based on the attributes received from the origin site, and the access rules that are defined by Apache directives in the Apache configuration file. The directives can only express basic simple access control rules based on regular expressions, for example "users with attribute 'staff' can have access to location A" or "users with attribute 'senior member' can have access to location B", but it can not express conditional rules (e.g. access if time is between 9am and 5pm), complicated rules (e.g. ones with multiple conditions based on the parameters of the user's request) or RBAC features such as separation of duties or role hierarchies. This is acceptable for simple applications, but for advanced applications this is a weakness for resource site administrators.

Because the basic access control rules are defined in the Apache configuration file and the authorisation function is carried out by the Shibboleth Apache module, then every time the target site needs to change its access rules, it needs to redefine the directives in the Apache configuration file and restart the Apache server. This means that the administrator of the Apache web-server has to manage the access control rules, and there is no way for the owner of the resources to directly specify the access control rules without going via the Apache administrator. This limits the resource owner's flexibility for management of access control over his resources.

From the above discussion we can see that the Shibboleth target site makes access authorisation decisions after it receives user attributes from the Shibboleth origin site. We want to improve Shibboleth so that:

(1) If the attributes at the origin site are digitally signed by a relevant AA, then the trust between the origin and target sites no longer solely relies on the security of the origin attribute repository, e.g. LDAP server, as the attribute certificates (ACs) are tamper-proof themselves. Consequently the security level for the whole system can be effectively improved; for example, X.509 ACs can be adopted as user attributes and they can be stored in the origin site's LDAP repository and be released to the target site for access control decision making;
(2) When the target site receives the attributes or ACs from the origin site, at this point a sophisticated RBAC decision engine (e.g. PERMIS) can be used to make access control decisions instead of Shibboleth's own simple authorisation function. This will help to implement sophisticated access control features such as separation of duties and role hierarchies, so a finer grained and more refined access control mechanism can be deployed at the target site.

Once Shibboleth and PERMIS are integrated, both the security and flexibility of the Shibboleth infrastructure can be effectively improved. An in-depth discussion of the trust models and different approaches to the integration of Shibboleth and PERMIS can be seen in [9]. But for any of these models, two common problems need to be solved for integrating Shibboleth and PERMIS. Firstly, how can PERMIS replace Shibboleth's original authorisation functionality and make decisions based on attributes without the need to modify Shibboleth at the source code level. Secondly, how can ACs replace attributes and be stored and transferred by Shibboleth. The first question is addressed below and the second question is discussed in Section 5.

3 Analysis of Apache and Shibboleth Authentication and Authorisation Functions

3.1 How Apache Performs Authentication and Authorisation

Apache [7] is a popular open-source HTTP server that is widely used in universities and institutions to provide a web resource sharing service. How does the Apache server handle HTTP requests? The Apache server breaks down HTTP request handling into a series of processing phases, including:

- URI to Filename translation;
- Authentication identity check: to check who the user is;
- Authorisation access check: to check if the user is authorized here;
- Module-specific access checks: to check if there is any restriction from this module upon the requested resource;
- Sending a response back to the user;
- Other phases, unrelated to this article.

The basic functionality of the Apache web-server can be extended by adding so-called *modules* to it, and each module can handle one or several processing phases. (The

authorisation phase at a Shibboleth target site is handled by one such Apache module - *mod_shib*.) When an Apache module is written, it must contain code to register specific handlers (functions) that are to be called for specific phases. Each of the above phases is processed sequentially by the Apache server and each registered handler (handling function) is called once for each phase, unless a preceding module completes the phase processing (see later). In Apache 1.3, the order in which the modules are called is fixed for all the phases, and depends upon the order in which the module is loaded. In Apache 2.0, a certain amount of flexibility has been introduced into the calling order, as each module can indicate its priority (FIRST, MIDDLE, LAST) for each phase. If two or more Apache modules have handling functions for the same processing phase, then these handling functions will be executed one after another according the order in which they were loaded. A simplified HTTP request handling process in the Apache server 1.3 is illustrated in Fig. 2.

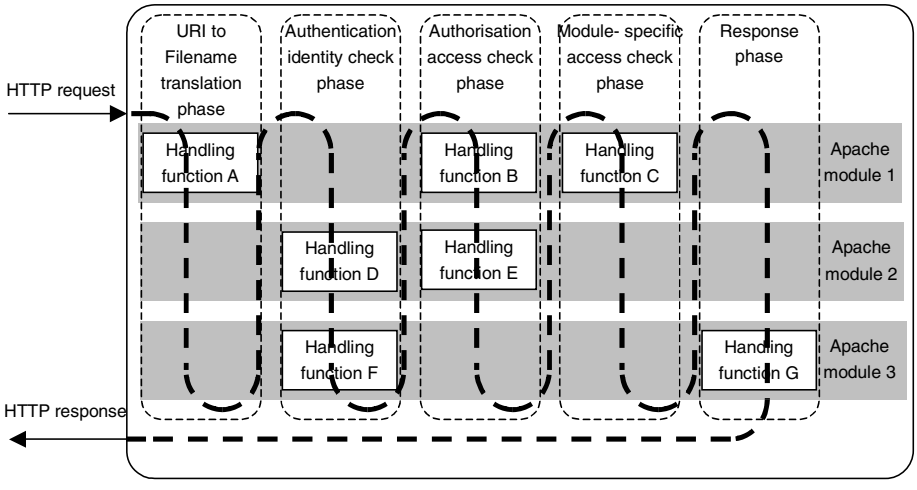

Fig. 2. HTTP Request Handling Process in the Apache Server 1.3

For the authentication and authorisation access check phases in the HTTP request handling process, the Apache server looks at a succession of Apache modules in sequence to match and invoke the corresponding handling functions. If a relevant module handling function is invoked by the Apache server for a phase, there may be three possible results:

- If the request is handled successfully, a magic integer constant OK will be returned to the Apache server and the subsequent Apache modules will not be invoked in this phase;
- If the module handling function finds an error in the user's request for whatever reason, one of the HTTP error codes will be returned, such as FORBIDDEN or HTTP_UNAUTHORIZED. This will also terminate the handling of the request, only this time the subsequent Apache modules will not be

invoked for this phase or subsequent phases, and the user will be informed of the error by the browser according to the HTTP error code;
- If the module handling function declines to handle this phase, then the magic integer constant DECLINED will be returned to the Apache server. In this case the Apache server will continue to look at the rest of the modules in order to find a handling function to serve this phase. Usually DECLINED is returned by modules when the request is not applicable, like in cases when the requested location is not protected by the module (e.g. the AuthType directive is missing or specifies a type of authorisation that is not supported by this module).

The first two results above are "definite", i.e. there can be no other opinion about the request. The third result is "indefinite" and means that the handling function of a module cannot make a decision. So only when the preceding Apache module handling function returns the constant DECLINED, can the subsequent Apache modules be invoked for this phase, otherwise the rest of the Apache modules are skipped as if they didn't exist.

3.2 Shibboleth Integration with Apache

The Shibboleth Apache module is called *mod_shib,* and it provides the access control service and single sign-on capabilities. Mod_shib is invoked at the target site during two phases of the Apache HTTP request handling process: the Authentication phase and the Authorisation phase. The SHIRE and SHAR are invoked during the Authentication phase and ShibAuthz is invoked during the Authorisation phase. During the Authentication phase the SHIRE redirects the user's browser to the user's home site for authentication if it is the first time for the user to access a federated target site in this session. Both the Shibboleth origin and target sites are issued with X.509 public key certificates and these certificates are configured into mod_shib. After a user is authenticated at the origin site, a digitally signed handle package is sent back to the SHIRE at the target site. The SHIRE checks the signature on the User Handle Package to validate that the handle package is really coming from a trusted origin site. In this way, the Shibboleth target site trusts that the user has been reliably authenticated. The SHAR then collects attributes of the user from the AA at the origin site via the attribute query communication. On subsequent access requests in the same session, the SHIRE and SHAR simply check the user's cookies and retrieve the attributes from there. In the Authorisation phase ShibAuthz is invoked to make the access control decisions based on the attributes of the user.

Our aim is to allow mod_shib at the target site to perform normal Shibboleth authentication and attribute collection in the Authentication phase, but to override its authorisation mechanism in the Authorisation phase with our PERMIS RBAC policy-controlled PDP instead. So the design of the PERMIS authorisation module is straightforward. It should be invoked before the mod_shib authorisation code in the Authorisation phase, and obtain the attributes that Shibboleth has already retrieved from the AA in order to make a decision in accordance with the PERMIS authorisation policy.

4 System Structure of the PERMIS SAAM

Based on the above analysis, we developed the PERMIS SAAM authorisation module to work in conjunction with Shibboleth to provide a generic authorisation function based on RBAC and the PERMIS Privilege Management Infrastructure. The SAAM works as an Apache module and provides an authorisation handling function called during the Apache authorisation phase. By proper construction of the Apache configuration file, SAAM can be loaded and registered before the Shibboleth module, and can take the responsibility for making authorisation decisions thereby bypassing the Shibboleth authorisation function, without disturbing the rest of the functionalities of Shibboleth.

4.1 Functions of PERMIS

The PERMIS infrastructure comprises a privilege allocation (PA) component, a privilege verification (PV) component, a policy decision point (PDP) and a policy management GUI. The PERMIS PA component is responsible for allocating privileges to users in the shape of roles stored in X.509 attribute certificates (ACs). The PA component may be distributed and used by many managers to give roles to their subordinates. The role ACs are then stored in one or more LDAP directories for subsequent use by the PV component. After a user is authenticated, the PERMIS PV component can access these LDAP directories to retrieve the role ACs for the user (the *pull* mode of operation). Alternatively, the ACs can be given to the PV component by the caller for instant validation (the *push* mode of operation).

The PERMIS infrastructure is driven by a PERMIS policy that comprises a Role Allocation Policy (RAP) and a Target Access Policy (TAP) (see later). This may be created using the policy management GUI.

The role ACs are verified against the RAP by the PV component and all valid roles/attributes are passed to the PDP. The PDP then makes its access control decision for the user's request based on the TAP and the valid attributes. The PDP returns a granted or denied response to the caller according to the policy in force at that time.

In the integration of Shibboleth and PERMIS, authentication is carried out by the Shibboleth system. Shibboleth is responsible for providing PERMIS with the user name as it appears in the X.509 Attribute Certificates. Shibboleth may push the X.509 ACs into PERMIS, otherwise PERMIS may pull them from LDAP directories.

Note that Shibboleth has to provide the name of the user (as held in the X.509 ACs). Whilst this may decrease the user's privacy somewhat, it does not have to seriously undermine it, as the name used by the system does not have to be the user's real name. To maintain user privacy, which is a core consideration in Shibboleth, pseudonyms can be adopted as holder names in X.509 ACs, just like pseudonyms are adopted as user names in Shibboleth. The X.509 pseudonym can be a distinguished name string, or it can be the hash of the user's public key (although this requires the user to be PKI enabled, which many are not today). The main difference between the Shibboleth and X.509 pseudonyms is that the former ones are dynamic whilst the latter ones are static, which means that the target site can still build up a profile of the static pseudonymous user. If even this is too sensitive, then the PERMIS SAAM can adopt the simple Shibboleth trust model and transfer (unprotected) attributes attached to an anonymous

handle, in which case X.509 ACs are not needed. In this scenario we would use the PERMIS PDP as a substitute for the original Shibboleth access control decision-making functionality, in order to benefit from its superior decision making functionality without sacrificing any of Shibboleth's privacy protection features, but conversely, we do not take advantage of the distributed role management functionality that X.509 ACs provide. Ultimately, the quality of user privacy can be determined by the origin site/application, but it is a trade off with the (loss of) trustworthiness and flexibility in the binding between a user and his/her attributes.

4.2 PERMIS RBAC Policy

The PERMIS RBAC policy is the basis for access control of resources. It is written in XML and is kept in an X.509 Attribute Certificate, digitally signed by the Source of Authority (SoA), who is typically the resource owner. This serves the dual purpose of separating the policy specification from system administration of the Apache web-server, and makes the policy tamperproof. This policy AC is the root of trust for the access control decision making. A hierarchical RBAC model is adopted by PERMIS to specify the authorisation policy for the whole domain of resources controlled by one SoA. One PERMIS RBAC policy is able to control access to all resources in a domain by the same set of rules.

In the PERMIS RBAC policy there are two main sub-policies: the RAP and the TAP. The RAP is responsible for defining a list of trusted AAs, the set of attributes they are trusted to assign, and the groups of users they can be assigned to[4]. When the PERMIS PV component is passed a set of attribute certificates, it can retrieve the valid and trusted attributes from them according to the RAP and discard the invalid and untrusted attributes[5].

The TAP is responsible for defining the set of targets that are protected by this policy, the associated actions that can be performed on them, the attributes that a user needs in order to be granted the actions, and the restraints/conditions that apply to granting access. After the PERMIS PDP gets the attributes of the user from the PV component, then it can make access decisions for the user based on the TAP.

Beside the RAP and TAP, the PERMIS RBAC policy also includes the following sub-policy components:

- The subject sub-policy specifies the subject domains, i.e. only users from these subject domains may be authorised to access resources covered by the policy;
- The role hierarchy sub-policy specifies the different roles and their hierarchical relationships to each other;
- The Source of Authority sub-policy specifies which SoAs are trusted to allocate roles, and permits the distributed management of role allocation to take place; these are, in effect, the multiple AAs at the Origin sites who are trusted by the Target;
- The target sub-policy specifies the target domains covered by this policy;

[4] This is where the user name is used by PERMIS.

[5] Note that since the RAP is defined at the Target site, the validity and trustworthiness of the user attributes is controlled by the resource owner.

– The action sub-policy specifies the actions (or methods) supported by the targets, along with the parameters that should be passed along with each action, e.g. action *GET* with parameter *Filename*; in the Shibboleth-PERMIS integration scenario the actions should be the HTTP methods defined by RFC2616: *GET, PUT, POST, DELETE,* etc [8].

A full description of the PERMIS RBAC policy can be found in [6]. By adopting and enforcing the PERMIS RBAC policy, flexible fine grained access controls can be achieved.

4.3 Structure of the PERMIS SAAM

Based on the PERMIS infrastructure and the Shibboleth system architecture, the system structure of the PERMIS SAAM is shown in Fig. 3. All the components of SAAM are enclosed by dashed round-cornered rectangles and the rest of the components in the figure are Shibboleth. As in PERMIS, there are three sub systems in the PERMIS SAAM: the PERMIS PA sub system which is distributed to the various origin sites, and the PERMIS PV/PDP sub system and Policy Management sub system which are entirely located within the target site. The PV/PDP sub system is responsible for validating the ACs and making access control decisions, while the PA sub system at the origin site is responsible for assigning privileges to users. The Policy Management sub system at the target site is responsible for defining the RBAC policy and digitally signing it and storing it to the policy LDAP repository (denoted as

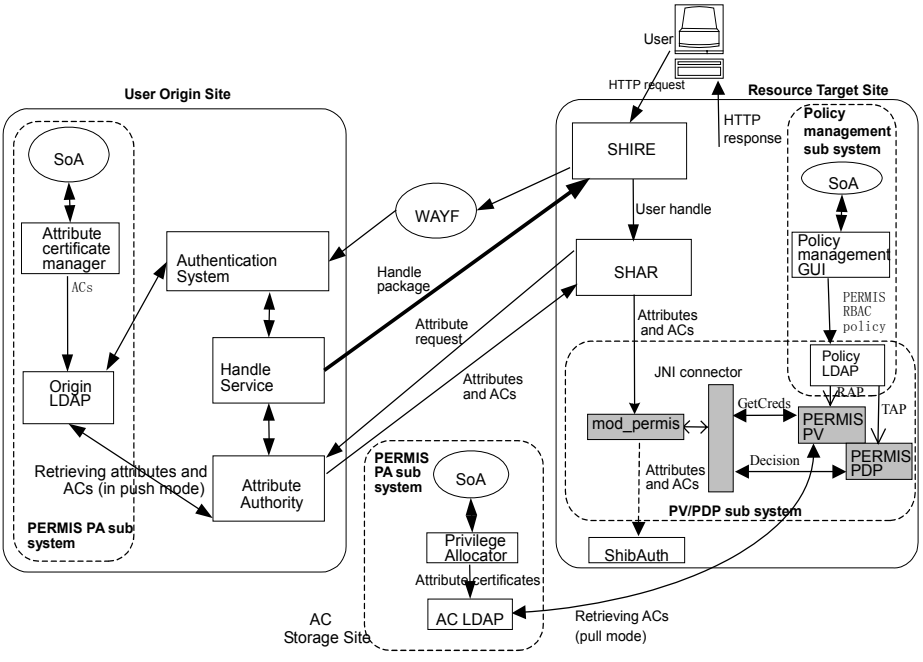

Fig. 3. Structure of Shibboleth-PERMIS SAAM Integration

"Policy LDAP" in Fig.3). If PERMIS is working in pull mode, the PERMIS PV fetches the user attribute certificates directly. Note that this requires the PV at the target site to be able to access the LDAP directories at the AC storage sites directly. Multiple AC LDAP directories are supported by SAAM in pull mode. If PERMIS is working in push mode, then Shibboleth is responsible for fetching the user attributes or ACs from the origin site and passing them to the PERMIS PV at the target site.

The PERMIS PV/PDP sub system consists of four parts: an Apache module - mod_permis which is written in C++, the PERMIS PV and PDP which are written in Java, and a Java JNI (Java Native Interface) connector which is written in C. The PERMIS PV and PDP can be called by mod_permis via the JNI connector. Mod_permis interfaces with Apache and Shibboleth to collect all the information necessary for making a decision and passing this information to the PERMIS PV and PDP. The PERMIS PDP which is based on RBAC makes a decision and passes it back to mod_permis, which translates it into "OK" or "HTTP_UNAUTHORIZED" error codes. Apache will either send the requested resource or an error information page back to the user browser depending on the result. Note that since PERMIS returns a "definite" result when the PERMIS SAAM is active, Shibboleth authorisation is not invoked. To ensure that PERMIS is called before Shibboleth authorisation, mod_permis should appear before the Shibboleth Apache module (mod_shib) in the Apache 2.0 [6] configuration file. (Since Apache 1.3 loads its modules in reverse order, mod_permis should appear *after* mod_shib in Apache 1.3.) Each location [7] in the Apache configuration file may use a different form of authorisation. The PERMIS SAAM is active only if the PermisAuthorisaton directive is present for the location (see below). If it is not present, mod_permis always returns "DECLINED", so that Shibboleth or any other configured authorisation module will be invoked in this case.

In the implementation of the integration of Shibboleth and the PERMIS SAAM, several global configuration directives are needed in the Apache configuration file (see below). Two local directives are also used for each protected location to indicate that the PERMIS SAAM is being used for this target resource, thereby bypassing the authorisation function of Shibboleth.

4.4 The PERMIS SAAM Apache Directives

The PERMIS SAAM is configured using the following directives in the Apache http.conf file:

PermisPolicyIdentifier – this holds the unique number for the PERMIS policy to be used

PermisPolicyIssuer – this holds the LDAP distinguished name of the SoA who signed the policy

PermisPolicyLocation – this contains the URL of the LDAP directory holding the policy

PermisAuthorisation – this is inserted into every <Location> that is to be controlled using PERMIS authorisation. (Note, in the absence of the PermisPullMode directive the AuthType for this <Location> must be set to Shibboleth.)

---

[6] Because mod_shib is already set as FIRST in Apache 2.0, we have no way to give mod_permis a higher precedence other than the order in which it is loaded.

[7] As indicated by the Apache <location> directive.

PermisPullMode (optional) – this is inserted into every <Location> that is to pull ACs from LDAP repositories pointed to by the PermisACLocation directives. When this directive is not present, the default mode of operation is the push mode (Shibboleth gets the Attribute Certificates from the Origin, then mod_permis pushes them to the PDP).

PermisACLocation (optional) – this contains a URL of an LDAP directory from where user ACs may be pulled (this directive may be repeated as often as required)

5 Interactions Between Shibboleth and the PERMIS SAAM

According to the different trust models adopted by the Shibboleth target and origin sites [9], the PERMIS SAAM can work in different modes with X.509 ACs - either push mode or pull mode. Furthermore, either plain attributes or X.509 ACs can be pushed to the PERMIS SAAM by Shibboleth. These are described in the following subsections.

5.1 PERMIS SAAM in Push Mode with X.509 ACs

If the origin site wishes to distribute attribute assignments to different managers, and perhaps implement dynamic delegation of authority, and the target site is willing to trust different attribute authorities at the origin site, then the origin site should store digitally signed attribute certificates in its LDAP repository (denoted as "origin LDAP" in Fig. 3). Alternatively, if either the target and/or the origin do not trust the origin site's attribute repository to securely store unsigned attributes, then the origin should assign ACs to users and store these ACs in its LDAP repository. In these cases, SAAM should work in push mode and accept ACs from Shibboleth.

In this mode of operation, one user attribute (the user's distinguished name) and all user ACs (*attributeCertificateAttribute;binary)* should be configured for release to the target by the origin AA server. The user's DN and the role ACs should be retrieved by Shibboleth and passed to the PERMIS PV/PDP for validation and making access control decisions for the user's requests. For the PERMIS PV to validate that these are the correct ACs, the user's DN should be available to match with the holder name in the ACs. The PV uses the RAP in the PERMIS policy to decide who is trusted to assign which attributes to whom. As discussed in [9], supplying ACs and user DNs in the integration of Shibboleth and PERMIS doesn't necessarily compromise a user's privacy since pseudonyms can be used as the DN and the AC holder name. On the other hand, some applications actually require the user's DN to be present in order to perform correct access controls, and so passing the user's DN in these cases is actually beneficial.

The interactions between Shibboleth and the PERMIS SAAM are as follows.

(1) When a user contacts a Shibboleth-protected resource site with the browser, requesting access to a Shibboleth-PERMIS protected URL, the user is redirected by the SHIRE to the WAYF site.
(2) After the user selects his/her home (origin) site at the WAYF site, the browser is redirected to the origin site's authentication server and the user is authenticated there.

(3) After successful authentication, the browser is redirected back to the SHIRE along with a handle package.
(4) The SHAR at the target site gets the handle and sends the handle to the AA of the origin site for attributes query.
(5) The AA retrieves the user's DN and the role ACs of the user from the origin LDAP directory, base-64 encodes the attribute certificates, and sends them back to the SHAR.
(6) The SHAR puts the attributes in the Apache HTTP headers whose names can be defined and configured in the Shibboleth Attribute Acceptance Policy (AAP). This is the last step of the authentication phase.
(7) In the authorisation phase in the HTTP request handling process, mod_permis is first invoked by the Apache server.
(8) If the location being requested by the user is not being protected by PERMIS, then mod_permis returns DECLINED and the Shibboleth authorisation function ShibAuthz will subsequently be invoked, otherwise the user's DN and role ACs are acquired by mod_permis from the HTTP headers.
(9) Mod_permis calls the PERMIS PV and PDP to make an authorisation decision, which is based on the user's DN, the role ACs, the target resource that the user is requiring, the action to the target resource (i.e. the HTTP method) and the current RBAC policy incorporating both the RAP and TAP.
(10) After the PERMIS PDP makes the granted/denied decision, the decision is returned back to mod_permis;
(11) Mod_permis returns the decision result to the Apache server, and the user can be granted or denied access to the target resource according to the decision result.

From the above interactions between Shibboleth and the PERMIS SAAM we can see that in this mode the only difference between normal Shibboleth and this integrated Shibboleth is that ACs are retrieved and passed by Shibboleth instead of plain text attributes. Since ACs are stored in the LDAP as digitally signed binary attributes and normal Shibboleth cannot retrieve binary attributes[8], Shibboleth needed to be slightly modified to handle them. On the origin side one Java class for retrieving attributes from LDAP - JNDIDirectoryDataConnector.class - was modified by us and another new Java class - Base64ValueHandler.class - was developed by the Shibboleth developers. The latter encodes the ACs into Base64 plain text (in Step 5 above). Now the encoded ACs can be transferred as plain text attributes from the origin to the target site, where they are decoded into normal binary ACs before being passed to the PERMIS PV/PDP (in Step 9 above), for use by the RAP and TAP in decision making.

If a user possesses multiple roles, then multiple ACs can be assigned to the user and stored in the LDAP directory at the origin site. Shibboleth will retrieve all the role ACs at the origin site, encode them and then join them with semicolons, before passing them to the target site as a multi-valued attribute (in Step 5 above). After mod_shib puts the combined text encoded AC into the HTTP header (in Step 6 above), mod_permis will retrieve it and restore it into separate encoded ACs which can then be passed to the PERMIS PV/PDP for access control decision making (in Step 7 above). In this way

[8] This is true as of version 1.2 of Shibboleth. However the Shibboleth developers have said that binary attributes will be supported in a future release.

multiple ACs can be handled and utilised in access control decision making in Shibboleth.

5.2 PERMIS SAAM in Push Mode with Plain Attributes

If the target site trusts the origin's attribute repository and the origin as a single AA, then the origin will store plain attributes in its repository, and pass them in digitally signed SAML messages to the target. This is the standard Shibboleth mode of operation. In this mode, the interactions between Shibboleth and the PERMIS SAAM are nearly the same as in Section 5.1 except that it is the user's attributes, not the user's DN and role ACs, that are passed by Shibboleth and used by the PERMIS PV/PDP to make decisions. In this case SAAM works in push mode, by pushing the attributes which were retrieved by Shibboleth, to the PERMIS PV/PDP.

The Shibboleth origin site can be configured to append a scope domain to each released attribute. Scope domains are used to distinguish between different attribute issuers at the origin site, for example some attributes could have a scope domain of "salford.ac.uk", while others could have a scope domain of "computing.salford.ac.uk" (note that the same attribute cannot have multiple scope domains, which effectively precludes dynamic delegation of authority). When the PERMIS PV is being passed "scoped" attributes instead of digitally signed ACs, the scope domains take the place of the AC signers (i.e. the SoAs). In order to validate "scoped" attributes, the PERMIS RAP should specify the scope domains as SoAs in place of AC issuer DNs. We have reserved a special URL "shib:<scope domain name>" for this. In the above example there would be two corresponding SoAs identified in the RAP by the special URLs: "shib:salford.ac.uk", and "shib:computing.salford.ac.uk"[9]. If scope domains are not being used by an origin site, then SAAM inserts the name of the origin site as the scope domain for all the attributes. The scoped attributes can now be validated against the RAP by the PERMIS PV in the same way as X.509 AC issuers, except that cryptographic validation cannot be performed. Thus there is no proof who actually issued the attributes as "scoped" attributes don't have digital signatures.

The other difference from the scenario in Section 5.1 is that the user's LDAP DN is not provided (since they have a pseudonym dynamically generated by the origin site). Therefore the PERMIS Subject Domain sub-policy should include the null DN (meaning any DN is allowed) and the RAP should refer to this subject domain when specifying whom the attributes can be assigned to. Otherwise the attributes of the pseudonymous users (users with a null DN) will not be valid and all access will be denied to them.

5.3 PERMIS SAAM in Pull Mode

If the target trusts different attribute authorities based at the origin site and elsewhere, and wishes to authorise users based on these, then the origin site may not always be able to push all the attributes to the target site. In this case the PERMIS SAAM should work in pull mode to fetch the ACs itself. An example might be: a graduate is issued

[9] The PERMIS policy syntax has been extended to allow URLs as SoA identifiers instead of LDAP DNs. Thus '<SOA ID="Salford" URL="shib:salford.ac.uk"/>' defines an SoA that is identified by the Shibboleth scope domain "salford.ac.uk" in the plain-text attributes.

with a degree certificate by a university, a doctor is issued with a "clinician" certificate by the General Medical Council, and an engineer is issued with a "certified MS engineer" by a Microsoft accredited agency. In this case various distributed LDAP repositories (denoted as "AC LDAP" in Fig. 3) may sit in various places other than the origin site, and should be accessible by the PERMIS PV. The PERMIS PV can operate in pull mode and fetch all the needed ACs from the LDAP repositories. In this working mode, only one attribute of the user - the user's DN, should be configured and stored in the origin LDAP repository. The user's DN denotes the holder identity of the ACs in the various LDAP repositories and this DN will be retrieved and passed by Shibboleth to the PERMIS PV, so that the PV can know which ACs to retrieve from the various LDAP repositories. Once the ACs have been retrieved by the PERMIS PV, the PV will use the RAP to determine which ACs are trusted, and the PDP will use the TAP to determine if the user has the necessary attributes to access the resource.

The interactions between Shibboleth and the PERMIS SAAM are as follows.

(1) A user contacts a Shibboleth-protected resource site with the browser, is redirected by the SHIRE to the WAYF site, is authenticated at the origin site; then the browser is redirected back to the target site along with a handle package. The SHAR at the target site gets the handle and sends the handle to the AA of the origin site with an attributes query. (the same as Step 1 to Step 4 in Section 5.1)
(2) The AA retrieves the user's DN from the origin LDAP repository and sends this back to the SHAR.
(3) The SHAR passes the user's DN to the Apache HTTP header.
(4) In the authorisation phase in the HTTP request handling process, mod_permis is first invoked by the Apache server, and the user's DN is acquired by mod_permis through the HTTP header.
(5) Mod_permis calls the PV and passes the user's DN to the PV. The PV retrieves the user's ACs from the various LDAP repositories according to the user's DN, then validates them against the RAP. Finally the PDP makes an authorisation decision based on the user's validated attributes, the target resource, the action being requested and the current RBAC policy.
(6) After the PDP makes the decision, the decision is returned back to mod_permis.
(7) Mod_permis returns the decision result to the Apache server, then the user can be granted or denied access to the target resource according to the decision result.

6 PERMIS SAAM with Apache and Without Shibboleth

Since the PERMIS SAAM can work in pull mode and the PV is able to directly fetch ACs from LDAP repositories elsewhere, we can integrate the PERMIS SAAM with other Apache authentication systems without requiring Shibboleth to provide authentication or the user's attributes. This will provide us with a PERMIS authorisation service for web based resources, provided the user's DN can be passed from the authentication system to SAAM. The PV can then use the user's DN to fetch the user's ACs from the various LDAP repositories and make access control decisions based upon them.

6.1 System Structure

In this section we describe how the PERMIS SAAM is configured to work with the Apache server where the Apache module mod_auth_ldap is used as the authentication module. The system structure is shown in Fig. 4. There is only one major difference between Fig. 4 and Fig. 3: in Fig. 3, the authentication service is performed by the Shibboleth system which is distributed between two computer systems (the origin and the target), while in Fig.4 the authentication service is performed by mod_auth_ldap which is an Apache module located in the same (target) computer system as the PERMIS SAAM. Note that the LDAP AC repository in Fig.4 doesn't necessarily need to sit in the same computer system as the PERMIS SAAM - it may sit somewhere else as it does in Fig. 3.

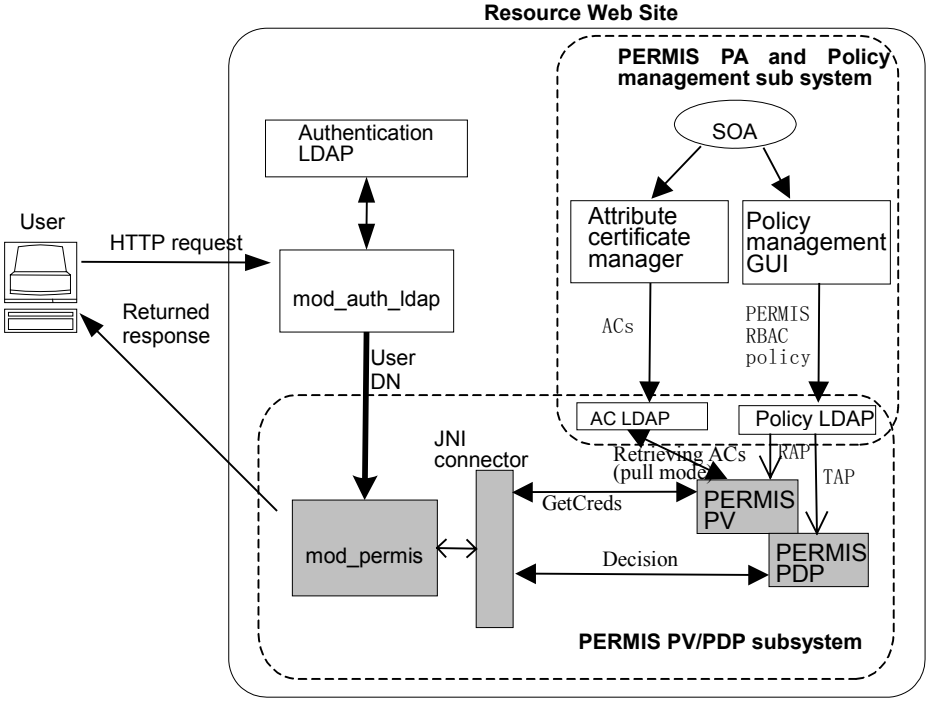

Fig. 4. Structure of Apache-PERMIS SAAM Integration

6.2 Interactions Between SAAM, the Authentication Module and the Apache Server

The interactions between SAAM, the authentication module (mod_auth_ldap) and the Apache server are as follows.

(1) When a user contacts an Apache web server, requesting access to a URL which is protected by mod_auth_ldap and mod_permis, the user is prompted by mod_auth_ldap to enter their username and password in order to be authenticated.

(2) Mod_auth_ldap authenticates the user by searching in the authentication LDAP server and locating the correct entry which matches the username and password, then retrieves and puts the user's DN in the Apache HTTP header.
(3) During the authorisation phase in the HTTP request handling process, mod_permis is invoked by the Apache server, and the user's DN is acquired by mod_permis through the HTTP header.
(4) Mod_permis calls the PV and passes the user's DN to it, the PV retrieves the user's ACs from the configured LDAP servers and validates them. The PDP then makes an authorisation decision based on the valid attributes and this is returned back to mod_permis.
(5) Mod_permis returns the decision result to the Apache server, and the user is granted or denied access to the target resource according to the decision result.

In our implementation, the Apache module mod_auth_ldap has been slightly modified so as to output the user's DN to the HTTP header during authentication[10].

7 Conclusions

The PERMIS SAAM module has been successfully developed and integrated with Shibboleth to replace the authorisation function in Shibboleth without modifying the Shibboleth source code. The flexibility, functionality and granularity of Shibboleth's authorisation decision making capabilities have been improved by adding PERMIS's policy controlled hierarchical RBAC implementation. When additional RBAC functionality, such as dynamic delegation of authority and separation of duties are added to future PERMIS releases, these will be automatically inherited by Shibboleth. Although the PERMIS SAAM was originally targeted at Shibboleth and was integrated with Shibboleth and the Apache server, an unexpected benefit is that it can work perfectly well with other Apache authentication modules without requiring Shibboleth to be present. By deploying the SAAM module, flexible, distributed, fine grained and more functional access control can be achieved by Apache web sites as well.

Acknowledgements

The authors would like to thank UK JISC for funding this work under the SIPS project.

References

1. S. Cantor. Shibboleth Architecture, Protocols and Profiles, Working Draft 02. 22 September 2004, see http://shibboleth.internet2.edu/
2. D. W. Chadwick, A. Otenko, E. Ball. Role-based access control with X.509 attribute certificates. *IEEE Internet Computing*, March-April 2003, pp.62-69.
3. ISO 9594-8/ITU-T Rec. X.509 (2001). *The Directory: Public-key and attribute certificate frameworks*

[10] Note that if other authentication modules are to be used with PERMIS, they also need to be modified to publish the authenticated DN in the HTTP headers of the request.

4. D. W. Chadwick, A. Otenko, V. Welch. Using SAML to link the GLOBUS toolkit to the PERMIS authorisation infrastructure. In *Proceedings of Eighth Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, Windermere, UK, September 15-18, 2004, pp.251-261.
5. OASIS. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, 2 September 2003.
6. D.W.Chadwick, A. Otenko. RBAC Policies in XML for X.509 Based Privilege Management. In M. A. Ghonaimy, M. T. El-Hadidi, H.K. Aslan, editors, *Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002)*, May 7-9, 2002, Cairo, Egypt. Kluwer Academic Publishers, pp.39-53.
7. The Apache Software Foundation. http://httpd.apache.org/
8. http://www.w3.org/Protocols/rfc2616/rfc2616.html
9. D. W. Chadwick, A. Otenko, W. Xu. Adding Distributed Trust Management to Shibboleth. In *Proceedings of 4th Annual PKI R&D Workshop: Multiple Paths to Trust*, NIST, Gaithersburg, MD, April 19-21, 2005.
10. R. Sandhu, D. Ferraiolo, R. Kuhn. The NIST Model for Role Based Access Control: Towards a Unified Standard. In *Proceedings of 5th ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 2000, pp.47-63.
11. M. Wahl, T. Howes, S. Kille. *Lightweight Directory Access Protocol (v3), RFC 2251*, Dec. 1997.
12. D. Ferraiolo, J. Barkley, and R. Kuhn. A role-based access control model and reference implementation within a corporate internet. *ACM Transactions on Information and System Security*, vol.2, no.1, February 1999, pp.34-64.
13. S. P. Joon, R. Sandhu, and G. Ahn. Role-based access control on the web. *ACM Transactions on Information and System Security*, vol.4, no.1, February 2001, pp.37-71.
14. J. S. Park, R. Sandhu. RBAC on the Web by smart certificates. In *Proceedings of 4th ACM workshop on role-based access control* (RBAC '99, Fairfax, VA, Oct. 28-29, 1999). ACM, New York, NY.
15. ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996. *Security Frameworks for open systems: Access control framework.*

CA-in-a-Box

Mark Franklin, Kevin Mitcham, Sean Smith, Joshua Stabiner, and Omen Wild

Dartmouth PKI Lab and Department of Computer Science,
Dartmouth College, Hanover NH 03755 USA
mark.franklin@dartmouth.edu
sws@cs.dartmouth.edu
http://www.dartmouth.edu/~pkilab/

Abstract. An enterprise (such as an institute of higher education) wishing to deploy a PKI must choose between several options, all expensive and awkward. It might outsource certification to a third-party company; it might purchase CA software and appliances from a third-party company; it might try to build and maintain its own CA. In the latter two options, the enterprise faces the additional challenge of showing sufficiently safe practices to have its CA certified or cross-certified, for broader inter-operability.

This paper presents our research and development effort to address this problem. We use OpenCA to provide the basic functionality; we package it on a Linux installation on a bootable CD; we use the 1.1b TCG trusted platform module (standard on many desktop and laptop machines) to hold the private key; we also use the TPM to add assurance that the key can only be used when the system is correctly configured as the CA. This tool enables an enterprise to operate a CA possessing a degree of physical security and the ability to attest proper configuration to a remote certifier simply by booting a CD in a commodity machine. The code (and CD image) are all open-source, and will be available for free.

1 Introduction

Deploying PKI has many advantages for an enterprise. As members of a university, we are particularly receptive to (and practitioners of) standard PKI evangelism directed toward academic enterprises. Within the enterprise, PKI enables encryption and signing of e-mail and workflow documents, and identification and authorization for Web-based information services and network access. An Educase Net@EDU survey [12] shows several universities with production PKIs serving applications including virtual private networks, S/MIME e-mail, and document signing. With cross-certification (such as via the *Higher Education Bridge Certification Authority* [3]), these services can extend to permit applications such as resource sharing and document exchange between universities.

However, deploying PKI is one of the tougher and more expensive exercises a university IT department can endure. *Certification authorities (CAs)* are the backbone of most PKIs, but installing and maintaining them is notoriously complicated. The need to simplify the process of PKI setup has been apparent for several years and providing this service has become a profitable industry. Many universities have opted to outsource their CA to such corporations; this choice increases financial burden but limits headache

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 180–190, 2005.

and responsibility. (More than one university CIO has expressed frustration that certain commercial CAs cannot cleanly project a priori the per-certificate cost the university will face.) Other universities operate CAs in-house, by purchasing or licensing software and cryptographic appliances and devoting IT man-hours to operating it. Still others try to reduce costs further by rolling their own CA from open source and commodity machines—but end up spending far more than expected in engineer and support staff time. The Educause survey cited above also showed that a university PKI's financial costs exceeded $50,000 per year—due to hardware, licensing, training, help-desk and upkeep.

These costs create a substantial barrier to adoption.

Our Project. This challenge motivated our project. As a university, we began our PKI effort by exploring the various options and their costs, and elected to pursue the middle option: operating our own CA from commercial CA tools. However, the costs of this option were high—particularly since, in principle, a roll-your-own option with open source should have been easy and cheap.

As a side-project, we tried the open-source approach, and found it was neither easy nor cheap. It took two software engineers two weeks to get OpenCA installed and working. There are a lot of configurable options for OpenCA that require knowledge of the concepts and design of the OpenCA code, which are not trivial to come by. To make this path easier for others, we first attempted to make a cookbook: "Do X, then Y, then Z." We then realized we could automate it and get rid of many of the opportunities for mistakes.

In this paper, we attempt to lower the barrier to university PKI adoption, by producing tools that let a university set up a CA with hardware security and integrity protection simply by booting a CD. Furthermore, our tools provide the framework for remote attestation about the system's integrity, easing cross-certification.

The code and CD image will be available as open source.

This Paper. Section 2 provides a brief background of certification authorities and their responsibilities. Section 3 presents the components we used to assemble our project; Section 4 presents the design and implementation of "CA-in-a-Box." Section 5 reviews related work. Section 6 discusses some directions for future research, and Section 7 concludes.

2 Certification Authorities

Although variants such as PGP and SPKI/SDSI have appeal, traditional X.509 PKI is the basis for the common PKI applications that motivate a university to adopt PKI. In this dominant paradigm, the enterprise depends on its certification authority to act as the intermediary between end users and relying parties (who may be other end users, or special entities such as the "course registration Web site").

In this standard framework, the CA acts as a trusted third party by using its private key to sign certificates. Each certificate binds a public key to information (typically

identity) about the holder of the corresponding private key. The CA attests to the correctness of this binding; the university will have some mechanism in place, perhaps via a separate *registration authority (RA)*, to verify this binding.

Any certificate holder can present his certificate as confirmation that the University CA claims the certificate contains valid information. Anyone who trusts the CA as a certificate issuer for this domain should trust that his information applies to any entity that demonstrates knowledge of the certificate's private key.

Given the central role the CA plays in holding together this trust infrastructure, the primary responsibilities of a CA are to ensure its private key is only used for appropriately authorized operations—and to convince relying parties and other stakeholders that a significant barrier exists to keep it from being used for unauthorized ones.

Much of the associated work with setting up a university PKI stems from ensuring the CA fulfills these responsibilities. We consider some issues.

Failure. Any system that relies heavily on its hardware must account for the possibility of hardware failure. CAs are no different in this case, particularly if the installation includes some type of special tamper-resistant hardware to hold the private key. The network environment might also be relevant. Some installations depend on the CA being online (in order to handle requests and post CRLs and such); others insist (to minimize the interfaces exposed to an adversary) that the CA remain isolated from the network.

One avenue of failure is *denial of service (DOS)*. A distributed DOS attack from another network (for online CAs) or a power failure can render the CA temporarily unavailable and cause user dissatisfaction. However, these failures are temporary and easily fixed. More critical are DOS failures that cause the destruction of the private root key such as a fire in the room containing the CA machine, or a defect in the secure hardware containing the key. (We have also heard anecdotes about vendors of secure hardware abandoning the product line and stranding their customers.) In these cases, the ability to sign certificates and add users to the PKI is lost. Additionally, if a CA root key protects a list of escrowed user encryption keys, we lose that data as well.

Alternatively, if the adversary *compromises* the CA's private key, consequences can cause chaos within a PKI. By learning the private key, the adversary has given himself the power to sign certificates as if he were the CA. The users of the PKI, therefore, can no longer trust the CA's signature. Each certificate signed by the private root key must be revoked and new certificates must be generated with a new key. The information leak causes a breakdown in the trust between users, as well as, between the CA and the PKI.

Human Element. In addition to hardware failure, we must also consider the human element in certification. At some point in the certificate creation process, the CA must determine if it is going to vouch for the person requesting the certificate. The entire PKI trusts the CA (perhaps in conspiracy with an RA) to identify users correctly.

Consequently, the decision of how to architect the CA machinery holds great weight, as the details of the architecture may influence how easy or hard it is for an adversary—including a rogue insider—to cause invalid certificates to be issued. In a typical installation, even if we keep the private key inside a special device, the host machine determines when the private key is used and what data it operates on. Thus, the configuration of this machine becomes of paramount importance: Trojans, backdoors, unpatched software,

or even extra accounts can subvert the system. The system administrator of the CA hardware may have unique power to sign or not to sign certain certificates and establish this trust between users.

Cross-Certification. To enable cross-certification, the CA operator also needs to be able to establish various properties about CA operation to the satisfaction of the certifier. These properties include practices such as how (and how carefully) the CA verifies identities of the entities for which it is issuing certificates; separation of duties so no single individual can do bad things; physical and network security; what is logged and how the logs are kept; who has access and how access is controlled; how processes and procedures are defined and enforced; and where the private key is stored and how it is protected.

Design Goals. This discussion leaves us with some design goals for our project.

- To keep equipment costs cheap, we need to use free software and commonly available commodity equipment.
- To ensure security of the CA private key when not in use, we need to exploit tamper-resistant hardware.
- To ensure security of the CA private key when in operation, we need to ensure that only a properly configured machine can request operations with this key.
- To assist in cross-certification, we need to make it possible for a remote CA to draw conclusions about the trustworthiness of this operation.
- To keep labor costs low, we need to make this easy to use.

3 Components

This section presents the components we used for this project.

3.1 OpenSSL

At its foundation, our CA needs to perform cryptographic operations.

OpenSSL is an open-source cryptographic library used by many standard applications that require cryptographic support [9]. OpenSSL is structured to permit use of underlying special-purpose cryptographic hardware; in OpenSSL, an *engine* is a module that enables OpenSSL to use some particular type of underlying hardware.

3.2 OpenCA

Our CA needs to carry out basic certification authority operations.

OpenCA is an open-source certification authority that uses OpenSSL [8]. We use OpenCA to manage the standard functionality of the CA: certificate generation, publication, revocation; we felt that OpenCA was the best choice to build upon as it is the most developed of the open source options. OpenCA supports publishing certificates and CRLs to LDAP. It provides an online RA component for handling certificate requests. Additionally, OpenCA is distributed with several Perl modules that can be used by other scripts we might choose to write ourselves [1].

3.3 Knoppix

Our CA needs to run on a properly configured system.

Proper configuration is necessary for basic operation. In theory, one can just install OpenCA and possess a CA. In practice, we found this process to be rife with subtle configuration issues. As discussed earlier, proper configuration is also necessary for secure operation.

We decided that a simple, easy-to-use way to ensure proper configuration would be to provide a properly configured system as a bootable CD. For this component, we chose *Knoppix*, which provides a customizable framework for putting a complete Linux system (based on the Debian distribution) on a bootable CD image [6].

3.4 USB Flash Drive

Our CA needs space for installation-specific state. Minimally, we need space for static state, such as the private key (perhaps encrypted); however, a CA may accumulate dynamic state, such as logs of signed certificates.

To provide this, we configure the system to store its home directory on a removable USB flash drive.

3.5 TCPA/TCG TPM

It would increase security if our CA could store its private key in a safe place. However, we would like to avoid the expense and awkwardness of special-purpose equipment.

For this component, we chose the *Trusted Platform Module (TPM)*. Specified by the *Trusted Computing Group (TCG)* (formerly the *Trusted Computing Platform Alliance, TCPA*), the TPM is a smart-card like chip that is attached to the motherboard of many commodity PCs. We worked with the 1.1b version of the TPM [13], since that already comes by default with many desktops and laptops from IBM, and so is already somewhat ubiquitous.

The TPM acts as a credential store, keyed to its *platform configuration registers (PCRs)*. The host machine can store a value in the TPM and key it to specified values in a specified subset of the PCRs. The TPM will then decrypt and release that credential only when those PCRs have those values. Additionally, if the credential is an RSA private key, the host can request the additional feature of having the TPM never actually release the key—but rather only *use* it internally, and only when the PCR conditions are satisfied. The PCRs themselves are set in an interleaved way, starting with the boot ROM and BIOS, in order to reflect the configuration of that machine.

3.6 Bear/Enforcer

For holding the CA's private key in a TPM to be effective, we also need to take steps to tie it to the correct configuration for the CA on that machine. Commercial support for the TPM is still scarce, at this point; Linux support for the official *TCG Software Suite (TSS)* has recently been announced, but was not available. Furthermore, besides talking to the TPM, we need to figure out how to express "secure configuration" as a suite of

PCR values—which may be complicated by issues such as security-related software updates, which change the configuration but are necessary for the CA to remain secure.

For this component, we chose our lab's *Bear/Enforcer* code [7]. Bear/Enforcer is an open-source Linux tool suite that works with the 1.1b TPM to bind credentials to dynamic system configuration. Bear/Enforcer divides data into three important categories based on lifespan. Long term data, like BIOS, the kernel, and Bear/Enforcer itself, are protected by the TPM-witnessed boot process. Medium term data, applications and daemons, are protected via a database of hashes. At initialization time, Bear/Enforcer checks that this database is current and properly signed by a potentially remote *security administrator*; at run time, a *Linux Security Module (LSM)* checks these hashes whenever an inode is touched. Shorter term data, like configuration files, live on a *loopback filesystem*[1] that can be encrypted and unmounted when not in use.

4 Design and Implementation

This section discusses how we put the above components together to solve our problem. Figure 1 sketches this design.

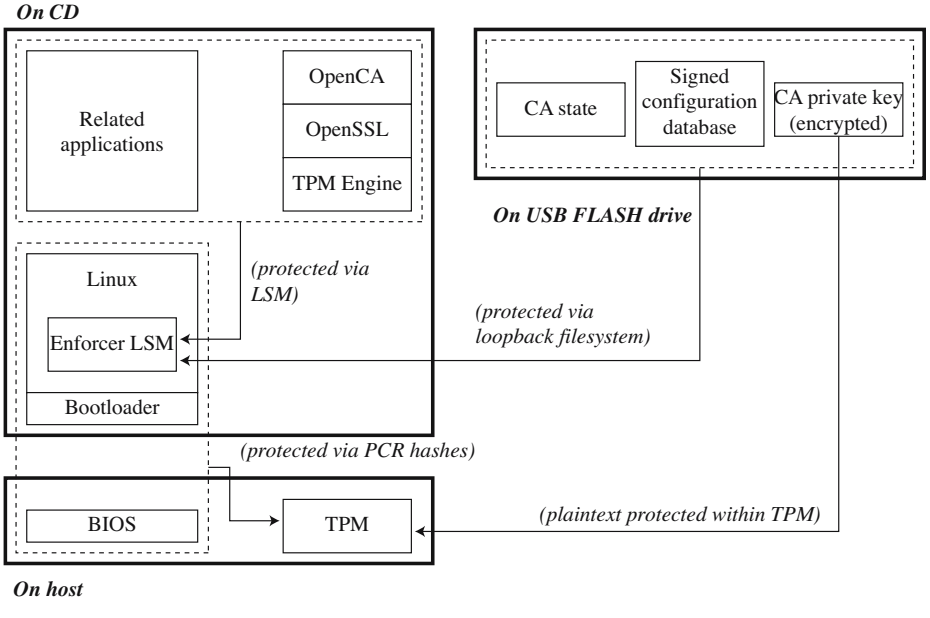

Fig. 1. A sketch of the system architecture for "CA-in-a-Box"

[1] A *loopback filesystem* is a single file that the kernel will mount and treat as if it were a filesystem.

4.1 Putting the Pieces Together

Knoppix. First, we remaster the Knoppix CD image by removing all unnecessary packages and features. This task both makes it easier to use, as well as decreases the *trusted computing base (TCB)*—e.g., if the kernel has no wireless extensions, then the CA cannot be compromised via a wireless attack.

OpenCA. Then, we need to automate the configuration and use of OpenCA. OpenCA works by having two or three bases of trust: the client enrollment base, the registration authority, and the actual CA.

Different institutions have different enrollment needs and strategies. For some, security is paramount, and this goal tends to drive these institutions to an *offline CA*. The design idea is that the actual CA lives offline and only communicates with the world via "sneakernet." The enrollment tool sends requests to the RA, who makes some decisions about them. If approved, the RA sends the data up to the CA. The CA actually issues the certificate and sends it back to the RA; the RA delivers the certificate to the client. This offline approach is the use model OpenCA developers had in mind and is how OpenCA works most naturally.

For other institutions, however, considerations such as convenience, minimization of administrator overhead, and immediate fulfillment of certificate requests tip the balance in favor of *online CAs*.

We set out to adapt OpenCA to work in an online mode (immediate fulfillment with no administrator intervention required) but were only able to get part way towards this goal. We combined the RA and CA functions onto one network-accessible system and combined the RA and CA databases to reduce the number of adminstrator steps needed to move certificates through the system (when RA and CA are separate, these steps are necessary).

Unfortunately, we found that the architecture of OpenCA prevented us from being able to make online enrollment totally automatic without undue engineering effort. For some applications (or certificate assurance levels), requiring administrator intervention for each enrollment is still desirable, but the number of clicks involved in our implementation is higher than would be possible in a fully refactored implementation, and for many mass end-user certificate deployments it would be useful to have an enrollment option that authenticates the end user and perhaps an RA "approver" and then automatically finishes the enrollment.

To set this all up, one must manipulate a large configuration file, that specifies all behaviors and that allows options such as changing the number of layers (e.g., add extra RA steps) or having multiple RAs. This configuration file was large, and documentation was sparse. We eventually figured out how to modify it to have an RA and CA running on the same machine without conflicting; the RA and CA communicate via file copying. This model best suited our installation, and we also felt that universities looking for low-barrier way to adopt PKI would not want the hassle of multiple machines and "sneakernet."

Private Key Security. One series of steps involves using the TPM to shelter the CA private key, to have the CA call the TPM for operations with this key. As discussed

earlier, OpenCA uses OpenSSL, and OpenSSL uses "engine" modules for implementations of cryptographic operations. The default engine (`openssl`) provides software functionality for OpenSSL cryptographic methods such as RSA, DSA, and RAND (a method for obtaining random numbers). The benefits of having this structure is the ease with which we can add new engines. When the user wishes to perform cryptographic operations on some specialized hardware device, he can simply load that engine into OpenSSL and then use the OpenSSL command-line normally. The loaded Engine has knowledge of the hardware device and how to interact with it [5].

To enable easy use of a TPM-housed RSA private key by OpenSSL, we built a `TPM Engine` module. We then built a custom compile of OpenSSL that contains this engine, and OpenCA to use the Engine API to look for the TPM.

We used TPM utilities from IBM to generate keys manually in the TPM. The engine takes over operations from there.

Configuration Security. We also need to set up Bear/Enforcer to ensure that the TPM only uses the private key when the system is properly configured.

To initialize, the operator first boots the system and runs a script to "take ownership" of the TPM. The operator then inserts the CD and FLASH drive and reboots the system. The BIOS, the boot-loader on CD-ROM, the kernel, and the OpenCA configuration all get hashed into the PCRs; the filesystem is initialized on the external device. If running in "local" mode, the operator can run a script here to generate the Enforcer database and sign it; if running in a scenario where a remote party specifies secure configurations, we check the validity of the database this party has signed. We generate the symmetric key for the encrypted loopback filesystem and set that up, and store the key as a credential bound to this PCR suite. The operator then runs our OpenCA configuration scripts, which stores its state in the loopback, and generates the CA private key within the TPM itself, bound to this PCR suite. The CA config files along with public keys are stored on this removeable device in the loopback filesystem. (The private keys are stored as encrypted blobs, usable only by the TPM.) This allows us to keep that information encrypted, on removable media, whenever it's not in use.

In normal boot, the BIOS, boot-loader and kernel are hashed into the PCRs. The OS gets loaded into system RAM. If Enforcer and the TPM determine the system configuration are satisfactory, the encrypted loopback filesystem is mounted so the CA configuration can be retrieved. OpenCA initializes and Enforcer checks the OpenCA binary. OpenCA tells openSSL to use the TPM Engine; if the configuration is still satisfactory, the encrypted private key is loaded into the TPM, which will then provide private key services to the CA.

Subsequent operation requires the CD, the FLASH drive, and that host machine.

4.2 Analysis

Security. By reducing the amount of hardware being used we are able to more directly protect the hardware we do use. Because our CA does not require disk access, we do not need to worry about software viruses or Trojans the CA machine might have picked up while performing non-CA duties; furthermore, the use of the bootable CD simplifies the problem of trying to maintain a special-purpose clean installation on that machine.

Denial of service can be a possibility if the TPM is destroyed or if hardware alterations fundamentally change the PCR values established during boot; Section 6 considers that further.

Protecting against a rogue system administrator is never an easy task. However, our approach provides us with an easy way to implement this protection. There are several components necessary for the "CA-in-a-Box" to work: a secret to help unlock the TPM's storage root key, the private key of the signer of the Enforcer database, and the dongle that contains the removable storage. We can distribute these elements among multiple people; potentially, we might distribute the database signer's key to a remote site (see Section 6).

There are several parts of OpenCA that an adversary can exploit. OpenCA depends on OpenRA to manage certificate requests. It needs a MySQL database to store completed certificates pre-signing. It uses Perl to process every request. OpenSSL is essential to signing every certificate. Even the Web browser plays a key role by acting as the user interface to the CA.

Our Enforcer/TPM integration adds several additional layers of protection. If the adversary discreetly replaced any of these binaries with modified versions, he could easily trick the CA into signing a certificate that the adversary generated. However, when Enforcer is active, this is not possible: the Enforcer is set up to monitor each of these programs and cause a kernel panic (and tell the TPM to render the private key unusable) when any of them changes in a way not permitted by the signed configuration file.

Suppose an adversary gains access to our CA and inserts an unsigned certificate into the MySQL database. Then the adversary modifies some OpenCA Perl scripts so the next certificate exported from the database to be signed is his. As soon as the administrator loads OpenCA and the scripts are touched, Enforcer will detect the change and cause a kernel panic, shutting down the CA and stopping the attack. Likewise, if an adversary tried the same attack by modifying the MySQL binary the same response occurs. Enforcer provides an extra layer of protection not previously part of any open source CA.

Scalability. The total number of users this system might support would be constrained by the database used to store the information, and the CA operator's time necessary to enroll them. If we assume one-year certificate lifetimes, and that a trained operator can reliably process an enrollment in five minutes, we project the system could get about 10,000 users in circulation (with 20 hours/week of operator time). OpenCA enrollment takes a lot of clicking, however; a more realistic projection might be 1000 users/certs as it stands now. Processing twenty requests a week is enough to keep things moving, but not overwhelm the operator.

Streamlining and batching the enrollment process is an area for future work. E.g., here at Dartmouth College, we issue students certificates when they first matriculate. However, these newly matriculated students go through many other physical processes where their identities have been validated and they are batched together in a room—perhaps even after they have been issued College ID cards with RFID chips. Considerable potential for streamlining exists; we plan to explore this in future work.

5 Related Work

Jeff Schiller at MIT suggested building an offline CA from a laptop and Dallas iButton [10]. In addition to OpenCA, other open-source CA options include *XCA*, a graphical front-end to OpenSSL [4]; *pyCA*, not currently in active development [11]; and *Papyrus*, based on PHP [2].

Other experimental CA projects include COCA [15] and MOCA [14],

6 Future Work

In future work, we plan both to finish some necessary features, as well as integrate and test new ones.

In the former category, we need to design and implement a way to back up the CA configuration for a second machine, should changes to the initial host render it unusable. We should be able to accomplish this task with a fairly straightforward application of the TCG design of exporting one TPM's secrets to be used by a second designated back-up machine, perhaps in combination with secret-sharing among trustees. We also need to examine the failure scenario of the USB token being removed before our code unmounts it; plaintext data may remain there. For this problem, we may decrypt into RAM instead. (This should not be a significant security issue, however, since the primary secret—the CA private key—is protected by the TPM; what matters for the loopback filesystem is integrity.) We also want to stay abreast of ongoing work in our Bear/Enforcer project—such as ensuring freshness of signed configuration files, and our recent integration of Enforcer with SE/Linux.

In the latter category, we want to finish building and testing tools to harness the configuration control and attestation features of Bear/Enforcer on the TPM in cross-certification. We plan to modify our Enforcer configuration-preparation tool for use by a bridge CA to establish signed databases for suitable CA configurations. We can then use the Bear/Enforcer attestation to communicate this status back to the bridge CA, thus easing enrollment in the bridge. We also plan to explore making this configuration information available to other relying parties, perhaps by setting up an attribute authority within Bear/Enforcer and having it sign attribute certificates about the CA configuration. We also plan to revisit the design decision to combine user enrollment, the RA, and the CA in one machine.

Eventually, we plan to validate these ideas in a broader pilot, perhaps in conjunction with HEBCA.

7 Conclusions

To conclude, our "CA-in-a-Box" project uses existing open source tools and commonly available commodity equipment to produce a CA that is easy to install and use, but which also exploits hardware protections for the CA private key and software configuration. We offer this work to the community, in the hope that this helps promote broader use of PKI (at least by fellow universities) by easing the burden of establishing an enterprise PKI and having it cross-certified.

Acknowledgments and Availability

The authors are grateful to our many helpful colleagues—particularly here in the PKI Lab, and in the greater higher education PKI community—for their helpful suggestions and comments.

We are currently preparing our code for release. For more information, please contact mark.franklin@dartmouth.edu.

This work was supported in part by the Mellon Foundation, by the NSF (CCR-0209144), by Internet2/AT&T, by Sun, by Cisco, by Intel, and by the Office for Domestic Preparedness, U.S. Dept of Homeland Security (2000-DT-CX-K001). The views and conclusions do not necessarily represent those of the sponsors.

References

1. Chris Covell and Michael Bell. OpenCA Guides for 0.9.2+. http://www.openca.org/openca/docs/online/.
2. John Douglass. The Papyrus Project (Version 4), 2005. http://www.cren.net/crenca/crencapages/papyrus.html.
3. Higher Education Bridge Certification Authority. http://www.educause.edu/hebca/.
4. Christian Hohnstadt. XCA, 2003. http://xca.sourceforge.net/.
5. Pravir Chandra John Viega, Matt Messier. *Network Security with OpenSSL*. O'Reilly & Associates, Sebastopol, CA, 2002.
6. Knoppix linux. http://www.knoppix.net/.
7. J. Marchesini, S.W. Smith, O. Wild, A. Barsamian, and J. Stabiner. Open-Source Applications of TCPA Hardware. In *20th Annual Computer Security Applications Conference*. IEEE Computer Society, December 2004.
8. OpenCA PKI Development Project. http://www.openca.org/openca/.
9. OpenSSL: the Open Source toolkit for SSL/TLS. http://www.openssl.org/.
10. Personal communication.
11. pyCA–X.509 CA, 2003. http://www.pyca.de/.
12. Barry R Ribbeck. The PKI Working Group End User Deployment Matrix, 2004. https://webspace.uth.tmc.edu/bribbeck/public/PKIWMATRIX.html.
13. Trusted Computing Platform Alliance. Main Specification, Version 1.1b. http://www.trustedcomputinggroup.org, February 2002.
14. Seung Yi and Robin Kravets. MOCA:Mobile Certificate Authority for Wireless Ad Hoc Networks. In *2nd Annual PKI Research Workshop*, 2002.
15. Lidong Zhou, Fred B. Schneider, and Robert Van Renesse. COCA: A Secure Distributed Online Certification Authority. *ACM Transactions on Computer Systems*, 20(4):329–368, 2002.

A Lower-Bound of Complexity for RSA-Based Password-Authenticated Key Exchange

SeongHan Shin, Kazukuni Kobara, and Hideki Imai

Institute of Industrial Science, The University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
shinsh@imailab.iis.u-tokyo.ac.jp
{kobara, imai}@iis.u-tokyo.ac.jp
http://imailab-www.iis.u-tokyo.ac.jp/imailab.html

Abstract. Some RSA-based PAKE protocols have been proposed using a challenge-response method for verifying the validity of the server's RSA public key due to the lack of a PKI. However, these kind of RSA-based PAKE protocols cannot specify the exact overall complexity of their protocols since there exists a system parameter l needed for the challenge-response method. In this paper we present an RSA-based PAKE (RSA-PAKE) protocol, followed by its lower-bound of complexity and the actual computation and communication costs.

1 Introduction

Both mutual authentication and generation of a cryptographically-secure key can be achieved by an authenticated key exchange (AKE) protocol. The typical candidates for AKE protocols are IKE (Internet Key Exchange) [9,13], SSL/TLS (Secure Socket Layer/Transport Layer Security) [8,12] and SSH (Secure SHell) [11] all of which are based on PKI (Public Key Infrastructures). This type of AKE protocols are believed to be secure against an active adversary who fully controls the communications. However, each party must verify the counterpart's certificate (e.g., X.509 Certificates) via CRL (Certificate Revocation Lists) or OCSP (Online Certificate Status Protocol), before running the actual protocol, which entails additional computation and communication costs [10]. That is, the hindrance to implementation is in the burden of PKI itself. The distribution and maintenance of certificates is particularly costly and resource intensive for environments where it's not already in place.

The alternative may be AKE protocols where a (strong) secret key, chosen from a large range of space, is shared between the parties and used for symmetric-key encryption or message authentication (e.g., [5,19]). In practice, the strong keys are often and commonly substituted by human-memorable passwords chosen from a relatively small size of dictionary (e.g., alphanumerical passwords). Owing to the usability and convenience of passwords, password-based AKE protocols have been extensively investigated for a long time where a client remembers a short password and the corresponding server holds the

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 191–205, 2005.

password or its verification data that is used to verify the client's knowledge of the password. However, designing a secure password-based AKE protocol is not trivial since there are two major attacks on passwords: on-line and off-line dictionary attacks. The on-line dictionary attack is a series of exhaustive searches for a secret performed on-line, so that an adversary can sieve out possible secret candidates one by one communicating with the target party. In contrast, the off-line dictionary attack is performed off-line in massively parallel computers where an adversary exhaustively enumerates all possible secret candidates, in an attempt to determine the correct one, by simply guessing a secret and verifying the guessed secret with recorded transcripts of a protocol. While on-line attacks are equally applicable to all of the password-based protocols, they can be prevented by letting a server wait appropriate intervals between invalid trials. But, we cannot avoid off-line attacks by such policies, mainly because the attacks can be performed off-line and independently of the parties.

In [3], Bellovin and Merritt first showed the feasibility that a combination of symmetric and asymmetric (public-key) cryptographic techniques can provide insufficient information for an adversary to verify a guessed password and thus defeat off-line dictionary attacks. They also proposed a set of protocols, known as Encrypted Key Exchange (EKE), which were very influential and formed the basis for what we call Password-Authenticated Key Exchange (PAKE) protocols. The 2-party PAKE protocols are designed in a strict setting in that a client remembers *only* his/her password (without any devices and any additional setup requirements) which is shared with the counterpart server. By asymmetric cryptographic techniques, we can roughly classify PAKE protocols into two categories: Diffie-Hellman based and RSA-based ones. When it comes to the lower-power computing devices (especially, on the client's side), RSA-based PAKE protocols may be preferable to the Diffie-Hellman based ones when computing one modular exponentiation the latter requires an algorithm that has cubic running time on average, while the former has a quadratic running time based on the bit-length of its inputs. Hereafter we focus on the RSA-based PAKE protocols.

1.1 Previous RSA-Based PAKE Protocols

In the RSA-based EKE (RSA-EKE) protocol [3], Bellovin and Merritt raised a variant of off-line dictionary attacks (so-called e-residue attacks), which exploit the RSA public key (e, n) such that $\gcd(e, \varphi(n)) \neq 1$. Due to the lack of a PKI, e-residue attacks are possible because a server generates an RSA key pair (e, d, n) and transmits the public key (e, n) without its certificate. The basic idea of the e-residue attack is that the encryption function $\mathsf{RSA}_{n',e}(x) \equiv x^e \bmod n'$ is no longer a permutation in $\mathbb{Z}^\star_{n'}$ which maps an element $x \in \mathbb{Z}^\star_{n'}$ to the set of e-residues (a proper subset of $\mathbb{Z}^\star_{n'}$). Since the adversary knows the factorization of n', it is easy to check whether an element $x \in \mathbb{Z}^\star_{n'}$ is e-residues or not.

With the e-residue attack, [17,23] showed the RSA-EKE protocol to be insecure. Based on number-theoretic techniques, Patel further investigated the security of the RSA-EKE variant with the conclusion that simple modifications

of RSA-EKE would not prevent off-line dictionary attacks [17]. In 1997, Lucks proposed an RSA-based PAKE protocol (called OKE) which was claimed to be secure against the e-residue attack [14]. Later, Mackenize et al., found that the OKE protocol is still subject to the e-residue attack and proposed an RSA-based PAKE protocol (SNAPI) along with a formal security proof in the random oracle model [16]. So far, two approaches have been taken to thwart an e-residue attack by ensuring that the public exponent e is relatively prime to $\varphi(n)$. As one approach, MacKenzie et al., [16] in their SNAPI protocol made explicit requirements on e and n: (1) the first method is to set e to be a prime, in the range of $2^k \leq e < 2^{k+1}$, greater than n where k is the bit-length for the RSA security parameter; (2) the second method is to set e to be a prime such that e is greater than $\sqrt{n}$ and $(n \bmod e) \nmid n$. These methods may render the SNAPI protocol impractical in resource limited devices in that the value of e is too large to exploit the efficiency of when e is a small prime. From this motivation, Zhang proposed two RSA-based PAKE protocols (PEKEP and CEKEP) where he removed the constraint of e using number-theoretic techniques, but the computation costs of the client still remains high [22].

When a client has low-power computing devices, e should be a prime as small as possible in order to attain a high efficiency of computation. The other approach in order to avoid the e-residue attack is using a challenge-response method, deployed in the subsequent RSA-based PAKE protocols [23,1,20,21,7] to the RSA-EKE one, with which a client can verify e interactively with a server. For a small e, Zhu et al., [23] proposed a challenge-response method that is revised from an idea of [3]. However, the security holes of [23] were found in [1,21] each of which suggested its countermeasure with no security proof. In [20], Wong et al., reduced the communication overhead involved in the challenge-response method but they only gave an informal security analysis. Recently, Catalano et al., [7] have proposed an isomorphic construction for PAKE protocols which is suitable for several group structures and have provided a security proof in the random oracle model. One of the constructions is an RSA-based PAKE protocol modified from [23]. A drawback of the challenge-response method is the large computation costs of the client and the communication overhead involved in the verification of the RSA public key.

Motivation. Unfortunately, all of the RSA-based PAKE protocols [23,1,20,21,7] deploying such a challenge-response method have an (implicit or explicit) assumption that the number of validity checks l is a system parameter. If the system parameter l is a small integer, then the e-residue attack is applicable. On the other hand, if the system parameter l goes to infinity then the probability that all the validity checks on the client side are passed with a fake RSA public key (e, n'), such that $\gcd(e, \varphi(n')) \neq 1$, goes to 0. When we implement an RSA-based PAKE protocol using a challenge-response method in practice and compare it with the Diffie-Hellman based ones for efficiency, the natural question is how can we determine the system parameter l without jeopardizing its security against off-line dictionary attacks?

1.2 Our Contributions

We present an RSA-based PAKE (RSA-PAKE) protocol, using a challenge-response method for verifying the validity of an RSA public key, which is in fact an instantiation of [7] but slightly modified. As we mentioned in the above Motivation, the previous RSA-based PAKE protocols (including [7]) which exploit the challenge-response method didn't deal with the lower-bound of complexity of their protocols. For that, we deduce its lower-bound of complexity by comparing it with the previous RSA-based and Diffie-Hellman based ones after getting the exact computation costs of the client and the communication overheads.

Organization. In Section 2, we show an RSA-based PAKE (RSA-PAKE) protocol, followed by its lower-bound of complexity in Section 3. Section 4 is assigned to efficiency comparisons with the previous PAKE protocols.

2 An RSA-Based PAKE (RSA-PAKE) Protocol

Before showing an RSA-based PAKE (for short, RSA-PAKE) protocol, we will start by giving some preliminary notations to be used. Let k and l_j denote the security parameters, where k ($k > l_j$) can be thought of as the security parameter for RSA and temporal random values (say, 1024 bits), and l_j can be thought of as the security parameter for hash functions (say, 160 bits). Let N be a dictionary size (cardinality) of passwords (say, 36 bits for alphanumerical passwords with 6 characters). Let l denote the system parameter for verifying the validity of an RSA public key. Let $\{0,1\}^\star$ denote the set of finite binary strings and $\{0,1\}^k$ the set of binary strings of length k. Let "||" denote the concatenation of bit strings in $\{0,1\}^\star$.

Let us define secure one-way hash functions (e.g., SHA-1). While both $\mathcal{H}$ and $\mathcal{G}$ denote full-domain hash (FDH) functions from $\{0,1\}^\star$ to $\mathbb{Z}_n^\star \backslash \{1\}$, hash functions from $\{0,1\}^\star$ to $\{0,1\}^{l_i}$ are denoted $\mathcal{H}_j$, for $j = 0, 1$. Here $\mathcal{H}, \mathcal{G}$ and $\mathcal{H}_j$ are distinct random functions one from another. Let $\mathcal{C}$ and $\mathcal{S}$ be the identities of the client and server, respectively.

2.1 The RSA-PAKE Protocol

As illustrated in Fig. 1., the RSA-PAKE protocol is run between client $\mathcal{C}$ and server $\mathcal{S}$ as follows. When client $\mathcal{C}$ wants to share a session key securely with server $\mathcal{S}$, they perform the first three flows in order to verify the validity of the server's RSA public key. At first, server $\mathcal{S}$ sends to client $\mathcal{C}$ the RSA public key (e, n) that is generated from $\mathsf{RSAKeyGen}(1^k)$. If either e or n has 2 as a factor, the client aborts the protocol. Otherwise, client $\mathcal{C}$ correspondingly returns a random number r chosen from $\{0,1\}^k$. Upon receiving r, server $\mathcal{S}$ checks whether r is in the right range $\{0,1\}^k$. If not, the server aborts the protocol. Otherwise, server $\mathcal{S}$ computes $x_i \equiv y_i^d \bmod n$ with y_i, for $i \leftarrow 1$ to l, under the RSA private key (d, n). Each of y_i is a full-domain hash value of (n, r, i). Server $\mathcal{S}$ then sends

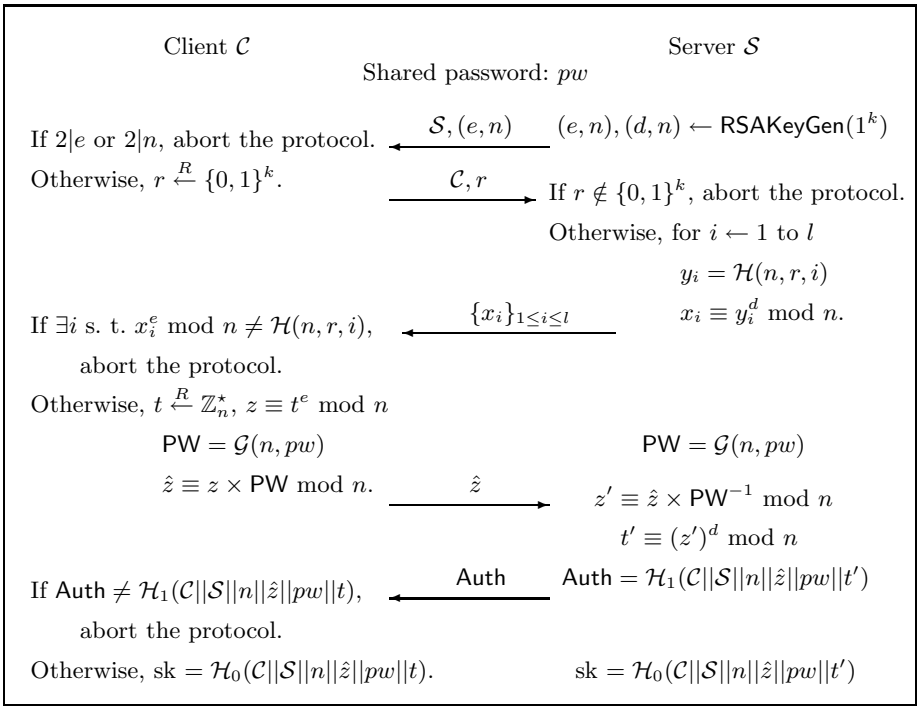

Fig. 1. An RSA-based PAKE (RSA-PAKE) protocol that is an instantiation of [7] but is slightly modified. The mutual authentication can be achieved by making client $\mathcal{C}$ send a simple authenticator for server $\mathcal{S}$

$\{x_i\}_{1 \le i \le l}$ to the client. The latter checks the correctness of $\{x_i\}_{1 \le i \le l}$ with the RSA public key (e, n) by accepting if, for all $i = 1, \cdots, l$, $x_i^e \bmod n = \mathcal{H}(n, r, i)$. If all of the validity checks are passed in turn, client $\mathcal{C}$ calculates $\hat{z}$ using a mask generation function as the product of an encryption z of a random value t under the public key (e, n) with a full-domain hash of n and pw, before sending it to server $\mathcal{S}$. Otherwise, the client aborts the protocol. The server can divide this encrypted value $\hat{z}$ by a hash of (n, pw), and then decrypt the resultant value z' under its private key (d, n) so as to obtain t' from which its authenticator Auth and the session key sk are derived. After receiving Auth from the server, client $\mathcal{C}$ computes the session key sk, as long as the authenticator Auth is valid, which is used for their subsequent cryptographic algorithms.

2.2 Security Proof

In this section we show the RSA-PAKE protocol of Fig. 1. is provably secure in the random oracle model [4] under the assumption that inverting an RSA instance is hard. Informally speaking, an adversary cannot determine the correct password through off-line dictionary attacks since generating the valid client's

authenticator after computing z or generating the valid server's authenticator falls into on-line dictionary attacks (which can be easily prevented and detected). Here we assert that the RSA-PAKE protocol distributes session keys that are semantically-secure and provides unilateral authentication for the server $\mathcal{S}$. The security reduction to the one-wayness of RSA is based on [6] where a challenge RSA problem is included in the answer of many hash queries so that the adversary is useful to the simulator with greater probability.

Theorem 1. (AKE/UA Security). *Let P be the* RSA-PAKE *protocol of Fig. 1., where passwords are chosen from a dictionary of size N. For any adversary $\mathcal{A}$ within a polynomial time t, with less than q_s active interactions with the parties (*Send*-queries) and q_p passive eavesdroppings (*Execute*-queries), and asking q_g and q_h hash queries to $\mathcal{G}$ and any $\mathcal{H}_i$ respectively, $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \leq 4\varepsilon$ and $\mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) \leq \varepsilon$, with ε upper-bounded by*

$$(q_{\mathcal{C}} + 2q_{\mathcal{S}})/N + 6q_{\mathcal{S}} \cdot \mathsf{Succ}^{\mathsf{ow}}_{RSA}\left(q_h^2, t + 2q_h^2\tau_{rsa}\right) + q_{\mathcal{C}} \cdot \mathsf{Succ}^{\mathsf{forge}}(t) + \frac{q_{\mathcal{C}}}{2^{l_1}} + \frac{Q^2}{2s} + \frac{(q_g + q_h)^2 + Q^2}{2^{k+1}}\ , \tag{1}$$

where $q_{\mathcal{C}}$ and $q_{\mathcal{S}}$ denote the number of $\mathcal{C}$ and $\mathcal{S}$ instances involved during the attack, l_1 is the output length of $\mathcal{H}_1$, Q denotes the number of involved instances ($Q \leq 2q_p + q_s$), s is the size of the possible RSA key pair, k is the security parameter, and τ_{rsa} is the computational time needed for performing one RSA operation.

Due to the restricted space, we omit the security model, definitions and the proof but those can be shown essentially in the same way as [7] with some adjustments.[1] This theorem actually motives us to derive its lower-bound of complexity for the RSA-PAKE protocol.

3 The Lower-Bound of Complexity

In this section we deduce the lower-bound of complexity (especially, regarding l) in the RSA-PAKE protocol. Since the overall computation and communication complexities of the RSA-PAKE protocol only depends on $\{x_i\}_{1 \leq i \leq l}$, it is crucial to determine the number of x_i (or, the lower-bound of l) enough to make the client ensure that e is not a divisor of $\varphi(n)$. For starters, we will show clearly the probability for an adversary to forge a proof of validity of an RSA public key.

Fact 1. *For odd integers e ($e \geq 3$) and n, such that $\gcd(e, \varphi(n)) \neq 1$, any e-th power residue modulo n should have at least three e-th roots.*

[1] However, the security reduction of Theorem 1 is not tight because of the factor q_h^2 in $\mathsf{Succ}^{\mathsf{ow}}_{RSA}(\cdot,\cdot)$; hence with $q_h = 2^{80}$, huge RSA modulus would be necessary to make the proof meaningful.

In the RSA function, the public key (e, n) is a pair of odd integers. Since n is the product of distinct odd primes p and q, n is always an odd integer. If e has 2 as a factor (or, $e = 2a$ for some positive integer a), $\gcd(e, \varphi(n)) = 2\delta$ for some δ so that e is also an odd integer. The check for (e, n) to be odd is done on the client's side after receiving the server's public key in the RSA-PAKE protocol. From now on, we define the RSA function by $\mathsf{RSA}_{n,f}(w) \equiv w^f \bmod n$ for all $w \in \mathbb{Z}_n^\star$.

Corollary 1. *We denote by* forge *an event that an adversary forges a proof of validity for* $\mathsf{RSA}_{n,e}$ *with a fake RSA public key. The probability of* forge *is upper-bounded by*

$$\Pr[\mathsf{forge}] \leq (1/3)^l \ . \tag{2}$$

Proof. From Fact 1, it is obvious. In the RSA-PAKE protocol (Fig. 1.), client $\mathcal{C}$ can detect fraudulent values of e by verifying $\{x_i\}_{1 \leq i \leq l}$ under the public key (e, n).[2] Suppose an adversary $\mathcal{A}$, impersonating server $\mathcal{S}$, who sends not only a fake RSA public key (e, n'), such that $\gcd(e, \varphi(n')) \neq 1$, but also $\{x'_i\}_{1 \leq i \leq l}$ each of which is one of the e-th roots of y_i. Since the adversary doesn't compute d satisfying $ed \equiv 1 \bmod \varphi(n')$, the probability that client $\mathcal{C}$ verifies $\{x'_i\}_{1 \leq i \leq l}$ correctly under the public key (e, n') is at most $(1/3)^l$. □

After receiving $\hat{z}$ from client $\mathcal{C}$ where $\hat{z} \equiv \mathsf{RSA}_{n',e}(t) \times \mathsf{PW}$, the adversary can try a guessed password pw' for $\mathsf{PW}' = \mathcal{G}(n, pw')$ in order to obtain $z' \equiv \hat{z} \times {\mathsf{PW}'}^{-1}$, and then check that z' is in the e-th power residues. If not, pw' is not the correct password and therefore can be removed from the password space $\mathbb{N}_{\mathsf{Password}}$ with probability of $(1/3)^l$. Of course, such an e-residue attack can be prevented if l is sufficiently large so that $(1/3)^l$ becomes to be negligibly small.

3.1 The Lower-Bound of l

As we have already discussed in the Introduction about the efficiency of computation on the client's side, we now fix $e = 3$ for the RSA-PAKE protocol since 3 is the most small prime in $\mathbb{Z}^\star_{\varphi(n)}$. The rationale of this section is that the success probability of an off-line (e-residue) attack can be upper-bounded by that of an on-line attack. That means, in the j-th interaction between an adversary and a party the number of the remaining password candidates in an off-line attack should be greater than or equal to that of the remaining password candidates in on-line attack. In order to deduce the lower-bound of l, we assume the following:

Assumption 1. *The passwords are* uniformly *distributed over password space* $\mathbb{N}_{\mathsf{Password}}$.

Here we specify on-line attacks for an adversary to break the RSA-PAKE protocol.

[2] If $\gcd(e, \varphi(n)) = 1$, each of $\{x_i\}_{1 \leq i \leq l}$ of course has a unique e-th root and $x_i^e \bmod n = \mathcal{H}(n, r, i)$.

- After honestly running the first three flows with server $\mathcal{S}$ in the protocol, an adversary guesses a password pw', computes $\hat{z} \equiv \mathsf{RSA}_{n,e}(t) \times \mathsf{PW}'$ as the product of an encryption of a random value t under the public key (e, n) with a full-domain hash of (n, pw'), and sends it to the server. On receiving Auth from the latter, the adversary can check that the guessed password is correct by seeing $\mathsf{Auth} = \mathcal{H}_1(\mathcal{C}||\mathcal{S}||n||\hat{z}||pw'||t)$ where the probability of $pw' = pw$ is $1/N$. If not, the adversary can remove one password pw' from a dictionary size of passwords N. Hence $pw' \neq pw$ happens with probability of $1 - 1/N$.
- An adversary honestly generates the RSA key pair $(e, n), (d, n)$, such that $\gcd(e, \varphi(n)) = 1$, and runs the first four flows with client $\mathcal{C}$. After receiving $\hat{z}$, the adversary guesses a password pw', computes $t' \equiv t \times \mathsf{RSA}_{n,d}\left(\mathsf{PW} \times \mathsf{PW}'^{-1}\right)$ with $\mathsf{PW}' = \mathcal{G}(n, pw')$, and sends the authenticator Auth to the client. Hence the probability that Auth is accepted as a valid one by the client is $1/N$.

With respect to the j-th $(1 \leq j \leq N)$ interaction between an adversary and a party ($\mathcal{C}$ or $\mathcal{S}$), the success probability of an on-line attack follows directly the binomial distribution $P(r; j)$

$$P(r; j) =_j C_r \left(\frac{1}{N}\right)^r \left(1 - \frac{1}{N}\right)^{j-r} = \frac{j!}{r!(j-r)!} \left(\frac{1}{N}\right)^r \left(1 - \frac{1}{N}\right)^{j-r} \tag{3}$$

where r $(1 \leq r \leq j)$ is the number of "success" (correctly guessing the password) in the j-th interaction. Furthermore, we set $f(r; j)$ as a function to evaluate the number of possible password candidates in the j-th interaction. The expectation value of $f(r; j)$ in a variable j is denoted by $E^{(\mathsf{on})}\{f(r; j)\}$. For a single discrete variable, it is defined as

$$E^{(\mathsf{on})}\{f(r; j)\} = \sum_{r=0}^{j} f(r; j) P(r; j) \ . \tag{4}$$

Without any interaction with a party ($\mathcal{C}$ or $\mathcal{S}$), $E^{(\mathsf{on})}\{f(r; 0)\} = N \cdot 1/N = 1$ where N is the number of all of the password candidates each of which has probability of $1/N$ to be the correct password. For any j, we can get the expectation value of the number of possible password candidates in the j-th interaction as the following:

$$E^{(\mathsf{on})}\{f(r; j)\} = \sum_{r=0}^{j} \left((N - j + r) \times_j C_r \left(\frac{1}{N}\right)^r \left(1 - \frac{1}{N}\right)^{j-r} \right) \simeq N - j \tag{5}$$

since $1/N \simeq 0$ in practice[3]. Equation (5) means that the remaining password candidates decreases linearly from N according to j. If an adversary runs N-th interaction with a party, she can sieve out the correct password pw anyway.

[3] $N = 2^{37}$ for MS-Windows passwords.

Theorem 2. *For $e = 3$,*

$$\max_{\forall j} \left\{ \begin{array}{ll} l = 0, & \textit{for } j = 0; \\ l \geq \left\lceil -\log_3 \left(1 - \sqrt[j]{1 - \frac{j}{N}}\right) \right\rceil, & \textit{for } j \ (1 \leq j \leq N). \end{array} \right\} \tag{6}$$

where j denotes the j-th interaction between an adversary and a party.

Proof. Since we deduced the expectation value in case of an on-line attack above, the remaining work is to calculate the counterpart for off-line attack. The e-residue attack can be mounted by an adversary who, impersonating server $\mathcal{S}$, deliberately generates an RSA key pair (e, n) such that $\gcd(e, \varphi(n)) \neq 1$. Here we describe how the e-residue attack works in the RSA-PAKE protocol, followed by its expectation value depending on the form of n.

Case 1. ($n = pq$ such that $e|\varphi(p)$ and $e \nmid \varphi(q)$) Let p and q be distinct primes, such that only $\varphi(p)$ has e as a factor, and $n = pq$. The adversary runs the first three flows with client $\mathcal{C}$ where the probability to pass the validity checks on $\{x_i\}_{1 \leq i \leq l}$ is $(1/3)^l$ by Corollary 1. When client $\mathcal{C}$ transmits $\hat{z}$ that is equivalent to $\mathsf{RSA}_{n,3}(t) \times \mathcal{G}(n, pw)$, the adversary can now try a password candidate pw' from the password space. In order to check whether $z'' \equiv \hat{z}/\mathcal{G}(n, pw')$ is a cubic residue mod n, the adversary has to check if it is a cubic residue mod p and mod q. If $(z'')^{\varphi(p)/3} \equiv 1 \bmod p$, z'' is a cubic residue mod p, where the number of cubic residues is equal to $\varphi(p)/3$, so that the adversary can leave that password pw' as a possible candidate. If z'' is not a cubic residue then the adversary can reject that password and try another password candidate. However, $(z'')^{\varphi(q)} \equiv 1 \bmod q$ by Fermat's theorem and the number of cubic residues is $\varphi(q)$. As a result, the number of cubic residues mod n equals $\varphi(p)/3 \times \varphi(q)$ or $\varphi(n)/3$. Since $1/3$ of the numbers will pass as cubic residues, only $1/3$ of the passwords in the password space will remain as possible candidates.[4] The other passwords will be rejected because those did not yield a cubic residue. Another interaction with client $\mathcal{C}$ allows the adversary to leave $1/3$ of the remaining password candidates with probability of $(1/3)^{2l}$. So at a logarithmic rate in the number of interactions, the adversary can narrow the space of possible candidates down to one while the probability of $(1/3)^l$ decreases to 0.

With respect to the j-th ($1 \leq j \leq N$) interaction between the adversary and client $\mathcal{C}$, the success probability of an e-residue attack follows directly the binomial distribution $P(r; j)$

$$P(r; j) =_j C_r \left(\frac{1}{3}\right)^{l \cdot r} \left(1 - \left(\frac{1}{3}\right)^l\right)^{j-r} \tag{7}$$

where r ($1 \leq r \leq j$) is the number of "success" (correctly passing the validity checks on $\{x_i\}$) in the j-th interaction. We also set $g(r; j)$ as a function to

[4] Note that the number of integers in $\mathbb{Z}_n^\star$ is equal to $\varphi(n)$.

evaluate the number of possible password candidates in the j-th interaction. So the expectation value of $g(r;j)$ in a variable j is defined as

$$E^{(\mathsf{off})}\{g(r;j)\} = \sum_{r=0}^{j} g(r;j)P(r;j) \ . \tag{8}$$

Without any interaction ($j = 0$) with client $\mathcal{C}$, $E^{(\mathsf{off})}\{g(r;0)\} = N \cdot 1/N = 1$ where N is the number of all of the password candidates each of which has probability of $1/N$ to be the correct password. Since $E^{(\mathsf{on})}\{f(r;0)\} = E^{(\mathsf{off})}\{g(r;0)\}$, we don't need any l ($l = 0$). By assuming that $(1/3)^{l+1} < 1/N$, we can get the expectation value of the number of possible password candidates in the j-th interaction as the following:

$$\begin{aligned} E^{(\mathsf{off})}\{g(r;j)\} &= \sum_{r=0}^{j} \left(\left(\frac{1}{3}\right)^{r} N \times_j C_r \left(\frac{1}{3}\right)^{l \cdot r} \left(1 - \left(\frac{1}{3}\right)^{l}\right)^{j-r} \right) \\ &\simeq N \times \left(1 - \frac{1}{3^l}\right)^{j} + \left(\frac{1}{3}\right) N \times j \left(\frac{1}{3^l}\right) \left(1 - \frac{1}{3^l}\right)^{j-1} \\ &\simeq N \times \left(1 - \frac{1}{3^l}\right)^{j} \qquad \because \left(\frac{1}{3^{l+1}} \simeq 0\right) \end{aligned} \tag{9}$$

Now we are ready to deduce the lower-bound of l by bounding the expectation value of the number of possible password candidates in e-residue attack to the counterpart in on-line attack per interaction j. For j ($1 \le j \le N$),

$$\begin{aligned} E^{(\mathsf{on})}\{f(r;j)\} &\le E^{(\mathsf{off})}\{g(r;j)\} \\ N - j &\le N \times \left(1 - \frac{1}{3^l}\right)^{j} \end{aligned} \tag{10}$$

which completes the proof of Theorem 2. The only thing to do is prove $(1/3)^{l+1} < 1/N$. Since Inequality (10) can be re-written as

$$\left(\frac{1}{3}\right)^{l} \le 1 - \left(\frac{N-j}{N}\right)^{\frac{1}{j}} \tag{11}$$

and it is correct for all j ($1 \le j \le N$), we can get $(1/3)^l \le 1/N$. That means,

$$\left(\frac{1}{3}\right)^{l+1} < \left(\frac{1}{3}\right)^{l} \le \frac{1}{N} \ . \tag{12}$$

Case 2. ($n = pq$ such that $e|\varphi(p)$ and $e|\varphi(q)$) Let p and q be distinct primes, such that both $\varphi(p)$ and $\varphi(q)$ have e as a factor, and $n = pq$. The maximum probability to pass the validity checks on $\{x_i\}_{1 \le i \le l}$ is irrelevant to the form of n so that $(1/3)^l$ remains unchanged by Corollary 1. In the same way as Case 1, with $\hat{z}$ the adversary can try a password candidate pw' in

order to check whether $z'' \equiv \hat{z}/\mathcal{G}(n, pw')$ is a cubic residue mod n or not. If $(z'')^{\varphi(p)/3} \equiv 1 \bmod p$ and $(z'')^{\varphi(q)/3} \equiv 1 \bmod q$, z'' is a cubic residue mod p and mod q, where the number of cubic residues mod n is equal to $\varphi(p)/3 \times \varphi(q)/3$ or $\varphi(n)/9$, so that the adversary can leave that password pw' as a possible candidate. Since 1/9 numbers will pass as cubic residues, only 1/9 passwords of the password space will remain as possible candidates. The other passwords will be rejected because those did not yield a cubic residue. Of course, another interaction with client $\mathcal{C}$ allows the adversary to leave 1/9 of the remaining password candidates with probability of $(1/3)^{2l}$.

When $j = 0$, we can easily see that $l = 0$ due to $E^{(\mathsf{on})}\{f(r;0)\} = E^{(\mathsf{off})}\{g(r;0)\}$. With minor changes in Case 1, we can get the expectation value of the number of possible password candidates in each interaction of an e-residue attack as the following:

$$E^{(\mathsf{off})}\{g(r;j)\} = \sum_{r=0}^{j} \left(\left(\frac{1}{9}\right)^r N \times_j C_r \left(\frac{1}{3}\right)^{l \cdot r} \left(1 - \left(\frac{1}{3}\right)^l\right)^{j-r} \right) . \quad (13)$$

By assuming $(1/3)^{l+1} < 1/N$, the above expectation value becomes to the same as Equation (9)

$$\begin{aligned} E^{(\mathsf{off})}\{g(r;j)\} &\simeq N \times \left(1 - \frac{1}{3^l}\right)^j + \left(\frac{1}{9}\right) N \times j \left(\frac{1}{3^l}\right) \left(1 - \frac{1}{3^l}\right)^{j-1} \\ &\simeq N \times \left(1 - \frac{1}{3^l}\right)^j \qquad \because \left(\frac{1}{3^{l+2}} \simeq 0\right) \end{aligned} \quad (14)$$

and the same lower-bound of l can be obtained. Therefore, showing $(1/3)^{l+1} < 1/N$ is trivial.

Case 3. (Generalized $n = p_1^{a_1} p_2^{a_2} \cdots p_m^{a_m}$ such that $e|\varphi(p_i^{a_i})$ for i $(1 \le i \le m)$) For every positive integer n, we can express n as a product of non-trivial powers of distinct primes $n = p_1^{a_1} p_2^{a_2} \cdots p_m^{a_m}$ where p_i are primes and a_i are positive integers for i $(1 \le i \le m)$. By rearranging the prime powers, n has a unique prime-power factorization. W.l.o.g., we proceed with $n = p_1^{a_1} p_2^{a_2} \cdots p_m^{a_m}$ where p_i $(1 \le i \le m)$ are pairwise relatively prime (due to the unique factorization of n) and all the $\varphi(p_i^{a_i})$s have e as a factor. Let $n_i = p_i^{a_i}$ $(1 \le i \le m)$. The maximum probability to pass the validity checks is $(1/3)^l$ by Corollary 1. In the same way as Case 1 and Case 2, with $\hat{z}$ and a password candidate pw' the adversary can check whether or not $z'' \equiv \hat{z}/\mathcal{G}(n, pw')$ is a cubic residue mod n. For that, the adversary has to check if $(z'')^{\varphi(n_i)/3} \equiv 1 \bmod n_i$.[5] If $(z'')^{\varphi(n_i)/3} \equiv 1 \bmod n_i$ $(1 \le i \le m)$, z'' is a cubic residue mod n_i, where the number of cubic residues mod n is equal to $\prod_{i=1}^{m} \varphi(n_i)/3$ (i.e., $\varphi(n)/3^m$), so that

[5] Since p_i is an odd integer and n_i always possesses a primitive root [18], z'' is a cubic residue mod n_i iff $(z'')^{\varphi(n_i)/b} \equiv 1 \bmod n_i$, where $b = \gcd(e, \varphi(n_i))$.

the adversary can leave that password pw' as a possible candidate.[6] Since $1/3^m$ numbers will pass as cubic residues, only $1/3^m$ passwords of the password space will remain as possible candidates. The other passwords will be rejected because those did not yield a cubic residue. Of course, another interaction with client $\mathcal{C}$ allows the adversary to leave $1/3^m$ of the remaining password candidates with probability of $(1/3)^{2l}$.

When $j = 0$, $l = 0$ due to $E^{(\mathsf{on})}\{f(r;0)\} = E^{(\mathsf{off})}\{g(r;0)\}$. As in Case 2, we can get the expectation value of the number of possible password candidates in each interaction of an e-residue attack as the following:

$$E^{(\mathsf{off})}\{g(r;j)\} = \sum_{r=0}^{j} \left(\left(\frac{1}{3^m}\right)^r N \times_j C_r \left(\frac{1}{3}\right)^{l \cdot r} \left(1 - \left(\frac{1}{3}\right)^l\right)^{j-r} \right) . \quad (15)$$

By assuming $(1/3)^{l+1} < 1/N$, we can obtain not only the same expectation value as Equation (9) but also the same lower-bound of l.

$$\begin{aligned} E^{(\mathsf{off})}\{g(r;j)\} &\simeq N \times \left(1 - \frac{1}{3^l}\right)^j + \left(\frac{1}{3^m}\right) N \times j \left(\frac{1}{3^l}\right) \left(1 - \frac{1}{3^l}\right)^{j-1} \\ &\simeq N \times \left(1 - \frac{1}{3^l}\right)^j \qquad \because \left(\frac{1}{3^{l+m}} \simeq 0\right) \end{aligned} \quad (16)$$

Therefore, showing $(1/3)^{l+1} < 1/N$ is trivial. □

3.2 The Other Candidates of e

Although we have used 3 as an example of a low factor repeatedly, the other candidates of e (e.g., 5, 7, $2^{16} + 1$ or higher factors) could have been used to speed-up an e-residue attack: the higher the factor, the larger the subset of passwords that are rejected at each interaction. However, from Corollary 1 we can state that the maximum probability to pass the validity checks on $\{x_i\}$ remains unchanged regardless of whatever the form of e and n. Additionally, as shown in Case 1, Case 2 and Case 3 we can prove that any odd integer e doesn't affect the lower-bound of l. Consequently, Theorem 2 holds for any e ($e \geq 3$).

4 The Exact Complexities of the RSA-PAKE Protocol

Since the PAKE protocols have been motivated by the requirement to be very practical implementations and RSA-based ones particularly have been designed

[6] If the p_is are pairwise relatively prime, then each prime power $n_i = p_i^{a_i}$ of the factorization of $n = p_1^{a_1} p_2^{a_2} \cdots p_m^{a_m}$ is also pairwise relatively prime and the system of simultaneous congruences $y \equiv x \bmod n_i$ $(1 \leq i \leq m)$ has a unique solution mod n by the Chinese Remainder Theorem.

Table 1. Comparison of PAKE protocols, satisfying perfect forward secrecy

Protocols based on		Computation complexity of client $\mathcal{C}$	Communication complexity	The number of flows*1
RSA	RSA-PAKE*2	$25Squ + 25Mul$	3476 Bytes	5
	PEKEP*3 [22]	$(\lfloor \log_3 n \rfloor + 1)\, Squ$ $+ (\lfloor \log_3 n \rfloor + 1)\, Mul$	316 Bytes	3
	CEKEP [22]	$160Squ + 82Mul$ *4	485 Bytes	5
	SNAPI [16]	$512Squ + 257Mul$ *5	380 Bytes	3
DH	OMDHKE*6 [2]	$320Squ + 162Mul$, $(160\ Squ + 81\ Mul)$	276 Bytes	2

*1: For unilateral authentication
*2: This is the case of N ($2^{36} \leq N < 2^{37}$) and $e = 3$.
*3: This is the case of $e = 3$ when $e \geq 3$.
*4: $2\left|e^{\lceil -\log_e \omega \rceil}\right|$ where $e = 3$ ($e \geq 3$) and $\omega = 2^{-80}$ ($0 < \omega \leq 2^{-80}$)
*5: The average complexity when $|e| = 512$. The primality test for large e is needed as well.
*6: Public parameters for the Diffie-Hellman protocol: let $\mathbb{G}$ be a finite, cyclic group of prime order q and g be a generator of $\mathbb{G}$. Here we assume that $|q| = 160$.

Table 2. Comparison of RSA-based PAKE protocols when (e, n) is fixed and pre-computation is allowed

Protocols	Computation complexity of client $\mathcal{C}$	Communication complexity	The number of flows
RSA-PAKE	0	148 Bytes	2
PEKEP [22]	$\lfloor \log_3 n \rfloor\, Squ + \lfloor \log_3 n \rfloor\, Mul$		
CEKEP [22]	$80Squ + 41Mul$ *1		
SNAPI [16]	0*2		

*1: $\left|e^{\lceil -\log_e \omega \rceil}\right|$ where $e = 3$ ($e \geq 3$) and $\omega = 2^{-80}$ ($0 < \omega \leq 2^{-80}$)
*2: The primality test for large e is needed.

for efficiency, we analyze the computation and communication complexities of the RSA-PAKE protocol. As for computation complexity of client $\mathcal{C}$, the number of modular exponentiations is a major factor to evaluate efficiency of a cryptographic protocol because that is the most power-consuming operation. In order to differentiate the computation cost of when e is small or large, we instead count the number of modular squarings (Squ) and multiplications (Mul) with the "repeated square-and-multiply" exponentiation algorithm [15]. As for communication complexity, the length of identities is excluded and $|\cdot|$ indicates its bit-length.

From Theorem 2, we can get the lower-bound of l according to the cardinality of passwords N. For the minimum security parameters recommended for use in current practice: $|k| = 1024$ (for RSA, Diffie-Hellman protocol and random

numbers) and $|l_j| = 160$ (for hash functions). We show the numerical complexity as follows.

- When N ($2^{36} \leq N < 2^{37}$) for alphanumerical passwords with 6 characters, $l = 24$: the computation complexity becomes ($25Squ + 25Mul$) and the communication complexity is around 3476 Bytes in case of $e = 3$.

In Table 1, we compare the RSA-PAKE protocol with the previous RSA-based [16,22] and the most efficient Diffie-Hellman based [2] PAKE ones all of which are provably secure in the random oracle model and achieve perfect forward secrecy. The figure in the parentheses of the OMDHKE protocol is the remaining computation cost after pre-computation. One can see that the RSA-PAKE protocol is indeed more efficient rather than [16,2,22] with respect to computation complexity. However, if the bandwidth of communication is restricted, OMDHKE [2] can be a good candidate over the others. Furthermore, we also compare the RSA-based PAKE protocols in Table 2 when the RSA public key (e, n) is fixed (and cached) and pre-computation is allowed. Note that fixed RSA keys no longer provide perfect forward secrecy. In the RSA-PAKE and SNAPI protocols, the client is not required to compute any modular exponentiation.

Acknowledgements

The authors would like to thank anonymous reviewers for their helpful comments on this paper. The footnote 1 comes from one of the reviewers.

References

1. F. Bao. Security Analysis of a Password Authenticated Key Exchange Protocol. In *Proc. of ISC 2003*, LNCS 2851, pages 208-217. Springer-Verlag, 2003.
2. E. Bresson, O. Chevassut, and D. Pointcheval. New Security Results on Encrypted Key Exchange. In *Proc. of PKC 2004*, LNCS 2947, pages 145-158. Springer-Verlag, 2004.
3. S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. of IEEE Symposium on Security and Privacy*, pages 72-84. IEEE Computer Society, 1992.
4. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS '93*, pages 62-73, 1993.
5. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Proc. of CRYPTO '93*, LNCS 773, pages 232-249. Springer-Verlag, 1993.
6. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In *Proc. of EUROCRYPT '96*, LNCS 1070, pages 399-416. Springer-Verlag, 1996.
7. D. Catalano, D. Pointcheval, and T. Pornin. IPAKE: Isomorphisms for Password-based Authenticated Key Exchange. In *Proc. of CRYPTO 2004*, Springer-Verlag, LNCS 3152, pages 477-493. Springer-Verlag, 2004. The full version is available at `http://www.di.ens.fr/~pointche/slides.php?reference=CaPoPo04`.

8. A. Frier, P. Karlton, and P. Kocher. The SSL 3.0 Protocol. Netscape Communication Corp., 1996. Available at `http://wp.netscape.com/eng/ssl3/`.
9. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). IETF RFC 2409, November 1998. Available at `http://www.ietf.org/rfc/rfc2409.txt`.
10. R. Housley and T. Polk. Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure. WILEY. March 2001.
11. IETF (Internet Engineering Task Force). Secure Shell (secsh) Charter. Available at `http://www.ietf.org/html.charters/secsh-charter.html`.
12. IETF (Internet Engineering Task Force). Transport Layer Security (tls) Charter. Available at `http://www.ietf.org/html.charters/tls-charter.html`.
13. C. Kaufman. Internet Key Exchange (IKEv2) Protocol. May 2003 (to be published as an RFC). Available at `draft-ietf-ipsec-ikev2-03.txt`.
14. S. Lucks. Open Key Exchange: How to Defeat Dictionary Attacks without Encrypting Public Keys. In *Proc. of Workshop on Security Protocols*, 1997.
15. A. J. Menezes, P. C.van Oorschot, and S. A. Vanstone. Handbook of Applied Cryptography. pages 613-616. CRC Press, 1997.
16. P. MacKenzie, S. Patel, and R. Swaminathan. Password-Authenticated Key Exchange Based on RSA. In *Proc. of ASIACRYPT 2000*, LNCS 1976, pages 599-613. Springer-Verlag, 2000, A full version is available at `http://cm.bell-labs.com/who/philmac/bib.html`.
17. S. Patel. Number Theoretic Attacks on Secure Password Schemes. In *Proc. of IEEE Symposium on Security and Privacy*, IEEE Computer Society, pages 236-247, 1997.
18. K. H. Rosen. Elementary Number Theory and Its Applications. 4th Edition, Addison Wesley Longman, 2000.
19. V. Shoup. On Formal Models for Secure Key Exchange. IBM Research Report RZ 3121, 1999. Available at `http://eprint.iacr.org/1999/012`.
20. D. S. Wong, A. H. Chan, F. Zhu. More Efficient Password Authenticated Key Exchange Based on RSA. In *Proc. of INDOCRYPT 2003*, LNCS 2904, pages 375-387. Springer-Verlag, 2003.
21. M. Zhang. Further Analysis of Password Authenticated Key Exchange Protocol based on RSA for Imbalanced Wireless Networks. In *Proc. of ISC 2004*, LNCS 3225, pages 13-24. Springer-Verlag, 2004.
22. M. Zhang. New Approaches to Password Authenticated Key Exchange based on RSA. In *Proc. of ASIACRYPT 2004*, LNCS 3329, pages 230-244. Springer-Verlag, 2004. Cryptology ePrint Archive, Report 2004/033, available at `http://eprint.iacr.org/2004/033`.
23. F. Zhu, D. S. Wong, A. H. Chan, and R. Ye. Password Authenticated Key Exchange Based on RSA for Imbalanced Wireless Networks. In *Proc. of ISC 2002*, LNCS 2433, pages 150-161. Springer-Verlag, 2002.

Recoverable and Untraceable E-Cash

Joseph K. Liu[1], Patrick P. Tsang[1], and Duncan S. Wong[2]

[1] Department of Information Engineering,
The Chinese University of Hong Kong
Shatin, Hong Kong
{ksliu, pktsang3}@ie.cuhk.edu.hk
[2] Department of Computer Science,
The City University of Hong Kong
Hong Kong
duncan@cityu.edu.hk

Abstract. In an electronic cash (e-cash) system, *Recoverability* means once you have lost your e-cash, you still can get back the amount of e-cash that you have lost. *Untraceability* means no one can trace where and when you have spent your e-cash. Obviously these are conflicting properties in an e-cash system. Most of the e-cash systems proposed in the literature do not include recoverability. Although some of them such as [17] contain recoverability, it is an on-line e-cash system. In this paper, we propose a new efficient e-cash protocol which possesses these two properties simultaneously. At the same time, it still remains off-line.

Keywords: E-Cash, Recoverable, Untraceable.

1 Introduction

Electronic payment systems allow people to carry out commercial activities in an electronic domain. There are many electronic payment systems that have been proposed in recent years. Generally they can be divided into two main categories: Credit-based and Debit-based. Credit cards are an example of credit-based system while electronic cash (e-cash) is an example of debit-based system. Credit-based systems allow users to get the service or the goods from the merchant first, then they pay for it later within a certain period. This kind of system requires the bank to be online. That is, the merchant needs to communicate with the bank interactively when users require the service or request the products. At the same time, the bank records down all the transaction records of its users. As a result, it is not anonymous.

Debit-based systems, in contrast, require users to prepay first and then get the service or goods from the merchant later. The main advantage is that the bank does not need to be online. In other words, the merchant does not need to communicate with the bank when users want to buy their goods. It just simulates our daily life in the paper cash world. Suppose you want to buy a book from a shop, the shop does not need to contact the bank if you pay by

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 206–214, 2005.

cash. Furthermore, the whole process is anonymous, or we call it untraceable. The bank cannot trace the owner of any e-cash it received from the merchant.

E-cash is a kind of debit-based payment system. Usually it contains three main protocols. In the withdrawal protocol, users get the e-cash from the bank while the bank debits their corresponding accounts. In the payment protocol, users transfer the e-cash to the merchants in exchange for the goods or service from them. In the deposit protocol, merchants send their received e-cash (from users) to the bank. The bank checks the validity of the e-cash and credits the merchants' accounts.

Obviously it is easier to implement a credit-based system than a debit-based system since most of the security checking can be done online. The most difficult task for an off-line credit-based system is the detection of double spending. For example, it requires that if a dishonest user double spends an e-cash token, his identity will be revealed.

According to [14], an ideal e-cash system should have six main properties: Independence, Security, Privacy (Untraceability), Off-line payment, Divisibility and Transferability. *Independence* implies that the security of an e-cash system does not depend on any physical location, medium, time or users. *Security* means that e-cash coins cannot be forged or double spent without being detected. *Untraceability* refers to the maintenance of the anonymity of any honest user. *Offline payment* does not need the bank to be involved during the payment process conducted between a customer and a merchant. *Divisibility* refers to the ability to divide an e-coin into smaller pieces provided that the total amount of those pieces equals the value of the original e-coin. *Transferability* allows a user to spend an e-coin received in a prior payment immediately without having to contact the bank first. Most of the previous papers focused on these areas and provided improvement on these topics. However, even using an ideal e-cash system, a user will lose all his e-cash if his computer crashes or relevant files are corrupted accidentally. Similarly, if his e-cash wallet, for example, a smart card, is lost, he also cannot get back the lost e-cash. In the latter case, someone else who picks up this wallet can spend all the remaining e-cash in a normal way. This situation will not happen if the user uses his credit card to go on-line shopping. If the user loses his credit card, or recognizes his credit card has been stolen, he can report this to the bank so that the credit card becomes invalid at the same moment. The tradeoff is that each transaction has to be online. In other words, a three-party connection has to be set up. At the same time the user's anonymity is lost. Both the shop and the bank can obtain the identity of the user which may not be desirable in some cases.

It is very easy for any e-cash system to support recoverability if it is traceable. Just by using a large database to store all the transaction records, the bank can easily find out the amount in difference between the withdrawal and deposit protocol of a particular user. The difference is the amount that the user has lost. However, the situation becomes more complex if we want to maintain untraceability. In this case, the bank does not know how much e-cash a user has spent, providing that the user does not double spend his e-cash.

Surprisingly there exists very few papers in the literature that discuss e-cash with recoverability. [17] contains recoverability, however it is an on-line e-cash system. In fact, there is a naive method to do recoverability in any e-cash system. A user can backup all the e-coins (that is, all the transcripts he has got from the bank). When he needs to do the recovery, he simply shows all the transcripts to the bank. The bank can then find out the e-coins he has spent from its database. Finally it can compute the remaining amount, that is, the amount he has lost. However, this naive method is neither practical nor scalable. If the number of e-coins a user withdraw is very large, for example, 10000, then the size of the transcript will be very large. Transmission and storage would be a major problem.

1.1 Related Work

Since the invention of e-cash by Chaum [5], there has been lots of research about electronic cash systems. Most of the papers such as [7,8,2,13,1,16] only focused on the basic properties of e-cash, such as untraceability and divisibility. Recently, fair e-cash [3,18,9,10,12,11] has been suggested in order to reduce the anonymity so that some kinds of fraudulent activities can be prevented. Some of the implementations use group signatures [6] to instantiate since group signature allows a group manager to revoke the identity of a misbehaved user within a group. However, they do not address the recoverability matter. [17] contains recoverability, but it is an on-line e-cash system.

1.2 Our Contributions

In this paper, we propose a new e-cash protocol that can support both recoverability and untraceability. At the same time, it remains off-line. That is, it combines the advantages of a debit-based and a credit-based system together. Moreover, it is very efficient and the physical bandwidth is only a small constant no matter how many e-coins a user withdraws each time. Thus it is practical to be used in daily life.

1.3 Organizations

The rest of the paper is organized as follow: Brand's e-cash protocol will be reviewed in Sec. 2. It is followed by our proposed e-cash system in Sec. 3. Some extensions and discussions are given in Sec. 4. Finally a conclusion is made in Sec. 5.

2 Review of Brands' E-Cash Protocol

Our proposed protocol is motivated from Brands' E-cash Protocol [2]. Therefore hereby we are going to give a brief introduction of his protocol.

2.1 Setup

The system consists of the bank $\mathcal{B}$, the user $\mathcal{U}$ and the shop $\mathcal{S}$. The bank also sets two databases: One is called *Account Database* which is used to store information about account-holders. The other is called *Deposit Database* which is used to store information from deposited payment transcripts. Let p, q, g, g_1, g_2 be publicly known system parameters published by the bank such that p, q are large prime and $g, g_1, g_2 \in \mathbb{Z}_p^*$ have order q. Let $x \in \mathbb{Z}_q$ be the secret key of $\mathcal{B}$ and $h = g^x \bmod p$ be the corresponding public key. Let u_1 be the secret key of $\mathcal{U}$ and $I = {g_1}^{u_1} \bmod p$ be the identity of the customer. $\mathcal{B}$ computes $z = (Ig_2)^x \bmod p$ and transmits it to $\mathcal{U}$.

2.2 Withdrawal Protocol

If $\mathcal{U}$ wants to withdraw a coin, he has to identify himself to the bank first. For example, digitally sign a request for withdrawal, or type in the correct password of his account. After that, the following steps are performed:

1. $\mathcal{B}$ randomly generates $w \in_R \mathbb{Z}_q$ and sends $a = g^w \bmod p$ and $b = (Ig_2)^w \bmod p$ to $\mathcal{U}$.
2. $\mathcal{U}$ randomly generates $s, x_1, x_2, u, v \in_R \mathbb{Z}_q$ and computes

$$\begin{aligned} A &= (Ig_2)^s \bmod p \\ B &= {g_1}^{x_1} {g_2}^{x_2} \bmod p \\ z' &= z^s \bmod p \\ a' &= a^u g^v \bmod p \\ b' &= b^{su} A^v \bmod p \end{aligned}$$

   He also computes the challenge $c' = H(A, B, z', a', b')$ where $H : \{0,1\}^* \to \mathbb{Z}_p$ is a cryptographic hash function. Then he sends $c = c'/u \bmod q$ to $\mathcal{B}$.
3. $\mathcal{B}$ sends the response $r = cx + w \bmod q$ to $\mathcal{U}$ and debits his account.
4. $\mathcal{U}$ checks whether

$$g^r = h^c a \bmod p \qquad \text{and} \qquad (Ig_2)^r = z^c b \bmod p$$

   If both equalities hold, $\mathcal{U}$ accepts and computes $r' = ru + v \bmod q$. Otherwise, he rejects.
5. $\mathcal{U}$ checks whether

$$\begin{aligned} g^{r'} &= h^{c'} a' \bmod p \\ A^{r'} &= (z')^{c'} b' \bmod p \\ c' &= H(A, B, z', a', b') \end{aligned}$$

   If all verifications pass, he stores $\{A, B, \mathsf{Sign}(A, B)\}$ where $\mathsf{Sign}(A, B) = \{z', a', b', r'\}$.

2.3 Payment Protocol

When $\mathcal{U}$ wants to spend his e-cash at shop $\mathcal{S}$, the following protocol is performed:

1. $\mathcal{U}$ sends $A, B, \mathsf{Sign}(A, B)$ to $\mathcal{S}$.
2. If $A \neq 1$, then $\mathcal{S}$ computes the challenge $d = H_0(A, B, I_S, \mathsf{date/time})$, where $\mathsf{date/time}$ is the information representing date and time of the transaction and I_S is the identity of the shop $\mathcal{S}$ and $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a cryptographic hash function. $\mathcal{S}$ sends d to $\mathcal{U}$.
3. $\mathcal{U}$ computes the responses

$$r_1 = d(u_1 s) + x_1 \bmod q$$
$$r_2 = ds + x_2 \bmod q$$

   and sends r_1, r_2 back to $\mathcal{S}$.
4. $\mathcal{S}$ checks whether $\mathsf{Sign}(A, B)$ is a signature on (A, B) and $g_1{}^{r_1} g_2{}^{r_2} = A^d B \bmod p$. If all pass, it accepts. Otherwise, it rejects.

2.4 Deposit Protocol

After a period of time (for example, at the end of each day), $\mathcal{S}$ sends $\mathcal{B}$ the payment transcript, consisting of $A, B, \mathsf{Sign}(A, B), (r_1, r_2)$ and $\mathsf{date/time}$ of the transaction. The following protocol is performed by the bank:

1. If $A = 1$, $\mathcal{B}$ rejects the payment transcript. Otherwise $\mathcal{B}$ computes $d = H_0(A, B, I_S, \mathsf{date/time})$
2. $\mathcal{B}$ checks whether $\mathsf{Sign}(A, B)$ is a signature on (A, B) and $g_1{}^{r_1} g_2{}^{r_2} = A^d B \bmod p$. If not both valid, it rejects.
3. $\mathcal{B}$ searches its deposit database to find out whether A has been stored before. If not, $\mathcal{B}$ stores $(A, \mathsf{date/time}, r_1, r_2)$ in the deposit database deposited by $\mathcal{S}$ and credits the account of $\mathcal{S}$.
4. Otherwise, that is, A has been deposited before, it means a fraud has occurred. If the previous transcript was deposited by $\mathcal{S}$ and the $\mathsf{date/time}$ is the same as the newly deposited transcript, that means $\mathcal{S}$ is trying to deposit the same transcript twice. If not, the coin has been double-spent.
5. If the coin has been double-spent, $\mathcal{B}$ computes the identity of the double-spent user:
   - Let (d, r_1, r_2) be the newly deposited payment transcript and (d', r_1', r_2') be the previous deposited payment transcript.
   - Solve the following linear equations for u_1:

$$r_1 = d(u_1 s) + x_1 \bmod q$$
$$r_1' = d'(u_1 s) + x_1 \bmod q$$
$$r_2 = d(s) + x_2 \bmod q$$
$$r_2' = d'(s) + x_2 \bmod q$$

   - Compute $I = g^{u_1} \bmod p$ which is the identity of the double-spent user.

3 The Proposed Protocol

In this section, we describe our proposed protocol. Here we use the same notation as in Sec. 2. In addition to the three entities $\mathcal{B}$ (Bank), $\mathcal{U}$ (User) and $\mathcal{S}$ (Shop), another entity $\mathcal{RC}$ (Recovery Centre) is introduced here. It is responsible for issuing recovery information which is necessary for the e-cash recovery protocol. We assume that $\mathcal{U}$ communicates with $\mathcal{RC}$ through an anonymous channel [4,15].

3.1 Withdrawal Protocol

After executing the withdrawal protocol described in Sec. 2.2, $\mathcal{U}$ gets the e-cash $\{A_i, B_i, \mathsf{Sign}(A_i, B_i)\}$, for $1 \leq i \leq n$, where n is the total number of e-coins $\mathcal{U}$ withdrawn last time. Practically, it can be a constant number set by $\mathcal{B}$. Then $\mathcal{U}$ communicates with $\mathcal{RC}$ (through an anonymous channel) to perform the following protocol in order to complete the withdrawal process:

1. $\mathcal{RC}$ prepares n numbers $x_1, \ldots, x_n$ such that

$$H_n(x_1) = \ldots = H_n(x_n) = y$$

   where $H_n : \{0,1\}^* \rightarrow \{0,1\}^*$ is a n-collision cryptographic hash function.
2. $\mathcal{U}$ sends $\{A_i, B_i, \mathsf{Sign}(A_i, B_i)\}$, $1 \leq i \leq n$, to $\mathcal{RC}$. It records down all the serial number of these n e-coins (in this case, it is A_i). $\mathcal{RC}$ checks whether these coins have been processed or not. If yes, it terminates. Otherwise, it continues.
3. $\mathcal{RC}$ checks the signature $\mathsf{Sign}(A_i, B_i)$ on (A_i, B_i). If it is valid, it computes another signature $S_{c_i} = \mathsf{Sign}_{\mathcal{RC}}\{A_i, B_i, \mathsf{Sign}(A_i, B_i), x_i\}$ where $\mathsf{Sign}_{\mathcal{RC}}(m)$ denotes a signature generated by $\mathcal{RC}$ on a message m. $\mathcal{RC}$ sends x_i, S_{c_i} back to $\mathcal{U}$. $\mathcal{U}$ attaches x_i, S_{c_i} to the e-coin $\{A_i, B_i, \mathsf{Sign}(A_i, B_i)\}$.
4. $\mathcal{RC}$ computes another signature $S_b = \mathsf{Sign}_{\mathcal{RC}}\{y, n\}$ and sends S_b, y back to $\mathcal{U}$. $\mathcal{U}$ should save S_b, y in a safe place for recovery purpose.

3.2 Payment Protocol

When $\mathcal{U}$ spends her e-coin $\{A_i, B_i, \mathsf{Sign}(A_i, B_i), x_i, S_{c_i}\}$ at the shop $\mathcal{S}$, the following protocol is performed:

1. $\mathcal{U}$ and $\mathcal{S}$ execute the payment protocol described in Sec. 2.3.
2. $\mathcal{S}$ checks if S_{c_i} is a valid signature on $\{A_i, B_i, \mathsf{Sign}(A_i, B_i), x_i\}$. If no, terminates. Otherwise, continues.
3. $\mathcal{S}$ computes $y = H'(x_i)$ and checks whether y is in the $\mathsf{blacklist}$, where $\mathsf{blacklist}$ is a list containing the *H'-hash value* of x_i of all lost coins published by $\mathcal{B}$. $\mathcal{S}$ accepts only if it is not in the $\mathsf{blacklist}$.

3.3 Deposit Protocol

After some period of time (for example, at the end of each day), $\mathcal{S}$ sends all the payment transcripts to $\mathcal{B}$. In addition to the checking procedures described in Sec. 2.4, $\mathcal{B}$ further checks whether the *H'-hash value* of x_i of each e-coin is in the blacklist. If yes, $\mathcal{B}$ rejects it. Otherwise, it accepts.

3.4 Recovery Protocol

If $\mathcal{U}$ lost the un-used e-coins, he has to execute the following protocol:

1. $\mathcal{U}$ has to reveal his identity to $\mathcal{B}$. $\mathcal{U}$ sends $\{S_b, y\}$ to $\mathcal{B}$.
2. $\mathcal{B}$ checks whether S_b is a valid signature on $\{y, n\}$. It continues if it is valid. Otherwise, terminates.
3. $\mathcal{B}$ looks up its database to find out all the e-coins with their H'-hash value are equal to y. The maximum number of such e-coins should be n. These e-coins are those $\mathcal{U}$ has already spent.
4. $\mathcal{B}$ computes the difference D between the total amount of e-coins $\mathcal{U}$ has withdrawn and the total amount he has spent. Thus D is the value that $\mathcal{U}$ has lost.

In order to prevent $\mathcal{U}$ pretending to lose some e-coins when in fact he does not, the following steps should be taken immediately after he has reported his lost coins to $\mathcal{B}$:

1. $\mathcal{B}$ adds y to the blacklist and publishes it and notifies all shops.
2. If any customer uses an e-coin with H'-hash value of x_i equal to y, $\mathcal{S}$ should not accept this transaction.

4 Extensions and Discussions

4.1 Extensions

Based on the basic system proposed in Sec. 3, there can be some extensions to improve the system in different ways:

Extension 1: There can be different values of n. For example, n can be $5, 10, 20$ so that users can choose the total number of e-coins they withdraw each time. Different hash functions should be used in order to facilitate this extension.

Extension 2: The proposed system with recoverability can be integrated with the original system without recoverability. For those users who want to have an *insurance* of recoverability, they may need to pay some fee or charges to the bank $\mathcal{B}$ or the recovery centre $\mathcal{RC}$ since there is some additional workload for allowing recovery. It can be seen as an insurance service. For those who do not want to have this insurance service, $\mathcal{RC}$ simply just puts 0 for all values of x_i and y. After then, when $\mathcal{S}$ or $\mathcal{B}$ notices these 0, they just execute the original protocol. By this way, the system becomes more general and practical.

4.2 Discussions

We have extended the e-cash protocol proposed by Brands to support recoverability and untraceability. However, our approach is not only limited to Brands' protocol. Instead, it can be also applied to other e-cash systems, provided that they are not divisible and transferable.

We regard it as an interesting open problem to construct an off-line e-cash system that can support recoverability, untraceability, divisibility and transferability at the same time.

5 Conclusions

In this paper, we have proposed a new e-cash protocol which supports recoverability and untraceability at the same time. That is, it allows users to recover their lost e-cash while maintaining anonymity provided that they have not double-spent their e-cash. Furthermore, it is more efficient than the naive method. Thus it is absolutely practical to be implemented for commercial use. We believe in the near future, our proposed recoverable and untraceable e-cash will have a high chance to be commercialized into the industrial sector.

References

1. Y. Frankel A. Chan and Y. Tsiounis. Easy come - easy go divisible cash. In *Proc. EUROCRYPT 98*, pages 561–575. Springer-Verlag, 1998. LNCS Vol. 1403.
2. S. Brands. Untraceable off-line cash in wallet with observers. In *Proc. CRYPTO 93*, pages 302–318. Springer-Verlag, 1994. LNCS Vol. 773.
3. E. Brickell, P. Gemmel, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In *6th ACM-SIAM*, pages 457–466. ACM Press, 1995.
4. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
5. D. Chaum. Blind signatures for untraceable payments. In *Proc. CRYPTO 82*, pages 199–203. Plenum Press, 1982.
6. D. Chaum and E. Van Heyst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. LNCS Vol. 547.
7. A. Fiat D. Chaum and M. Naor. Untraceable electronic cash. In *Proc. CRYPTO 88*, pages 319–327. Springer-Verlag, 1990. LNCS Vol. 403.
8. N. Ferguson. Single term off-line coins. In *Proc. EUROCRYPT 93*, pages 318–328. Springer-Verlag, 1993. LNCS Vol. 765.
9. Y. Frankel, Y. Tsiounis, and M. Yung. Indirect discourse proofs: Achieving efficient fair off-line e-cash. In *Proc. ASIACRYPT 96*, pages 286–300. Springer-Verlag, 1996. LNCS Vol. 1163.
10. Y. Frankel, Y. Tsiounis, and M. Yung. Fair off-line e-cash made easy. In *Proc. ASIACRYPT 98*, pages 257–270. Springer-Verlag, 1998. LNCS Vol. 1514.
11. M. Gaud and K. Trapre. On fair e-cash systems based on group signature schemes. In *ACISP 03*, pages 237–248. Springer-Verlag, 2003. LNCS Vol. 2727.

12. M. Gaud and K. Trapre. On the anonymity of fair off line e-cash systems. In *Financial Cryptography 03*, pages 34–50. Springer-Verlag, 2003. LNCS Vol. 2742.
13. T. Okamoto. An efficient divisible electronic cash scheme. In *Proc. CRYPTO 95*, pages 438–451. Springer-Verlag, 1995. LNCS Vol. 963.
14. T. Okamoto and K. Ohta. Universal electronic cash. In *Proc. CRYPTO 91*, pages 324–337. Springer-Verlag, 1992. LNCS Vol. 576.
15. C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Proc. EUROCRYPT 93*, pages 248–259. Springer-Verlag, 1994. LNCS Vol. 765.
16. T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In *Proc. CRYPTO 99*, pages 555–572. Springer-Verlag, 1999. LNCS Vol. 1666.
17. B. Schoenmakers. Security aspects of the Ecash$^{\text{TM}}$ payment system. In *State of the Art in Applied Cryptography 1997*, pages 338–352. Springer-Verlag, 1998. LNCS Vol. 1528.
18. M. Stadler, J.M. Piveteau, and J. Camenisch. Fair blind signature. In *Proc. EUROCRYPT 95*, pages 209–219. Springer-Verlag, 1995. LNCS Vol. 921.

A Method for Detecting the Exposure of OCSP Responder's Session Private Key in D-OCSP-KIS

Younggyo Lee[1,⋆], Injung Kim[2], Seungjoo Kim[1], and Dongho Won[1]

[1] Department of Computer Engineering, Sungkyunkwan University, Republic of Korea
{yglee, dhwon}@dosan.skku.ac.kr, skim@ece.skku.ac.kr
[2] Electronics and Telecommunication Research Institute, Republic of Korea
ciper@etri.re.kr

Abstract. D-OCSP-KIS proposed by Koga and Sakurai not only reduces the number of OCSP Responder's certificate but also offers the certificate status validation about OCSP Responder to the client. Therefore, D-OCSP-KIS is an effective method that can reduce the communication cost, computational time and storage consumption in client, but it has some problems. In case an attacker accidentally acquires an OCSP Responder's session private key in a time period (e.g., one day), she cannot derive any other OCSP Responder's private key unless she obtains master private key. And she cannot derive the hash value of previous period because the hash value is impossible in inverse computation. But, the attacker can disguise as the OCSP Responder in the time period unless the OCSP Responder recognizes. She can offer the wrong response to the client using the hash value intercepted. And the server and user on E-commerce can have a serious confusion and damage. And the computation and releasing of hash chain can be a load to CA. Thus, we propose a method detecting immediately the exposure of OCSP Responder's session private key and the abuse of hash value in D-OCSP-KIS. In our proposal, the hash value is only used one time for the status validation of OCSP Responder's session private key and the load for computation of X-chain in CA is distributed to each OCSP Responder.

Keywords: D-OCSP, D-OCSP-KIS, OCSP Responder, hash function.

1 Related Works and Motivation

PKI and CRL. PKI (Public Key Infrastructure) is widespread and strong technology for providing the security (integrity, authentication, and non-repudiation) using public key techniques. The main idea of PKI is the digital certificate that is a digitally signed statement binding an entity (user or authority)'s identity

⋆ This work was supported by the University IT Research Center Project funded by the Korean Ministry of Information and Communication.

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 215–226, 2005.

information and his public key. If the entity's private key is compromised or the entity's identity information is changed, the entity makes a request to the CA (Certificate Authority) for revoking its own certificate. The information whether the certificate is revoked or not is called CSI (Certificate Status Information) and the CRL (Certificate Revocation List) is most well-known method for CSI [2,11].

CRL size and communication costs. The well-known CRL systems has advantage of its simplicity, however, it has disadvantage of its high communication costs between the user and the CA's Repository (or Directory) storing CRL. Therefore, in order to reduce the size of certificate revocation list and the communication costs, several methods have been suggested nowadays. There are Delta-CRL, CRL DP(Distributed Points), Over-issued CRL, Indirect CRL, Dynamic CRL DP, Freshest CRL, CRT(Certificate Revocation Tree), NOVOMODO, Authenticated Directory et al[1,2,4,7,9,10,11,12,14].

OCSP and computation costs. If the client or user needs very timely CSI, an online certificate status service such as the OCSP (Online Certificate Status Protocol) is more convenient than off-line method such as CRL et al[7]. In OCSP, because the client does not need to download a CRL from the CA's Repository, the high communication costs between client and the CA's Repository and the storage spaces for storing the CRL are not required. But, if the CSI requests are centralized to an OCSP Responder, OCSP Responder can have a risk of DoS (Denial of Service) attack[13]. For reducing a risk of DoS attack, the OCSP Responder can pre-produce a signed value for response in short time. However this also will give a possibility of replay attack[13,14].

T-OCSP and D-OCSP. For reducing overload of single OCSP Responder in "T-OCSP (Traditional-OCSP)", "D-OCSP (Distributed-OCSP)" is introduced [3,13]. In D-OCSP, if distributed OCSP Responders have a same private key, the possibility of OCSP Responder's private key exposure is very high[14]. Therefore, each OCSP Responder has a different private key generally and clients must have all of OCSP Responder's certificates for verifying CSI response of OCSP Responder. This gives an increase of storage consumption or communication costs for acquiring the OCSP Responder's certificate. For solving the problem, the method of single public key was proposed in D-OCSP-KIS (Distributed OCSP based on Key-Insulated Signature) by Koga and Sakurai[13]. Each OCSP Responder has a different private key but, it has only a same public key. Also for solving problem that the length of the single public key is in proportion to the number of OCSP Responders in D-OCSP-KIS, D-OCSP-IBS(Distributed OCSP based on Identity-Based Signature), where the length of the single public key is constant and short, was proposed by Yum and Lee[3].

Our Contributions. D-OCSP-KIS not only reduces the number of OCSP Responder's certificate but also offers the certificate status validation about OCSP Responder. Therefore, D-OCSP-KIS is an effective method that reduces a communication cost, computational time and storage consumption. However, D-OCSP-KIS has some problems. In case an attacker accidentally acquires an

OCSP Responder's session private key in a time period (e.g., one day), she cannot derive any other OCSP Responder's private key unless she obtains master private key. And she cannot derive the hash value of previous period because the hash value is impossible in inverse computation. But, the attacker can disguise as the OCSP Responder in the time period unless the OCSP Responder recognizes. She can offer the wrong response to the client using the hash value intercepted. Thus, the server and user on E-commerce can have a serious confusion and damage. And the computation and releasing of hash chain can give a load to CA. In this paper, we propose a method for detecting the exposure of OCSP Responder's session private key in D-OCSP-KIS. In our proposal, the hash value is only used one time for the status validation of OCSP Responder and additional load of CA is decreased. The OCSP Responder and clients can detect immediately the exposure of OCSP Responder's session private key and the abuse of hash value in the method. The rest of this paper is organized as follows. In section 2, we analyze the D-OCSP-KIS and show some of its problems. In section 3, we propose a solution to the problems in D-OCSP-KIS. In section 4, we compare the proposed method and other methods. Finally, in section 5, we bring to a conclusion of this paper.

2 D-OCSP-KIS and Its Analysis

In this section, we analyze the D-OCSP-KIS and show some of its problems.

2.1 D-OCSP-KIS

Fig.1 shows the concept of D-OCSP-KIS. It is composed of CA, n-OCSP Responders and client. D-OCSP-KIS uses a one-way hash function H satisfying the following properties.

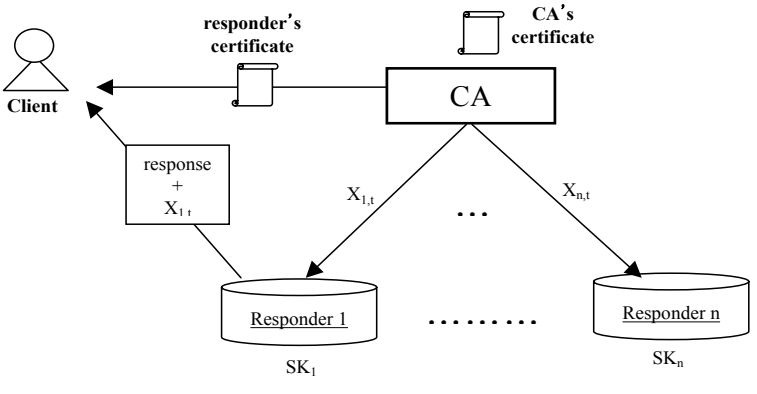

Fig. 1. Concept of D-OCSP-KIS

[One-way hash function]

1. H operation is at least 10,000 times faster in computation than a digital signature operation.
2. H produces 20-byte outputs, no matter how long its inputs are; and
3. It is too hard to find X such that $H(X) = Y$. Finding this solution is practically impossible.

[Issuance of OCSP Responder's certificate]

1. Let T be the total number of time periods. For example, T is 365 if each OCSP Responder's certificate expires 365 days after issuance. CA produces T-hash values using H as follows.

$$X_T \xrightarrow{h} X_{T-1} \xrightarrow{h} X_{T-2} \xrightarrow{h} \ldots \xrightarrow{h} X_1$$

Let n be the total number of OCSP Responders. CA repeatedly produces n hash-chain as different input value $X_{T,i}$. $X_{t,i}$ denotes the hash value in time period t for the status validation of OCSP Responder i's private key . These hash values are stored on the CA.

$$\begin{gathered} X_{T,1} \xrightarrow{h} X_{T-1,1} \xrightarrow{h} X_{T-2,1} \xrightarrow{h} \ldots \xrightarrow{h} X_{1,1} \\ \vdots \\ X_{T,n} \xrightarrow{h} X_{T-1,n} \xrightarrow{h} X_{T-2,n} \xrightarrow{h} \ldots \xrightarrow{h} X_{1,n} \end{gathered}$$

2. The CA issues OCSP Responder's certificate C_{res} as follows by using own private key. SN is the serial number of certificate and V represents the validity period. I and S denote the issuer and subject of certificate, respectively. And $X_{1,1}, ..., X_{1,n}$ are the hash values for m OCSP Responders.

$$C_{res} = Sig_{SK_{CA}}(PK_{res}, SN, I, S, V, X_{1,1}, ..., X_{1,n})$$

[Status validation of OCSP Responder's private key]

1. The CA delivers the hash value $X_{t,i}$ to OCSP Responder i, if OCSP Responder i's private key Sk_i is valid in period t.
2. When OCSP Responder i returns the response to the client in period t, she also delivers the hash value $X_{t,i}$ to the client.

$$< i, t, X_{t,i}, R, Sig_{SK_i}(R) >$$

3. When the client receives the response from OCSP Responder i, she verifies the digital signature using OCSP Responder's public key PK_{res}. Then the client can check the status validation of OCSP Responder's private key using the hash value $X_{t,i}$ received and $X_{1,i}$ contained OCSP Responder's certificate. In detail, the client checks the following equation. If the equation is satisfied, the client can certify that SK_i is valid.

$$X_{1,i} = H^{t-1}(X_{t,i})$$

In this way, the client can verify the status validation of the OCSP Responder's private key. The OCSP Responder's certificate is not revoked during its validity unless all of OCSP Responder's private keys are revoked[13].

2.2 Characteristics of D-OCSP-KIS

1. *Communication costs :* Each OCSP Responder has different private key but, have a same public key. Therefor, he clients need not to obtain the several OCSP Responders' certificate because the OCSP Responder's certificate is only one. Also, the clients need not to acquire the CRL about OCSP Responders. Thus, D-OCSP-KIS can reduce the communication costs.
2. *Storage amount :* The clients need not to obtain the several OCSP Responders' certificate and the CRL about OCSP Responders. Therefore, D-OCSP-KIS also can reduce the storage amount in clients.
3. *Computation costs :* The status validation of OCSP Responder's private key is performed by hashing operation instead of signature operation. The hash computation is much faster than digital signature computations. Therefore, the clients can reduce the computation costs for the status validation of OCSP Responder's private key.

2.3 Analysis of D-OCSP-KIS

Distributing of wrong hash values. Typically, a certificate may be revoked before expiration time because of the loss or compromise of the associated private key, in response to a change in the owner's access rights, a change in the relationship with the issuer or as a precaution against cryptanalysis[5]. In most cases, the user(or entity) can make a request to the CA for revoking own certificate and the CA publishes CRL including list of revoked certificates. In the reasons of certificate revocation, the compromise of private key may be done accidentally

Table 1. Additional load in CA at each time interval

Time interval		1 day	1 hour	1 minute	15 seconds	1 second
To 1 OCSP Responder	Computation costs of hash chain	365 hashings	8,760 hashings	525,600 hashings	2,102,400 hashings	31,536,000 hashings
	Storing X-chain	7.3 K bytes	175.2 K bytes	10.3 M bytes	41 M bytes	616 M bytes
	Distributing times of hash values	365 times	8,760 times	525,600 times	2,102,400 times	31,536,000 times
To 1000 OCSP Responders	Computation costs of hash chain	365 hashings	8,760 hashings	525,600 hashings	2,102,400 hashings	31,536,000 hashings
	Storing X-chain	7.3 K bytes	175.2 K bytes	10.3 M bytes	41 M bytes	616 M bytes
	Distributing times of hash values	365,000 times	8,760,000 times	525,600,000 times	2,102,400,000 times	31,536,000,000 times

or secretly by the attacker. In this case, the user cannot make a request to the CA for revoking own certificate. Suppose that an OCSP Responder's session private key is compromised by an attacker accidentally and securely in a time period (e.g., one day). The OCSP Responder cannot request the revocation to CA. And the CA may distribute to the OCSP Responder the wrong hash value that validates the certificate status in spite of the compromise of session private key. After all, the server and the user on E-commerce can have a serious confusion and damage.

Additional load in CA. The CA computes and stores X-chain at each time interval such as Table 1[14]. And the CA distributes the hash values to each OCSP Responder at the beginning of each period. In case that the number of OCSP Responder is 1000 and the time accuracy is 1 day, the CA must distribute the hash value to OCSP Responders by the number of 365000 times in total. In case that the time accuracy is 1 minute, the CA must distribute the hash value by the number of 8760000 times in total. Because the CA has a basic mission (such as certificate issue and revoke, CRL publishing, etc), the generating, storing and distributing (most critical) of hash values are additional loads to CA.

No detection of exposure of OCSP Responder's session private key. Suppose that an attacker steals OCSP Responder R_i's session private key secretly in period t. In this case, she can acquire the hash value $X_{t,i}$ easily and cannot derive any other OCSP Responder's private keys because she cannot obtain SK^*. And she cannot derive the hash value $X_{t+1,i}(H(X_{t+1,i}) = X_{t,i})$ because H is a one-way function. Therefore, an attacker cannot cheat the clients after period t[13]. However, If OCSP Responder R_i cannot recognize the fact that the its session private key is stolen in period t, she can disguise as the OCSP Responder until the period t is finished. She can offer the wrong OCSP response to the clients and the server and user on E-commerce can have a serious confusion and damage[14].

3 A Method for Detecting the Exposure of OCSP Responder's Session Private Key

As we mentioned earlier, D-OCSP-KIS can distribute the wrong hash value to OCSP Responder, give an additional load to CA, and cannot detect the exposure of OCSP Responder's session private key. Therefore, in this section, we propose a method for detecting the exposure of OCSP Responder's session private key in D-OCSP-KIS.

3.1 Proposed method

[Requirements]

1. Let n be the total number of OCSP Responders and m be the total number of clients. In general, n is much less than $m(n << m)$.
2. Suppose that the end user gets the CSI service through the client.

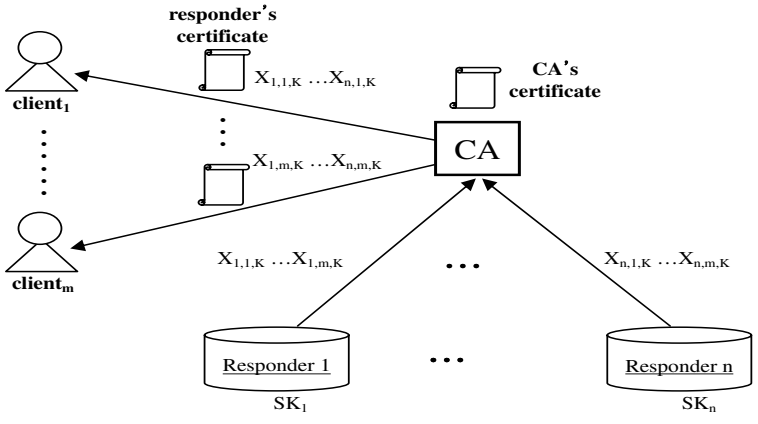

Fig. 2. Computation of hash value and issuance of OCSP Responder's certificate

3. Suppose that the client gets the CSI service from the OCSP Responder after the registration to CA.

[Computation of hash values for each OCSP Responder]

1. Let K be the total number of signature usage in an OCSP Responder. For an example, K is 10,000 if each OCSP Responder's certificate is expired after 10,000-signing operations for response. Thus, the certificate of the OCSP Responder is expired after 10,000-signature operations. The OCSP Responder can produce the hash value X_K using H as follows.

$$X_0 \stackrel{h}{\rightarrow} X_1 \stackrel{h}{\rightarrow} X_2 \stackrel{h}{\rightarrow} \ldots X_k \ldots \stackrel{h}{\rightarrow} X_K$$

2. The OCSP Responder repeatedly produces m hash-chain as different input values $X_{j,0}$ for m clients. $X_{j,k}$ denotes the hash value of time k for validation of OCSP Responder j.

$$\begin{array}{c} X_{1,0} \stackrel{h}{\rightarrow} X_{1,1} \stackrel{h}{\rightarrow} X_{1,2} \stackrel{h}{\rightarrow} \ldots X_{1,k} \ldots \stackrel{h}{\rightarrow} X_{1,K} \\ \vdots \\ X_{j,k} \\ \vdots \\ X_{m,0} \stackrel{h}{\rightarrow} X_{m,1} \stackrel{h}{\rightarrow} X_{m,2} \stackrel{h}{\rightarrow} \ldots X_{m,k} \ldots \stackrel{h}{\rightarrow} X_{m,K} \end{array}$$

3. Each OCSP Responder repeatedly produces $n \times m$ hash-chain as different input values $X_{i,j,0}$. $X_{i,j,k}$ denotes the hash value of time k computed in

OCSP Responder i for distribution to client j.

$$X_{1,1,0} \xrightarrow{h} X_{1,1,1} \xrightarrow{h} X_{1,1,2} \xrightarrow{h} \dots X_{1,1,k} \dots \xrightarrow{h} X_{1,1,K} \quad In\ OCSP\ Responder\ 1$$

$$\vdots$$

$$X_{1,m,0} \xrightarrow{h} X_{1,m,1} \xrightarrow{h} X_{1,m,2} \xrightarrow{h} \dots X_{1,m,k} \dots \xrightarrow{h} X_{1,m,K}$$

$$\vdots$$

$$X_{i,j,k} \quad In\ OCSP\ Responder\ i$$

$$\vdots$$

$$X_{n,1,0} \xrightarrow{h} X_{n,1,1} \xrightarrow{h} X_{n,1,2} \xrightarrow{h} \dots X_{n,1,k} \dots \xrightarrow{h} X_{n,1,K}$$

$$\vdots$$

$$X_{n,m,0} \xrightarrow{h} X_{n,m,1} \xrightarrow{h} X_{n,m,2} \xrightarrow{h} \dots X_{n,m,k} \dots \xrightarrow{h} X_{n,m,K} \quad In\ OCSP\ Responder\ n$$

4. Each OCSP Responder stores the input values of $X_{i,1,0}, ..., X_{i,m,0}$ and all intermediate hash values and sends all the final hash values of $X_{i,1,K}, ..., X_{i,m,K}$ to CA, securely.

[Issuance of OCSP Responder's certificate in CA]

CA gathers $X_{i,1,K}, ..., X_{i,m,K}$ from each OCSP Responder and issues m OCSP Responder's certificates C_{client_j} for distribution to the clients by using its own private key. SN is the serial number of certificate and V represents the validity period. I and S denote the issuer and subject of certificate, respectively. Then, the hash values included in each certificate are different from each other. The contents of certificate distributed to m clients are as follows.

$$C_{client_1} = Sig_{SK_{CA}}(PK_{res},\ SN\ , I, S, V, X_{1,1,K}, ..., X_{n,1,K}) \quad Certificate\ for\ client\ 1$$

$$\vdots$$

$$C_{client_m} = Sig_{SK_{CA}}(PK_{res},\ SN\ , I, S, V, X_{1,m,K}, ..., X_{n,m,K}) \quad Certificate\ for\ client\ m$$

[Status validation of OCSP Responder's private key in client]

1. When OCSP Responder i returns the response to the client j, she also delivers the hash value $X_{i,j,k}$ to the client. She delivers $X_{i,j,K-1}$ to the client at the first response and delivers $X_{i,j,K-2}$ to the client at the next response. Thus, she delivers $X_{i,j,0}$ at the last response.

$$< X_{i,j,K-1},\ R\ , Sig_{SK_i}(R) > \quad At\ the\ first\ response\ to\ client\ j$$

$$\vdots$$

$$< X_{i,j,k},\ R\ , Sig_{SK_i}(R) > \quad At\ the\ k-th\ response\ to\ client\ j$$

$$\vdots$$

$$< X_{i,j,0},\ R\ , Sig_{SK_i}(R) > \quad At\ the\ last\ response\ to\ client\ j$$

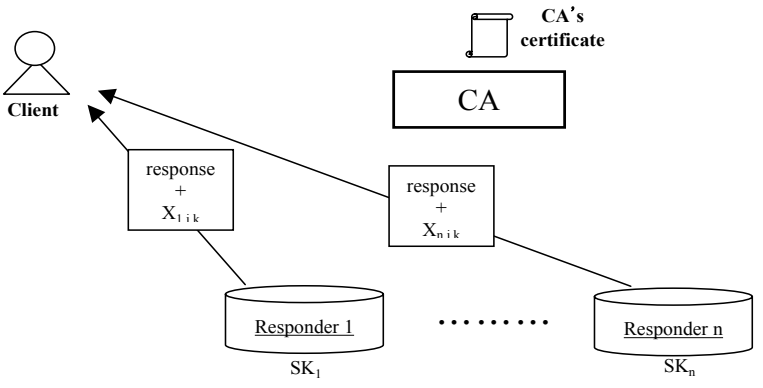

Fig. 3. Status validation of OCSP Responder's private key

2. When the client j receives the response from OCSP Responder i, she verifies the digital signature using OCSP Responder's public key PK_{res}. Then the client can check the status validation of OCSP Responder's private key using the hash value $X_{i,j,k}$ received in response and $X_{i,j,K}$ contained in OCSP Responder's certificate. In detail, the client can certify that SK_i is valid by performing 1-hash function operation at the first response and by performing 2-hash function operations at the second response. And the client can certify by performing K-hash function operations at the last response. Thus, the client computes $K/2$(5,000 in this case)-hash function operations on the average for the status validation of OCSP Responder's private key.

$$X_{i,j,K} = H(X_{i,j,K-1}) \qquad At\ the\ first\ response$$
$$\vdots$$
$$X_{i,j,K} = H^k(X_{i,j,K-k}) \qquad At\ the\ k-th\ response$$
$$\vdots$$
$$X_{i,j,K} = H^K(X_{i,j,0}) \qquad At\ the\ last\ response$$

3. Then, the client stores the *Count* k at each response and compares it with the *Count* of previous response. If the present *Counter* is larger than the previous Counter by 1 in value($C_{now} = C_{before} + 1$), the client can recognize that the response is valid. Otherwise, the client can recognize the exposure of session private key and the abuse of hash value.

[Detection procedure of OCSP Responder's session key's exposure in client]

1. The client performs 1-hashing operation using the hash value $X_{i,j,k}$ included in response, sets X_{temp} to the hash value, and increments the counter C_{now} by 1.

$$X_{temp} \leftarrow H(X_{i,j,k})$$

$$C_{now} \leftarrow C_{now} + 1$$

2. The client then compares X_{temp} with $X_{i,j,K}$ contained in OCSP Responder i's certificate.

$$X_{temp} \stackrel{?}{=} H(X_{i,j,K})$$

If this holds, goto step 3. Otherwise, the client sets $X_{i,j,k}$ to X_{temp} and goto step 1.

$$X_{i,j,k} \leftarrow X_{temp}$$

3. The counter C_{now} is compared with C_{before}. If following condition $C_{now} \stackrel{?}{=} C_{before} + 1$ is satisfied, then the client accepts the response and goto step 4. Otherwise, he rejects the response because of recognizing the exposure of session private key and the abuse of hash value.

4. After setting C_{before} to C_{now} and C_{now} to 0,

$$C_{before} \leftarrow C_{now}$$
$$C_{now} \leftarrow 0,$$

the client proceeds to perform step 1.

4 Characteristics and Comparisons

In this section, we explain the characteristics of proposed method and compare of traditional D-OCSP, D-OCSP-KIS and proposed method. The detailed characteristics are as follows:

1. *Detection for the exposure of session private key and hash value*
In D-OCSP-KIS, anyone can easily acquire the hash value transferred from the OCSP Responders to the clients and from CA to OCSP Responders. Therefore, an attacker acquiring accidentally an OCSP Responder's session private key can disguise as the OCSP Responder and can cheat the clients. And she can offer wrong response to clients during the time period unless the OCSP Responder recognize it. In the proposed method, however, the clients can immediately detect the exposure of OCSP Responder's session private key and the abuse of hash value.
2. *Usage times of OCSP Responder's private key*
In the proposed method, the private key of OCSP Responder is used in limited times because the validity proof of private key is one time hash value. At above, it was supposed that K usage time is 10,000. In this case, the client computes 1-hash function at the first response and the 10,000-hash functions at 10,000-th response for validation of OCSP Responder's private key. Thus, the client computes average 5,000-hashing functions such as in Table 2. K can be set bigger(e.g., 20000, 50000, 100000) or smaller(e.g., 8000,

5000, 2000) than the average value. Setting K more than 10,000 is inefficient because the computation time for the status validation is larger than digital signature operation. Of course, the number of hashing operations in client can be reduced by the method adding 3-hash values to OCSP Responder's certificate. Suppose that these hash values are computed by 5,000-hashing function operations from different initial values. Total usage time K is 15,000 but, each client only computes an average of 2,500-hashing function operations for the status validation. When the OCSP Responder spends all of the usage time, it computes a new hash-chain and transfers the only hash value to the client through CA unless its private key is compromised or the public key and other information is changed.

3. *Decreasing of CA′s Load*
In D-OCSP-KIS, CA should compute and store all of the hash chain such as in Table 1. Also, CA distributes the hash values to each OCSP Responder at the beginning of each period periodically. This job requires additional passes between CA and each OCSP Responder and can give the extra load to CA. However, in proposed method, each OCSP Responder computes the hash chain. And CA collects them and issues an certificate of OCSP Responder. Also, additional passes for distribution of hash value are not required. Thus, our method does not give the additional load to CA.

Table 2. Comparisons of the proposed method and other methods

	Traditional D-OCSP	D-OCSP-KIS	Proposal
Structure of Res's Cert.	maintain	modify (+20n byte)	modify (+20n byte)
The number of Res's Cert. acquired in client	n	1	1
The number of signning for issuring Res's Cert. in CA	n	1	m
Structure of response	maintain	modify (+ 60 byte)	modify (+ 40 byte)
Addition of passes	-	n x T (at beginning of period)	m (at initial)
Computation costs of Res's certificate status in client	online or offline	t-hash computation (average:365/2)	k-hash computation (average:10,000/2)
Usage period of certificate	365 days	365 days	10,000 times (more or less is possible)
Detection of Res's private key exposure	X	X	O

Notes. n : the number of OCSP Responder, m : the number of client, T : total number of time-periods, K : the number of sign usage, × : Not supported, ○ : Supported

5 Conclusion

In this paper, we proposed a method that can immediately detect the exposure of OCSP Responder's session private key and the abuse of hash value in D-OCSP-

KIS. In our proposal, the hash values are only used one time and and the load for computation of X-chain in CA is distributed to each OCSP Responder. The method can decrease the additional load to CA. Our future work is to increase the usage time of OCSP Responder's private key and to decrease the number of hash function operation for the status validation.

References

1. A. Malpani, R. Housley, T. Freeman.:Simple Certificate Validation Protocol(SCVP), IETF Internet Draft, June, 2002.
2. C. Adams, P. Sylvestor, M. Zolotarev, R. Zuccherato.: Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols. IETF RFC 3029, February, 2001.
3. Dae Hyun Yum, Pil Joong Lee.:A Distributed Online Certificate Status Protocol Based on GQ Signature Scheme, ICCSA 2004, LNCS 3043, pp.471-480, 2004.
4. ITU/ISO Recommendation.:X.509 Information Technology Open Systems Interconnection-The Directory:Authentication Frameworks, 2000.
5. Jose L. Munoz, Jordi Forne, Oscar Esparza, and Miguel Soriano.:A Certificate Status Checking Protocol for the Authenticated Dictionary, MMM-ACNS 2003, LNCS 2776, pp.255-266, 2003.
6. Leo Reyzin.:General Time/Storage Tradeoffs for Hash-Chain Re-comoutation, unpublished manuscript.
7. M. Myers, R. Ankney, A. Mappani, S. Galperin, C. Adams.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, IETF RFC 2560, June, 1999.
8. NIST FIPS (Federal Information Processing Standards Publication) 186-1.: Digital Signature Standard, December, 1998.
9. P.C.Kocher.:On Certificate Revocation and Validation, Financial Cryptography (FC'98), LNCS 1465, pp.172-177, Springer-Verlag, 1998.
10. Paul. Kocher.:A Quick Introduction to Certificate Revocation Tree(CRTs), Technical Report, Valicert, 1999.
11. R. Housley, W. Ford, W. Polk, D. Solo.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, IETF RFC 2458, January, 1999.
12. R. Housley, W. Ford, W. Polk and D. Solo.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile, IETF RFC 3280, April, 2002.
13. Satoshi Koga, Kouichi Sakurai.:A Distributed Online Certificate Status Protocol with a Single Public Key, Public Key Cryptography 2004, LNCS 2947, pp.389-401, 2004.
14. Silvio Micali.:NOVOMODO ; Scable Certificate Validation And Simplified PKI Management, 1st Annual PKI Research Workshop Preproceedings, pp.15-25, 2002.
15. http://www.eskimo.com/∼weidai/benchmarks.html.

Installing Fake Root Keys in a PC

Adil Alsaid and Chris J. Mitchell

Information Security Group,
Royal Holloway, University of London
Egham, Surrey TW20 0EX
{A.Alsaid, C.Mitchell}@rhul.ac.uk

Abstract. If a malicious party can insert a self-issued CA public key into the list of root public keys stored in a PC, then this party could potentially do considerable harm to that PC. In this paper, we present a way to achieve such an attack for the Internet Explorer web browser root key store, which avoids attracting the user's attention. A realisation of this attack is also described. Finally, countermeasures that can be deployed to prevent such an attack are outlined.

1 Introduction

As is widely known [10], most web browsers (e.g. Microsoft Internet Explorer or Netscape) have a repository of root public keys designed for use in verifying digitally signed public key certificates. These public keys are bundled with distributions of the web browser, and are used to verify certificates for applet providers [13]. Specifically, web-sites may download applets to a user PC without the PC user knowing it. Depending on the security settings selected by the PC user, these applets may be executed with or without further checks. Typically, the browser will only execute the applet if the following conditions are satisfied.

1. The applet must be digitally signed, and the signature must verify correctly.
2. The public key required to verify the signature on the applet must be contained in a (valid) public key certificate signed using a private key corresponding to one of the stored root public keys. That is, the certificate must be verifiable using a stored root key.
3. The PC owner answers 'yes' to a question along the following lines: 'Are you prepared to trust software signed by X', where X is the name in the certificate verified in the previous step.

Suppose a malicious entity generates two key pairs. One key pair is designated the CA key pair, and the other key pair is designated the software supplier key pair. The private key from the CA key pair is used to sign a certificate for the public key from the software supplier key pair, and the name of a reputable software supplier is included in this certificate. Now, if the malicious party could insert his CA public key into the list of root public keys stored in a PC, then this party could successfully sign applets (using the software supplier private

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 227–239, 2005.

key) which will appear to a user of the PC as if they come from the reputable software supplier.

This is clearly a possible route for an attack on a PC. However, there are two obvious questions which must be answered before it is worth considering this further.

1. If an attacker is able to insert false public keys into the PC repository, why not simply insert a rogue application directly? There are two possible answers to this question. Firstly, the insertion of a false public key allows arbitrary numbers of rogue applications to be executed on a PC, at any time in the future. This means that installing a rogue root CA public key is an attack that "cascades". Secondly, a false public key is undetectable by current attack detection software, whereas a malicious application will often be detected by such software. The reason that rogue public keys are not detected by virus scanners is that there is no simple way of distinguishing between public keys which should be in the list, e.g. because they were supplied by the browser or because they have deliberately been added by the user, and those which should not.
2. If an attacker is able to insert false public keys into the PC repository, why not simply corrupt the web browser to remove the checking of downloaded applets? The answer to this is straightforward; it may be a lot simpler to insert a single false public key into a PC repository than to come up with a patch to Internet Explorer that stops the checking of applets. The latter would presumably require a sophisticated understanding of the Internet Explorer executable.

The rest of the paper is organized as follows. Section 3 discusses at a high level possible means by which a root public key can be installed into a PC. Section 4 describes in detail one practical method for installing a root public key without user intervention, which has been successfully implemented. Section 5 analyses possible countermeasures that can be deployed to prevent such an attack.

2 Related Work

The authors are not aware of any other work that addresses this exact problem. However, Levi pointed out this problem and the dangers posed by root public keys [10]. He proposed that root certificate installation should be avoided, and that access to the root certificate store should be controlled. Moreover, he recommended that users should check certificate details to make sure that every certificate is valid and genuine.

Hayes [8] discusses a practical solution enabling a CA to provide a secure in-band update of a CA X.509 v3 certificate in a user's personal security environment. In a further paper [7], Hayes discusses the vulnerability of multiple roots in web browsers and the dangers of certificate masquerading. The need for improved methods for verifying the binding of a root CA to the source of protocol messages is described.

3 Installing Root Certificates

Installing a root certificate is a straightforward process. In this paper we will limit the discussion to the Microsoft Windows 2000 operating system and the Microsoft Internet Explorer web browser [14]; other operating systems and web browsers have similar means for installing root certificates. This discussion provides the necessary background for the attack described in section 4.

Before proceeding, observe that a root public key is always stored by Internet Explorer in a special format known as a 'self-signed certificate'. This means that the public key is actually stored in an X.509 certificate, where the certificate is signed using the private key corresponding to the public key inside the certificate. Whilst such a certificate does not function like a normal certificate, i.e. it does not guarantee the binding between subject name and public key, it does guarantee that the subject of the certificate knows the private key corresponding to the public key (so called 'proof of possession', [11]). This is because the creator of the certificate must have had the private key in order to sign the certificate. In order to trust the content of the self-signed certificate, i.e. to believe the binding between the name and public key that is inherent in the certificate, one needs *a priori* to trust the owner of the public key used to verify the self-signed certificate. As a result these root public keys are typically (rather confusingly) referred to as 'root certificates' or 'X.509 root certificates' and we follow this convention in the remainder of this paper.

In the remainder of this section we therefore first consider how a root public key can be put into the X.509 root certificate format. We follow this by describing the conventional method for adding such a root certificate to the list stored by Windows. This is then followed by a general discussion of means by which this might be achieved without the PC user's knowledge or consent.

3.1 Creating a Root Certificate

Creating an X.509 root certificate [13] can be achieved using any of the freely available certificate creation tools. One such tool is makecert.exe [2] as supplied by Microsoft. Using makecert.exe, the following command will issue a self-signed root certificate and save it to a certificate file 'root.cer'. It creates a public and private key pair for digital signatures. It stores the private key in the file that was passed as part of the command line, 'root.pvk' in the case of the given example. If the file does not exist, the command creates it to store the private part of the key. Two command line arguments are of particular significance here, namely the -r and the -n options. The -r option is used to issue a self-signed root certificate and the -n option is used to specify the subject certificate name in a way that conforms to the X.509 standard.

```
makecert -r -n "CN=MyRootCA,OU=MyOrganization,O=CompanyName,
E=Emailaddress" -sv root.pvk root.cer
```

We next explore various ways in which a root certificate, e.g. created using makecert.exe, can be added to the list used by Internet Explorer.

3.2 Installing a Root Certificate Under User Control

Once a root public key has been created and inserted into a self-signed (root) certificate, double clicking on the root certificate file launches the certificate management program (the Microsoft Certificate Import Wizard) to view and install certificates. The certificate management program then displays a set of dialog boxes to allow the user to manage the root certificate installation process. In a typical scenario, a user will keep clicking 'OK' and accept the default settings for each of the dialog boxes [6].

We next consider what processes are being executed by Windows when these dialog boxes are shown. This will provide the basis for an understanding of how adding a root certificate might be achieved without user consent.

1. The user double clicks on the certificate file. Microsoft Windows then launches the certificate management program to open the certificate, (see Fig. 1).
   Installing the certificate can be initiated by clicking on the "Install Certificate" button, which displays a dialog box requesting the user to select a store in which to place the new certificate, see Fig. 2.
2. If the user accepts the default settings, the wizard will select the certificate store based on the type of the certificate. In the case of a root certificate, the certificate will be stored in the certificate authority (CA) store, which is located in the Windows Registry.

Fig. 1. 'Installing a new certificate' dialog box

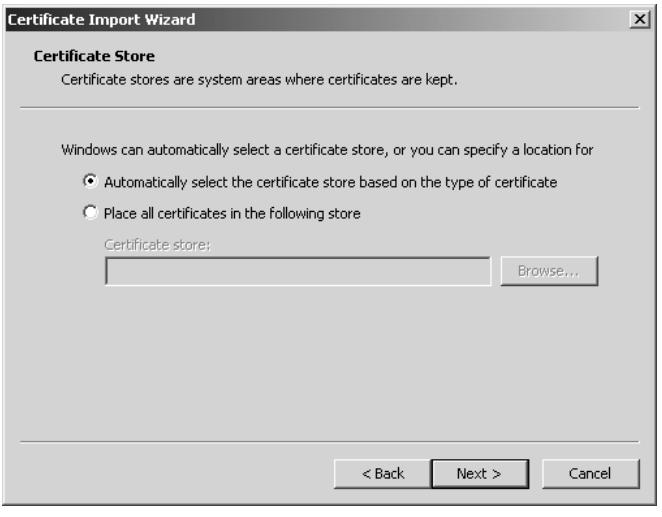

Fig. 2. 'Selecting the certificate store' dialog box

3. When the next button is clicked, and if the certificate type is a root certificate, a message box will be displayed warning the user and waiting for user input to complete the task. This box will ask the user for confirmation that the user wishes to add the new certificate to the root store, see Fig. 3. The message box shows the issuer name and thumbprint for the certificate, i.e. a hash-code computed as a function of the certificate. The thumbprint is shown in the message box to help the user confirm the origin of the certificate. For example, the user could obtain the correct thumbprint from the certificate issuer, and compare this with the thumbprint displayed in the message box. Normal Users, i.e. users without administrative privileges, can still install root certificates.

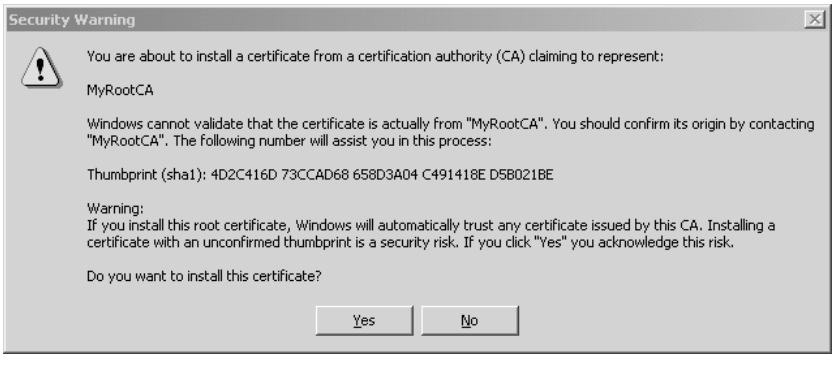

Fig. 3. 'Adding a root certificate' message box

3.3 Malicious Installation of a Root Certificate

A malicious third party could install a root certificate by running a special applet that inserts a self-issued root certificate into the browser's list of root CAs. However, if the malicious applet uses the certificate import wizard to achieve this, the certificate import wizard will display a message box to alert the user to the fact that a third party is trying to install a root certificate on their machine, as described in Section 3.2. The challenge is to 'silently' install the root certificate without user intervention. In the next subsection, general approaches to silent root certificate installation are discussed.

3.4 General Approach to Silent Root Certificate Installation

In order to silently install a root certificate, a malicious third party must first be able to convince the user to run a special applet that will install the root certificate. This could be achieved in a variety of ways, e.g. by a virus, a trojan horse, or simply a Java or Visual Basic script. The malicious code could use more than one approach to silently install a root certificate into a PC. We next describe two ways that the malicious code might achieve such a task.

1. **Using standard tools**
   This approach uses the standard tools, e.g. the Microsoft certificate import wizard, to install the certificate, but somehow manages to hide the 'security warning' message box. As above, a malicious third party must first convince the user to run a program that will insert the root certificate into the PC. The program can use features of the Windows operating system Graphical User Interface (GUI) to hide the 'security warning' message box and simultaneously simulate user acceptance that a new root certificate should be added to the store. This approach will be discussed in more detail in Section 4.
2. **Writing directly to the root certificate store**
   In this approach, the malicious program writes the false root certificate directly to the certificate store, i.e. the Registry in the case of Internet Explorer, without using any of the provided tools. The Registry [9] is the data repository in the Microsoft Windows environment in which most of the Windows settings and program information are kept. The Registry has a hierarchical structure analogous to the directory structure in a file system. However, instead of using folders and subfolders, it uses keys and subkeys. When a root certificate is installed, certain changes are being made to the Registry, as shown in Fig. 4. First, a subkey is created for the new certificate in the root certificates store underneath the 'Certificates' key. The value of the subkey is the Thumbprint of the newly added certificate, i.e. the subkey that starts with '4D2C41...'. Second, an entry is created under the '4D2C41...' subkey to store the certificate details, i.e. 'Blob' in the case of the example shown in Fig. 4. Finally, the subkey 'ProtectedRoots' is created underneath the 'Certificates' key, which is a binary value that needs special access control privileges to change or manipulate.

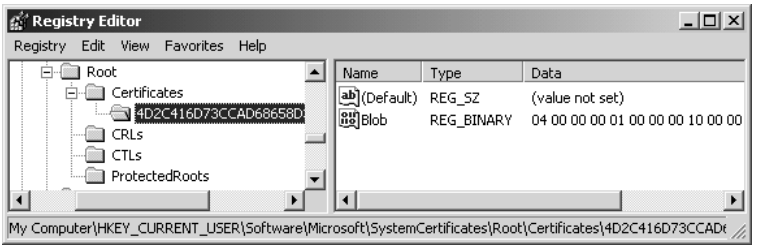

Fig. 4. Changes made to the Registry when installing a new root certificate

The authors were able to write a small program to write directly to the registry and to produce most of the keys. However, the authors were not able to reproduce the value stored in the 'ProtectedRoots' subkey. Moreover, there is access control protection on the 'ProtectedRoots' that requires a special privileged user, i.e. SYSTEM, to change the value of the key. The details of how to correctly make such modifications to the Registry is far from obvious and, as a result, it has not so far been possible to successfully implement such an attack.

4 A Practical Method for Silently Installing a Root Certificate

In this section, a practical method for silent installation of a root certificate is introduced. This method is an implementation of the first approach outlined in Section 3.4. The method relies on the Microsoft Windows Cryptographic Application Programming Interface (CryptoAPI) [3] to install a root certificate. It uses the CAPICOM, which is the Microsoft Cryptographic API with COM [1] support. It also uses features of the Microsoft Windows message system [4] to hide the 'security warning' message box. The following paragraphs describe the solution in more detail.

First, as previously, we suppose that a user executes a malicious third party program that will install the fake root certificate. In order for the malicious third party program to achieve such a task it performs the following steps.

1. The program must have access to a copy of the false root certificate. The fake root certificate can be hard coded in the program or stored in an external file or link. Makecert.exe or any other certificate creation tool could be used to create the fake root certificate, as described in Section 3.1.
2. When the program starts, it creates another running thread that monitors all windowing activities in the user's environment; we call this thread the 'monitoring thread'. The main task of the monitoring thread is to monitor all windows activities on the system until it detects the 'security warning' message box, get a 'handle' to it, and then take actions to both hide the box and provide a fake user confirmation (as described below). A more reliable

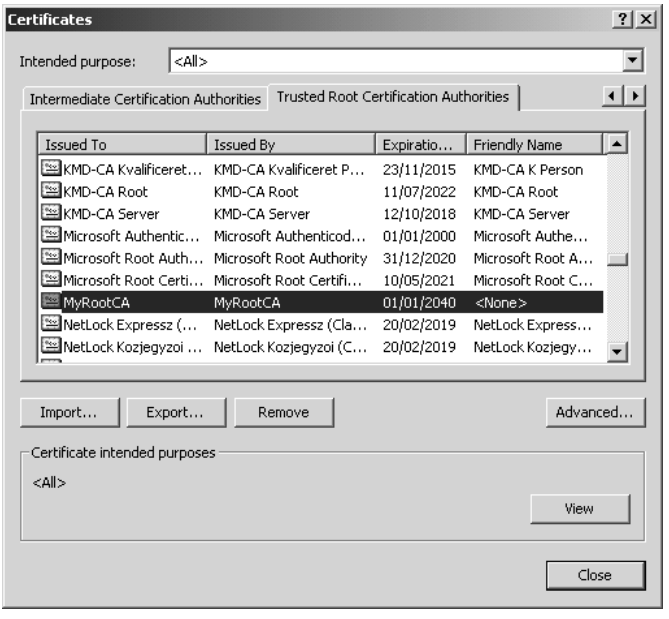

Fig. 5. 'List of root certificates' dialog box

way to detect the 'security warning' message box creation event is to use Windows Hooks [5], a mechanism to intercept system events. Using Windows Hooks, obtaining the handle of the 'security warning' message box can be achieved by intercepting the window creation system message that is sent to the application when creating the 'security warning' message box.

3. After creating the monitoring thread, the program makes a CryptoAPI call to add the fake root certificate to the list of root certificates in the system. When the program executes the call to the CryptoAPI to add the new root certificate, the CryptoAPI displays a security warning message box and waits for the user to confirm the addition of the root certificate. At this moment, the monitoring thread detects the security warning message box and obtains a handle to it.
4. The monitoring thread now takes steps to immediately provide a positive user response to the message box. This is achieved by the program sending a WM_CHAR message to the message box window handle. This message contains 'Y', i.e. it simulates the effect of the user pressing 'Y' on the keyboard as a positive response to the request made by the message box. The message box will immediately disappear, and the user will probably not detect anything untoward as the box will disappear almost as soon as it appears.
5. Now, as shown in Fig. 5, the root certificate will have been added to the list of root certificates in the user's PC.

This approach to implementing a 'silent' root key installation attack would also work for other web browsers, and/or for browsers running on other plat-

forms. For example, we believe that a similar approach could be used to silently install a fake root public key in the root key store for the Netscape/Mozilla browser running on a Linux platform. However, the exact method of implementing such an attack is dependent on the version of the Netscape/Mozilla browser being used, as well as the graphical user interface installed on the user machine.

Code implementing the attack described above is provided in Appendix A. The code successfully performs the addition of a root certificate without user intervention or user knowledge.

5 Countermeasures

We conclude this paper by suggesting some countermeasures to the threat of installation of a fake root certificate in a user PC. As with any security issue, there are two fundamental approaches to such a problem: (pro-active) prevention and (reactive) after the-event detection. We first mention two possible preventative measures.

1. When carrying out such a security sensitive task, users should always be re-authenticated. This will eliminate the problem of a malicious third party adding a root certificate without user intervention.
2. The attack could also be prevented by restricting access to the list of root public keys to special privileged users or processes.

Whilst prevention is the ideal solution, this can only be achieved in the long-term, since it requires modifications to the Windows environment. To address the problem in the immediate future requires reactive measures which detect when a false root certificate has been added (and take steps to remove it). One approach to the problem involves producing a small tool that scans the list of root certificates for malicious third party certificates. Such a utility would need to have access to the list of 'good' root certificates. One approach would be for the utility to store the list of root certificates that comes with the browser on its first installation. The user can then run this scanning utility routinely to check for the presence of malicious third party root certificates.

A second approach is to use the Online Certificate Status Protocol (OCSP) [12] to verify the status of a certificate before using it, and only allow 'current' certificates to be used. However, a motivated attacker might set up a rogue OCSP server to engage in such a protocol and fake the status of the certificate.

A further approach is for the browser to maintain two lists of root keys. One list is for the genuine root keys that were verified by the publisher of the browser, i.e. shipped with the browser. A second list will contain root public keys that were added by the user and that were not shipped with the browser. In this scenario, when engaging in transactions that use one of the root public keys in the second list, the browser will indicate the fact that the root public key being used is not from amongst those shipped with the browser, and hence is less reliable. As a consequence, the browser would give the user the option to stop the transaction.

Both the pro-active and reactive approaches to addressing this threat are the subject of ongoing research.

6 Conclusions

It is likely that most web browsers and operating systems are candidates for the attack discussed in this paper. Users should take special care when installing root certificates. Normal users should not be allowed to install new root certificates or make any changes to the root certificate store. Implementing such steps would eliminate most of the problems associated with a malicious third party installing a fake root certificate.

References

1. D. Box. *Essential COM*. Addison-Wesley, Boston, MA, 1998.
2. Microsoft Corporation. Certificate creation tool (makecert.exe), May 2004. http://msdn.microsoft.com/.
3. Microsoft Corporation. Cryptography, CryptoAPI, and CAPICOM, May 2004. http://msdn.microsoft.com/.
4. Microsoft Corporation. Messages and Message Queues, May 2004. http://msdn.microsoft.com/.
5. Dino Esposito. Windows Hooks in the .NET Framework. *MSDN Magazine*, 17(10), October 2002.
6. Peter Gutmann. A reliable, scalable general-purpose certificate store. In *16th Annual Computer Security Applications Conference, December 11-15, 2000, New Orleans, Louisiana*, pages 278–287. IEEE, 2000.
7. James M. Hayes. The problem with multiple roots in web browsers – certificate masquerading. In *IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 306–311. IEEE Computer Society, 1998.
8. James M. Hayes. Secure in-band update of trusted certificates. In *IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 168–173. IEEE Computer Society, June 1999.
9. Jerry Honeycutt. *Microsoft Windows XP Registry Guide*. Microsoft Press, Richmond, Washington, 2003.
10. Albert Levi. How secure is secure web browsing? *Communications of the ACM*, 46(7):152, July 2003.
11. C. J. Mitchell and R. Schaffelhofer. The personal PKI. In C. J. Mitchell, editor, *Security for Mobility*, chapter 3, pages 35–61. IEE, London, UK, 2004.
12. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol — OCSP. RFC 2560, June 1999.
13. Andrew Nash, William Duane, Celia Joseph, and Derek Brink. *PKI: Implementing and Managing E-Security*. Osborne/McGraw-Hill, Berkeley, California, 2001.
14. Scott Roberts. *Programming Microsoft Internet Explorer 5*. Microsoft Press, Redmond, Washington, 1999.

Appendix A: Code to Add a Root Certificate without User Intervention

```
#include <tchar.h> #include <atlbase.h> #include <windows.h>

#pragma warning (disable : 4192)

#import "capicom.dll" using namespace CAPICOM; HWND RootHWnd=0;

BOOL CALLBACK EnumChildProc(
    HWND hwnd,
    LPARAM lParam)
{
    char TitleBuf[255];
    GetWindowText(hwnd, TitleBuf, 255);

    // Get a handle to the Security Warning Message box
    if(!RootHWnd) {
        if((strcmp(TitleBuf,"Root Certificate Store")==0)
            // a new update changed the window's title to
            // 'Security Warning'
            || (strcmp(TitleBuf,"Security Warning")==0)){
            RootHWnd=hwnd;
            // stop enumeration
            return FALSE;
        }
    } else {
            // Already got the Security 'Warning Message Box'
            // handle, then get the handle of the Yes Button
            // and emulate user input by sending the yes message
        if(strcmp(TitleBuf,"&Yes")==0) {
            PostMessage(hwnd,WM_CHAR,'y',1);
            return FALSE;
        }
    }
    return TRUE;
}

DWORD WINAPI ThreadFunc( LPVOID lpParam ) {
    LONG lRet;
    lRet = EnumChildWindows(GetDesktopWindow(), EnumChildProc, 0);
    if(RootHWnd)
        lRet = EnumChildWindows(RootHWnd, EnumChildProc, 0);
    return 0;
}
```

```
int __cdecl _tmain (int argc, _TCHAR  * argv[]) {
    HRESULT hr = S_OK;

    CoInitialize(0);

    try
    {

        _bstr_t   bstrName = _T("Root");
        IStorePtr pIStore(__uuidof(Store));

        if (FAILED(hr = pIStore->Open(CAPICOM_CURRENT_USER_STORE,
                                bstrName,
                                CAPICOM_STORE_OPEN_READ_WRITE)))
        {

          ATLTRACE(
          _T("Error [%#x]: pIStore->Open() failed at line %d.\n")
                  , hr, __LINE__);
          throw hr;
        }
        CAPICOM::ICertificate2Ptr pICert2 = NULL;
        pICert2.CreateInstance("CAPICOM.Certificate");

        // load the fake CA to be installed from disk....
        if (hr = pICert2->Load("root.cer","",
                                CAPICOM_KEY_STORAGE_DEFAULT,
                                CAPICOM_CURRENT_USER_KEY) != 0)
            exit(1);
        else { // Load succeeded


            DWORD dwThreadId, dwThrdParam = 1;
            HANDLE hThread;

            // Create the Monitoring thread
            hThread = CreateThread(
                NULL,           // default security attributes
                0,              // use default stack size
                ThreadFunc,     // thread function
                &dwThrdParam,   // argument to thread function
                0,              // use default creation flags
                &dwThreadId);   // returns the thread identifier

           // Check the return value for success.
```

```
            if (hThread == NULL)
               MessageBox( NULL, "CreateThread failed.",
                             "main", MB_OK );

            else {
            // Thread is monitoring the windows activities...
            // Then, try to install the fake root CA
             hr=pIStore->Add(pICert2);
             CloseHandle( hThread );
            }
        }
    }
    catch (_com_error e)
    {
        hr = e.Error();
        ATLTRACE(_T("Error [%#x]: %s.\n"), hr,
            e.ErrorMessage());
    }

    catch (HRESULT hr)
    {
        ATLTRACE(_T("Error [%#x]: CAPICOM error.\n"), hr);
    }
    catch(...)
    {
        hr = CAPICOM_E_UNKNOWN;
        ATLTRACE(_T("Unknown error.\n"));
    }
    CoUninitialize();
    return (int) hr;
}
```

Provision of Long-Term Archiving Service for Digitally Signed Documents Using an Archive Interaction Protocol

Aleksej Jerman Blazic[1] and Peter Sylvester[2]

[1] SETCCE, Jamova 39, SI-1000 Ljubljana, Slovenia
aljosa@setcce.org
[2] Edelweb, 15, quai de Dion-Buton, 92816 Puteaux cedex, France
peter.sylvester@edelweb.fr

Abstract. This paper presents the basic design considerations and implementation of a trusted long-term preservation service for electronic records based on a protocol for archival service interaction. The main focus of the research is on e-government and e-business enabled applications and the preservation of electronic heritage in complex and open system environments including PKI enabled infrastructures for digitally signing documents. The operational solution must ensure an easy to use and reliable digital objects archiving service. The archival service is being developed as an open, scalable, modular and extensible service using Internet technologies. It uses international standards under development by the IETF and ETSI in the field of digital signatures and trusted long term archiving services and has provided feedback to the standardization activities.

1 Introduction

Digital objects (e.g. electronic records) face a continuous and threatening technology change, often affecting and replacing the environment where objects had been initially created. Digital objects require constant and perpetual maintenance since they depend on elaborate systems of hardware, software, data and information models, and standards that are upgraded or replaced every few years. Preservation of such electronic heritage is a multi-dimensional challenge with the single aim of providing a stable environment for the continuous usability of content value. Furthermore, the ability to rely on digital records is an issue of increasing concern with the proliferation of paperless environments in non B2B scenarios (i.e. citizen/customer to business/government). Taking into account critical system environments (e.g. e-business or e-government enabling environments), preservation techniques must answer the main questions of accountability, consistency and existence in a timeline of formally used electronic documents.

Preserving electronic records is challenged by several conceptual and technical problems. In the past, archival and recordkeeping approaches have been focused mainly on the functions, processes, and uses associated with the records, rather than on physical object control. Most of these researches have addressed the technical strategies to preserve the readability of archived objects (emulation, migration and encapsulation). These preservation strategies are important as they help overcome the

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 240 – 254, 2005.

hardware and software obsolescence, without necessarily retaining the functionality of a record. Other strategies deal with archival object management (submission, management, extraction and deletion), initially addressed by NASA in the Open Archival Information System (OAIS) [1] (designed after the complete loss of Voyager mission data) now recognized as an OSI standard.

However, while managerial and readability perspectives of preservation are a multi-domain concern (e.g. document management systems), an important aspect of electronic records preservation is the capability to demonstrate existence and integrity (e.g. against a third party) or in other words, an electronic preservation service might be required sometime in the future to prove that an object (and its related elements), which is not a subject of current operation or (e.g. business/government) process, existed at a certain time in the past and has not been modified since that time.

Furthermore, archived electronic records may contain digital signatures or time stamps to prove the origin and the creation time of these objects and signatures. In the course of time the value of evidence of these signatures or timestamps can decrease or can get lost for many reasons, like non-existing revocation information on issued digital certificates guaranteeing the authenticity of the signatory (e.g. due to decommissioning of the certificate authority), expiration or revocation of a certificate (due to key loss or damage associated with a digital signature or weakness of the cryptographic algorithms). In particular, the validity of the digital signature can no longer be based on a validity check of a certificate after the expiration of the certificate (usually after five years). We note in particular, that using a timestamp only shifts the problem of signature validation of a document to the problem of validation of a time stamp leaving us in a situation that a time stamp cannot be validated in the long term contrary what its name may suggest, at least not by using its signature.

The situation described in the previous paragraph is largely influenced by common law culture, or, in other words, in recent years, legal activities concerning electronic documents are more concerned with the aspects of creation and digital signatures rather than conservation and archiving. But, in many legal systems, e.g. those inherited from the Austro-Hungarian realm, most documents need explicit descriptions concerning their conservation and destruction. One could say that, without treating conservation issues, digital data cannot even have the status of a document. Another problem is the perpetual confusion of legal and technical terms, in particular when different legal systems are involved. This results in totally different interpretations of the same terminology and functional decompositions (e.g. ETSI's archive time stamp versus the IETF's archive service) [2, 3].

Also, one has to distinguish different use case classes, in particular those where deleting of information must occur after a relatively short time in order to respect privacy requirements, from those of very long term archiving without deletion of any information. The latter case includes contract conservation for 30 years and the former includes storage of customer usage data for telephone operators. Such use cases are not totally independent, and are chained together, i.e. the same information or some extracts may be transferred from one use case to another, in particular, after the normal and active lifetime of a document like a contract, the same document may be transferred to an archive where it is normally kept classified for an another long period and made public afterwards. All these use cases share at least partially the need for stability of the archive (existence, integrity, authenticity, etc.).

In order to provide authenticity of archived records and accompanied signatures in the future it is obvious that in addition to the whole techniques and organization required to preserve a digitally signed document, it is necessary that complete reference information should be preserved in a trustworthy manner, even if this information is not included in the signed document itself. Reference information also consists of statements about format transformations that have been applied to the information in order to keep it exploitable by current technology.

This paper describes some important design, development and proof of concept deployments based on the ongoing multidisciplinary work of various individuals and groups, whose purpose is to define the basic concepts and technological elements of an electronic preservation service, known as a Trusted Archive Authority (TAA). The presented work is related to the on-going development of the IETF proposed standard for Long Term Archive and Notarization Services (LTANS) [3, 4, 5] and ETSI definitions for XML Advanced Electronic Signatures (XAdES) [2].

2 Electronic Records Preservation

Electronic records containing information of any type are by nature vulnerable to alteration, erasure and obsolescence. In this context several specific issues should be taken into account when addressing the preservation of electronic records and designing a Long Term Archive (LTA) service. Digital objects to be preserved are in the form of records that are heterogeneous and comprise selected information data elements that are pulled together through a process prescribed by a business or administrative procedure. Identifying the boundaries of such intellectually complex objects and then moving those objects forward through time and through migrations without compromising their authentic status is a significant issue. The degree to which a record can be considered reliable is mostly dependent upon the level of procedural and technical control exercised during its creation and management in its active life. Issues such as these that relate to the general preservation of archival materials have been addressed from several different perspectives [2, 6]. However, only a few initiatives [2, 7, 8] pay attention to the preservation of record authenticity and prolongation of digital signature validity, that may be achieved with periodic reviews of time related evidence and use of cryptographic mechanisms such as reapplying times tamps by the use of stronger algorithms.

Initially, problems appear when dematerialization of formal business (e.g. electronic invoicing) and governmental processes (e.g. electronic taxation) takes place. Electronic documents (as a direct outcome of such processes) require special treatment dictated by the general technical principles, international directives and national legislation. These problems are related to the loss of features of the well known paper based infrastructure. Some internationally supported actions have triggered the general public to move towards solving such problems [2, 3, 4, 5, 8]. Public key infrastructure (PKI) enabling systems delivered the very first answer to authenticity issues for formal electronic documents. However, PKI techniques are affected by the same rapid technology progress and most worryingly, the possibility of losing cryptographic reliability. Actions proposed by different standardization groups (in particular ETSI and IETF) are trying to address general preservation

questions by providing different, yet parallel approaches for preserving electronic record intactness and prolonging the validity of cryptographically applied signatures over long periods of time. While it is still not clear which approach will pave the road of e-business and e-government evolution in the years to come, general understanding of the problem has resulted in the unveiling of fundamental concepts of electronic heritage preservation, which this paper will reveal.

Both approaches answer the question of electronic record (or digital objects in general) consistency and digital signature validity over time. We can define one approach as a single instance technique and the other as a service based preservation solution. Although both approaches share similar technological concepts, the main differences derive from the conceptual understanding of the problem. A single instance approach builds on top of digital signatures, addressing mainly the problems of signature time restrictions as proposed by advanced digital signatures – XML Advanced Electronic Signature (XAdES) proposed by ETSI and its equivalent documented by the IETF's RFC 3126 Archive Electronic Signature (ES-A) [9].

The single instance approach enhances digital signature parameters with additional reference information such as digital certificates, Certificate Revocation Lists (CRL), etc. Applying a single or multiple timestamps over a signature and its reference data, assures long term validity even when the reference information ceases to exist. In addition, digital time stamping assures consistency and time evidence of stamped data. This way we can assure the consistency of a single archived object, not part of any multi object instance.

The latter approach, as currently under way in the IETF, answers the general question of digital object preservation. Designed as a service it mainly collects archive data and builds evidential information for demonstration of data existence and data integrity over the required archival time. Enhanced with reference information the service design may provide an attestation of a digital signature's validity over any period of time. It is important to understand that both approaches share the same fundamental techniques (e.g. reference data collection and digital stamping), while the service based archive does not rely on a particular technique (time stamping) but uses Evidence Record Syntax (ESR), which may come in a form of e.g. time stamp or hash linking techniques or any substitute technological means.

Also, the latter approach tries to define a global infrastructure of technological models and components for setting up and maintaining a preservation service as an internal or external type of service integrated in e-business or e-government enabling infrastructures, regardless of a record type. This paper describes an archive service in detail together with its main building blocks: Archive Object (AO), Evidence Record (ER) and Archive Interaction Protocol (AIP).

For clarity, we first identify the basic requirements for the LTA service:

Data Retrieval

An LTA service must retrieve different types of data. Such data may come in different forms and formats, while the service concept recognizes two general data types: unsigned (raw) and signed data. Both data types may be provided in plain or encrypted form (e.g. for confidential purposes). Different data types have impact on LTA service performance. Signed data require additional preservation related information to prolong the validity of the applied signatures, while the general preservation concepts and procedures remain the same.

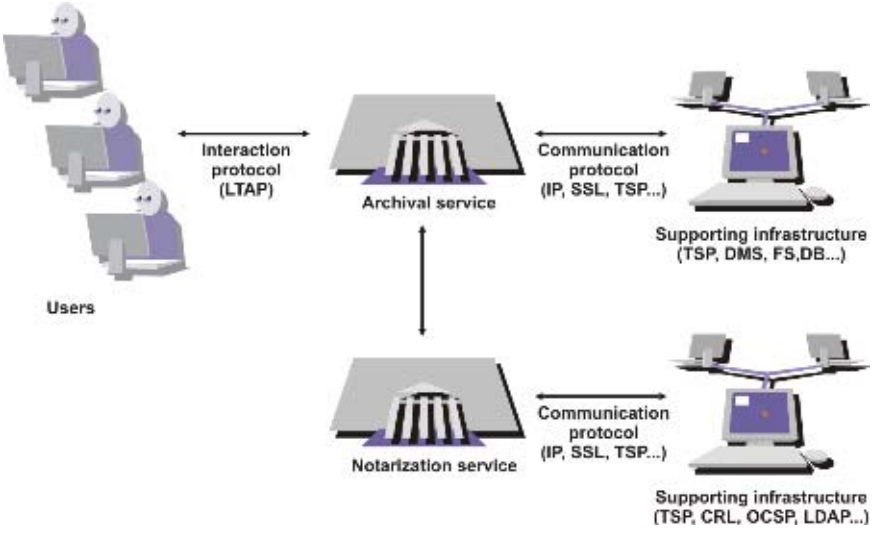

Fig. 1. Trusted electronic archive infrastructure

Demonstration
An LTA service operates on single instances or aggregation of data in a way that produces and provides authentic information for the demonstration of data existence, data integrity and data validity continuously over the requested preservation period. Data existence refers to the time when the data entered the archive or in other words, the existence of data on a timeline is when the evidence record is generated. Integrity provision must rely on continuous maintenance of data in its original form identifying any alteration of the data, while long term validity applies on those types of data whose validity may evaporate over time due to formal and technical constraints.

Interaction
A user interacts with an LTA service using a set of functions interpreting formal methods used against an archive. We distinguish five basic functions of a user interaction: archiving, status checking, validating, exporting and deleting of archive data.

Following these principal requirements, conceptual basic design elements are defined. An archive object is a collection of information including archive data objects, security attributes and the associated conservation information that are archived and have to be preserved for a long time by the LTA service. Conservation information delivers proof of existence of events in time and may be used to resolve a dispute about various aspects of authenticity of archived data objects. Conservation information includes some formally required data objects (e.g. document author or owner), verification data like certificates, revocation information, trust anchors, policy details, role information, etc., and most importantly, a collection of evidence compiled for a presented single or collection (aggregation) of data objects. Evidence information may include timestamps or other time related evidence information.

A timestamp and its technology interpretation in this context is a signed confirmation generated by a Time Stamping Authority (TSA), declaring that a data item existed at a certain time. The third building block of a trusted archive infrastructure is the interaction between users and the service, which handles requests and delivers responses, (i.e. an operation request and attestation of a successfully finished operation). The interaction with the service must follow the formal interpretation of using an archive and must deliver formal proof of events triggered by the user and carried out by the service.

2.1 Archive Object

Archive objects are treated by the LTA service in a common and reproducible way over a long and possibly undetermined period of time. An archive object is defined as a collection of associated data having the same collection identifier. Data are grouped either by the submitter or by the entity providing preservation capabilities. In general, an archive object is assembled by the LTA service after retrieving archive data sent by the submitter. Such data (e.g. electronic document) may be associated with other data (e.g. documents) or meta information (e.g. document owner), which should be treated by the service within a single collection instance. The service then collets required information and other data objects to demonstrate the existence of the archive data (e.g. an exact time when archive data entered the archive) and archive data integrity. The purpose of a collection is the ability to act on several data objects as a whole, in particular to provide an evidence record for the collection.

An archive service should optimize its internal treatment of data collections in such a way that all known elements of a collection can be easily handled by operations that use the collection identifier as a parameter. For example, in order to simplify management and retrieval, elements can be grouped together on the same storage media. The server also creates evidence data solely based on the elements of a collection, in particular, the production of e.g. hash trees. Archive requirements [2] address data aggregation as grouping archive data collections for the purpose of producing a single evidence instance. Such grouping may be based on hash linking [10], where nodes represent message imprints of single data objects and only the root value is time stamped. Data aggregation may be based on different parameters (e.g. aggregation of all data submitted by a specific user) for specific requirements (e.g. scalability). A single time stamp instance may unburden the overall preservation infrastructure (with significant implication on the overall charging model).

In e-business and e-government evolving scenarios, an archive object must include some binding information that may be formally required by the legislation (e.g. for court evidence). Such information is generally recognized as meta information including data owner and data creation time and location. Data description is a general problem already addressed by several projects and initiatives [11, 12]. A trusted archive service does not follow specific directions, however at least simplified information elements have to be included and openly defined according to implementation or formal requirements.

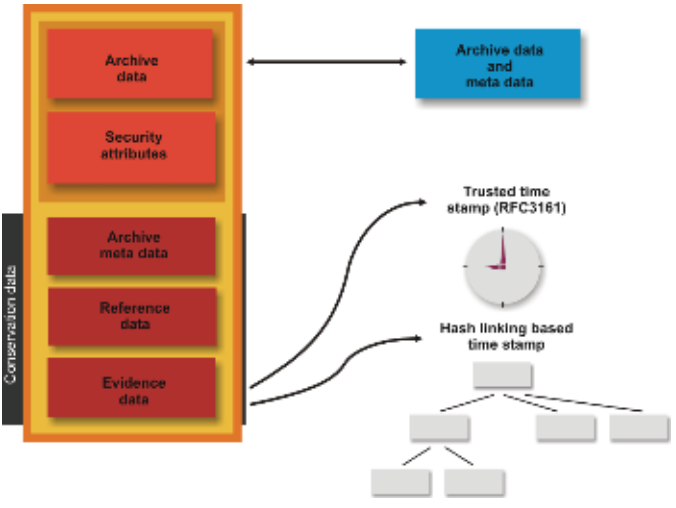

Fig. 2. Archive object

2.2 Evidence Record Syntax

This section briefly describes the syntax and processing of an Evidence Record Syntax (ERS), designed for long-term non-repudiation of existence of data, which particularly should be used for conservation of evidence of e-business or e-government related electronic records. Important examples are digitally signed data, which sometimes have to be archived conclusively over long periods of time (30 years and more). During the archiving period hash algorithms and public key algorithms or their parameters can become weak or certificates can become invalid. To avoid the digital signatures losing their probative force it has to be provable that the data already existed before such a critical event. One proposed approach is to generate timely digital stamps for these data and to renewal them during the entire archival period, but it can also be possible to rely on the physical protection and timely ordered data storage only. The necessary renewal interval for time stamps may also be close to the need for general data maintenance and transformation.

The purpose of ERS is to standardize data formats and processing procedures for such time evidence in order to be able to verify and communicate preserving evidence. An ER is a unit of data, which is to be used to prove the existence of an AO or an aggregation of AOs at a certain time. The ER contains time related information, generated during the long period of archiving, and possibly other useful data for validating such information. ERS specifies the syntax for an ER, which contains archive time stamps and its associated data. An ER may be related to a single data object or to a group of objects and is included in each AO. An archive time stamp can be derived from hash-trees, first described by Merkle [10] and later evolved in linear or binary hash linking schemes [13, 14], optionally combined with a single time-stamp. The leaves of the simple hash tree are hash values of data objects. A time stamp is requested only for the root hash of the hash tree. The deletion of any data objects referred to in the hash tree, does not affect the provability of other data

objects. This tree can be reduced to a few small sets of hash values, necessary to prove the existence of a single data object or a data object group. A LTA may use the same or similar tree techniques to request a single instance of ER for AO grouping.

However, time stamping is based on the same technique as used for digital signing and therefore suffers the same consequences. Before the cryptographic algorithms used within the archive time stamps become weak or the time stamp certificates become invalid, archive time stamps have to be renewed by generating a new instance of them. ERS distinguishes two ways for time stamp renewal, the simple time stamp renewal method and the complex hash tree renewal method. In the case of the simple renewal procedure, the previously generated stamps only have to be hashed and time stamped with a new archive time stamp. In this scenario it is not necessary to access the initially archived data objects themselves, since the algorithms used for the previous time stamps are recognized as being reliable enough.

The simple time stamp renewal method is not sufficient if the hash algorithms used become insecure. In this scenario not only the initial time stamps but also the referred to archived objects have to be hashed and time stamped again by a new archive time stamp. Therefore in such conditions, it is necessary to re-access all the referred to AOs.

ERS foresees the inclusion of reference data for applied time stamps. In the case of an RFC 3161 based time stamp [16], the certificates, CRL lists or OCSP responses needed to verify the archive time stamp must be stored in the evidence record itself. Using reference data, the validation of time evidence can be performed even in cases when the time stamping authority has ceased to exist.

Hash linking concepts [13, 14] propose time resistant time stamping not based on digital signing mechanisms over the long term, but solely on hash trees and validation against published hash roots. These techniques have already been implemented and operated as a service and are available as open source as a result of the OpenEvidence project [15].

2.3 Trusted Archive Interaction

In this section the LTA service interaction protocol is described. The protocol allows clients to interact with an LTA service in a technical and formal manner. The Long Term Archive Protocol (LTAP) is intended for a client-server architecture, where the client is simply presented as an end user (a physical user or another service) and the server as an LTA service. The protocol does not include other transport and security means but binds itself to underlying transport and security protocol layers e.g. SSL, IPSec or SAML.

The protocol uses a messaging service to exchange requests and responses between requesting and responding entities. The data structure of messages wraps basic protocol information elements together with archive data into a single or multiple messages. Protocol messages include operational functions pushed towards the preservation service, and communicating entities may use an arbitrary exchange mechanism (e.g. http, SOAP or e-mail).

In this context an LTA service is understood as an entity that accepts formal user requests including data objects to be preserved, and performs specific actions on the

received requests such as object management, aggregation, hash linking, evidence creation, etc. The LTA physical interface must therefore enable clients to:

- submit data to the LTA (and request creation of evidence records for the data)
- check the status of submitted data
- extract, transfer or simply retrieve data (archive data and evidence data) from the LTA
- delete data and/or evidence records from the LTA
- verify the integrity and authenticity of data archived by the LTA

The primary aim of this protocol is to ensure the technical and formal interaction between a user or client and a service. The result of the interaction is the attestation of procedures performed by the LTA (e.g. archive data). An LTAP request is defined as a global message type including all supported services. Request messages must always carry essential information declaring the type of service requested and the data objects associated with it. According to the service requested, specific request message fields are used. The data fields of an LTAP request are as follows:

- Request information with mandatory information on specific requests including entities, service, data and policy identification and some configuration parameters.
- Raw data, i.e. data to be preserved by an LTA server. Raw data can be presented also as a reference to some remote resource.
- Meta data providing some information about the raw data.
- Authorization and authentication information of the entities participating in the procedure.
- Other (not yet defined) information required for supporting functions, e.g. charging and billing.

An LTA server may use the LTAP protocol to provide users with some attestation of the procedures performed. Response information may provide enough information for third parties about the procedures performed, such as data verification. Such a scenario is useful when the third party is unable to provide the capabilities to investigate the authenticity and integrity of the archive data and its digital signatures, based as they are on complex sets of evidence data (hash links, references, etc.). However, for such procedures to exist, a trust link between the involved parties must be present, specifically, the third party must trust the archive service to perform its procedures correctly.

Furthermore, attestations of procedures taken are essential for the audit trail, e.g. what was archived and what was deleted. Therefore, in LTAP for some request types (e.g. deletion) authentication is required (e.g. based on the user's digital signature). The LTAP protocol does not include security assertions by itself. It uses available means to protect messages and assure integrity and authenticity (e.g. digital signatures). An LTA server may respond in an asynchronous way (e.g. e-mail), when procedures performed by an LTA server require longer processing response times. The structure of such responses may follow the same structure. In all cases, specific procedures require attestations about the performed actions to be stored for later audit. Attestation preservation may require the same long term persistency as the archive data.

3 Long Term Archive Service

An LTA service has to be designed as a service that provides answers to several problems associated with long term data preservation and its concepts built on top of defined building blocks. As described previously, according to the IETF specifications, an LTA service provides the means of preserving digital objects that are needed to prove the existence and integrity of data objects to users of the service and in case of dispute to the court. These means are mainly hash trees, references and time stamps, periodically generated during the archiving period of the data objects, taking into account the possible special legal requirements for digital signature validity prolongation.

Signature validity extension relies on collecting signature reference data, which proves that document signatures existed in the preserved form before the reference data expired or before the algorithms used for signing became weak and unreliable. In this context, rather than producing the reference data itself, the LTA service uses verifiers provided by PKI services like the Simple Certificate Validation Protocol (SCVP) [17], Data Validation and Certification Server (DVCS) [18], or Digital Signature Services (DSS) [19]. This separation of the functionality of the services is recommended in order that separation of roles and processes be assured in order that the archive service be capable of operating without knowledge of the numerous signed data formats and document formats belonging to the stored digital object.

3.1 Service Deployment

When defining the LTA service, several application scenarios for the long term were envisaged. An LTA service must provide a user with the procedures that assure complete confidentiality of the preserved information. Also, an LTA service may lack the capability of providing a competent and reliable data store for millions of documents. Therefore two basic scenarios are specified below:

Integral document conservation service assuring long term non-repudiation – the LTA service retrieves archive data objects like signed or unsigned documents for identified and authenticated users. It generates an evidence record for these data objects and obtains necessary reference data over a given time or until a request of deletion by this authorized user is received.

Pure long term non-repudiation document conservation service – the LTA guarantees non-repudiation of existence of archive data only. It periodically generates time stamps and obtains additional verification data for a given period of time. It stores archive data (e.g. documents, but also relevant parts of documents containing signatures) locally for the purpose of evidence information creation. Alternatively (for confidentiality purposes) it may only retrieve an interpretation of archive data, e.g. message imprints. It is not a document archive and therefore does not provide retrieval of documents and no deletion of data objects. It may be part of a different application system (e.g. Content Management System) or a stand alone service for subscribed users with a deployed content management infrastructure.

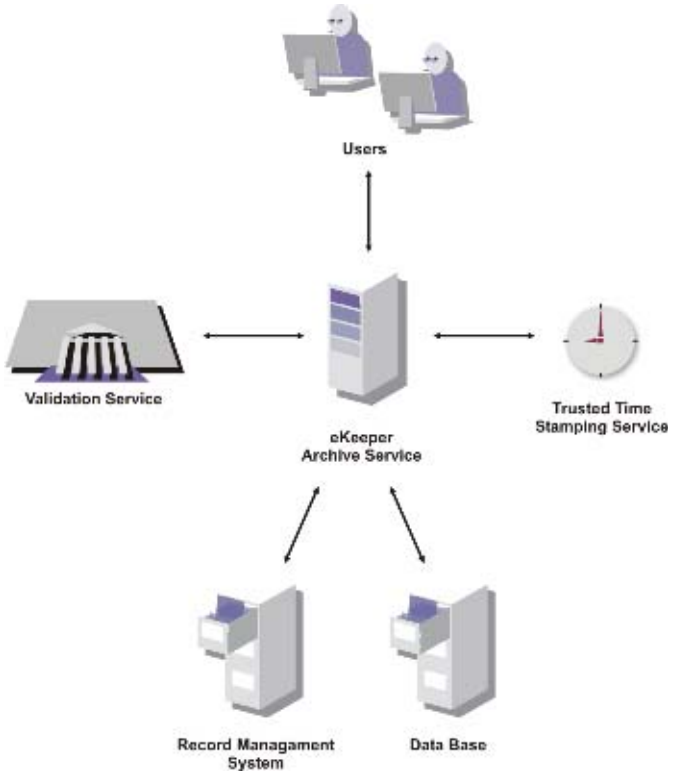

Fig. 3. Deployed LTA service infrastructure

An LTA service is based on several technology entities. Technology blocks that are specific for LTA services, which were described in a previous section, are defined as archive infrastructure, while the remaining blocks are described as supporting infrastructure (Figure 3). Client applications serve as the interface for using the service. Communication protocols consist of several layered elements, the communicating protocols (TCP/IP), security protocols (TLS) and messaging and interaction protocols (HTTP/SOAP), data presentation (XML, CMS…) and the abstract application data (LTAP). The LTA server uses the data management service, data validation service and evidence creation service. Storage capabilities are outsourced from specialized services providing physical reliability (redundancy and disaster recovery devices). An important aspect is that the protocol is essentially asynchronous concerning the acceptance of responsibility by the archive service, i.e. archiving activity is confirmed when the appropriate operation to physically secure the object has taken place (e.g. backups taken). The supporting infrastructure is not relevant for service design, since the technology is known to exist and proven to operate correctly.

3.2 Demonstration Service

The eKeeper demonstration trusted archive infrastructure [20] is built around existing technological blocks with critical elements designed and developed for the service purpose. The deployed electronic records preservation service is the authors own implementation and addresses the conceptual and technical results of the work done by the standardization bodies. The client is an application user interface allowing interaction of a user with the archival service through a dedicated application or web interface. Evidence creation and archival functions on archive data are performed manually or automatically, based on workflow events or human interaction by use of the service client. Each document or batch of documents submitted from the application of choice via the client is enveloped in a tamperproof and trustworthy manner. The current service is using as test application Document Management System (DMS) Documentum™ version 4.2 as a record management server. Each submitted document is analyzed and handled by the LTA server in a way to provide unique interpretation in a digested form (fingerprint). Applied digital signatures are validated using reference data (e.g. CRLs) obtained from accredited CAs upon archiving, and complementary data is collected to prove the existence of a signature and its validity at the exact time of archiving. The trusted time stamp server provides digital stamps compliant to RFC 3161 using either a local nCipher Document Sealing Engine 200 (hardware based) time stamping solution based on a trusted time source (collected from the Global Positioning System) for creating evidences and preserving the electronic record intactness, or the time stamping demonstration service provided by EdelWeb, where a complete historical log of all issued time stamps is maintained.

The service automatically executes periodic re-time-stamping to prolong the digital object integrity, security attributes validity and the original time evidence proof (initial timestamp). This process takes into account the whole archived content including the archive data itself, the metadata provided and the security attributes associated with it.

Prior to archiving of a digital object, validation mechanisms verify the validity of applied signatures. Verification is based on reference information, which must exist and be valid at the time of evidence creation. The service collects reference data (certificates, CRLs) and performs validation. Each archive data is associated with meta information provided by the user. AOs keep complete information required for preserving the document integrity, evidence information and prolonging the lifetime of the security attributes over undefined periods of time.

Scalability of the LTA service is mandatory. It is expected to operate with large amounts of data such as bank transactions or electronic invoices. Critical transactions require prompt responses from the service and prompt execution of the functions requested. Next to reliability, scalability is of highest importance in e-business and e-government environments. The deployed LTA service builds hash trees that significantly reduce the amount of generated data in the preservation procedures. These trees combine the digital object archival data in groups and apply single timestamps over the aggregated archive data. The trees however provide full consistency when an object or a group of objects is later removed from a batch.

Document physical storage is out of the scope of this type of archival service since physical storage technology exists on the market and is proven to comply with

security requirements. The service uses standard protocols and mechanisms for secure and auditable interaction between all components and provides management of the AOs. Archive data meta fields are minimal and constitute the basic data required to comply with the formal and legislative requirements. Used archival meta fields are: digital object internal/external identification, digital object title, digital object owner, digital object creation location and time. In case the data object is digitally signed, the reference information collected is: digital certificate(s), digital certificate chain(s) and CRL list(s).

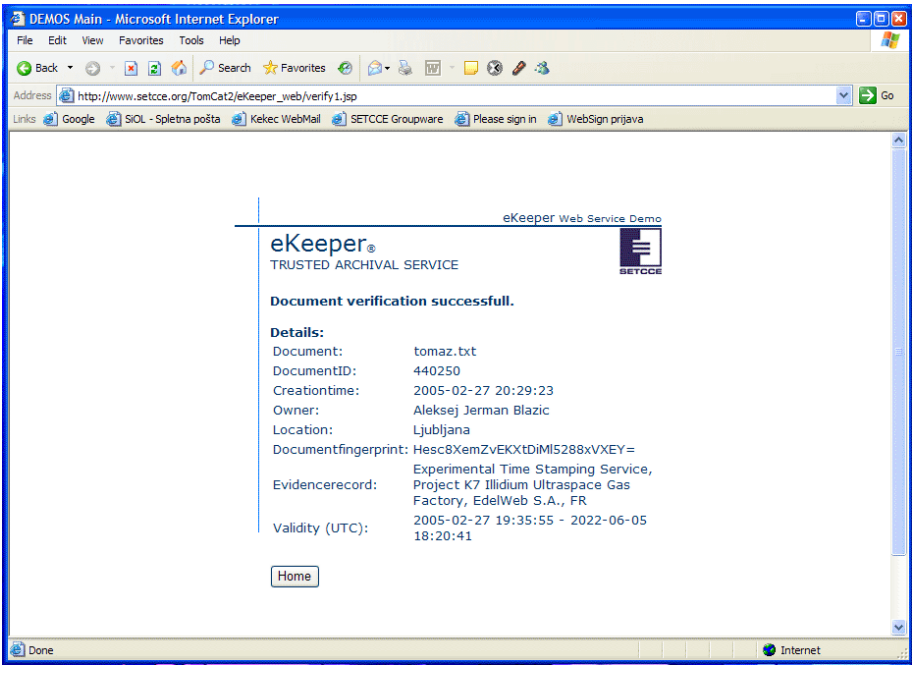

Fig. 4. Demonstration trusted archive service

The archive service is also designed to operate with multiple evidence mechanisms providing redundant proofs of integrity and evidence. This is accomplished mainly by controlled replication of functions. Multiple trust anchors provide redundancy information which is needed when evidence records become crippled (e.g. time stamp root certificate revocation, hash break, etc.). Optionally the re-encryption of archive data is provided for cases when users require storage of non encrypted data in their site and are discarding the encrypted data used by the archival service (the data sent to the archival service is encrypted).

A deployed archive service does not address the global electronic preservation problems. Its purpose is to demonstrate the techniques and mechanisms for maintaining long-term stability of electronic records from the standpoint of continuous integrity protection and digital signature validity maintenance. Compared

to related work it presents complementary mechanisms for preserving electronic heritage on a long term basis. The eKeeper service therefore relies on technology infrastructures for maintaining electronic records from the viewpoint of digital object management (e.g. as defined by [1] and [6]).

4 Conclusions

Responding to a request from a national regulation body for dematerialization of records such as formal e-invoices and e-orders, SETCCE has designed and developed the eKeeper system and set up a demonstration pilot in order to provide practical experience in order to provide feedback and contribution to the ongoing IETF LTANS work. The public demonstration service uses EdelWeb's time stamping demonstration service.

The eKeeper archive web based service became available on 1st October 2004 to demonstrate the functions and mechanisms of the LTA service. The technology is already being deployed for some governmental organizations like the Ministry of Defense and Constitutional Court. The web based service is still in its pilot phase and further improvement will follow. The initial results are encouraging. The next step to follow is the definition of notarization services required for attesting security attributes (e.g. digital signatures) based on standards such as DVCS.

Some parts of the developed technology blocks are currently being proposed as Internet standards and will serve as an important element in the future of electronic archive and notarization services. A deployed web service answers the main questions of preserving formal documents in their original form and maintaining the validity of the associated signatures. It is certainly a service that is crucially needed as documents such as digital contracts, tax declarations or invoices are evolving in electronic form already. All these documents need special care in order to ensure persistent readability, permanent availability, integrity and proper protection of confidentiality. Standardization is an important process for the wide scale deployment of e-business and e-government enabling services and related electronic heritage preservation.

References

1. Consultative Committee for Space Data Systems: Space data and information transfer systems – Reference Model for an Open Archival Information System (OAIS) – Blue Book, Issue 1. CCSD, S2002. Also available as: International Standards Organization – ISO 14721.
2. European Telecommunications Standards Institute: XML Advanced Electronic Signatures. ETSI, 2004.
3. Wallace C., Brandner R., Pordesch B.: Long term archive service requirements, draft-ietf-ltans-reqs-03.txt. Internet draft. IETF, 2004.
4. Brandner R., Hunter B.: Evidence Record Syntax – ERS, draft-ietf-ltans-ers-00.txt. Internet draft. IETF, 2004.
5. Schmidt A., Gondrom T., Masinter L.: Requirements for Certification Services, draft-ietf-ltans-notareq-01.txt. Internet draft. IETF, 2004.

6. Assistant Secretary Of Defense For Command, Control, Communications And Intelligence: Design Criteria Standard for Electronic Records Management Software Applications – DoD-5015.2-STD. Department Of Defense Office of DASD, 2002.
7. European Electronic Signature Standardization Initiative: Trusted Archival Services (Phase #3, Final Report). EESSI, 2001.
8. ArchiSig project: Conclusive and Secure Long Term Archiving of Digitally Signed Documents, http://www.archisig.de/english/index.html.
9. Pinkas D., Ross J., Pope N.: Electronic Signature Formats for long term electronic signatures, RFC3126. IETF, 2001.
10. Merkle R. C.: Protocols for Public Key Cryptosystems. IEEE, 1980.
11. Dublin Core Metadata Initiative, http://www.dublincore.org.
12. Nottingham M., Sayre R.: The Atom Syndication Format, draft-ietf-atompub-format-05. IETF, 2005.
13. Buldas A., Laud P., Lipmaa H., Willemson J.: Time-Stamping with Binary Linking Schemes. CRYPTO, 1998.
14. Bayer D., Haber S., Stornetta W. S.: Improving the efficiency and reliability of digital time-stamping. Communication, Security, and Computer Science, 1992.
15. Open Evidence project, http://www.openevidence.org.
16. Adams C., Cain P., Pinkas D., Zuccherato R.: Internet X.509 Public Key Infrastructure Time-Stamp Protocol – TSP, RFC 3161. IETF, 2001.
17. Malpani A., Housley R., Freeman T.: Simple Certificate Validation Protocol – SCVP, draft-ietf-pkix-scvp-17.txt. Internet draft. IETF, 2004.
18. Adams C., Sylvester P., Zolotarev M., Zuccherato R., Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols – DVCS RFC3029, Internet draft. IETF, 2001.
19. Federal Information Processing Standards: Digital Signature Standard – DSS, FIPS 186. FIPS, 1994
20. Garrett J., Waters D., Preserving Digital Information: Report of the Task Force on Archiving of Digital Information commissioned by the Commission on Preservation and Access and the Research Libraries Group. Report of the task force on archiving of digital information. Washington, D.C. 1996.
21. Feeney M., Towards a national strategy for archiving digital materials. Alexandria, 1999.
22. Thibodeau K.: Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years. Conference Proceedings: The State of Digital Preservation: An International Perspective. Washington, D.C., 2002
23. Wallace C., 2004. Trusted Archive Protocol – TAP , draft-ietf-pkix-tap-00.txt. Internet draft. IETF 2004
24. Klobucar T., Blazic B. J., An infrastructure for support of digital signatures. Journal paper: Informatica, 1999.
25. Myers M., Ankney R., Malpani A., Galperin S, Adams C., X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, RFC2560. Internet draft. IETF, 1999.
26. Lorie R., A Project on preservation of digital data. On-line magazine: RLG DigiNews, Issue 5(3). On-line, 2001.
27. InterPARES project: International Research on Permanent Authentic Records in Electronic Systems, http://www.interpares.org.
28. EdelWeb time stamping project, http://timestamping.edelweb.fr.

Legal Security for Transformations of Signed Documents: Fundamental Concepts

Andreas U. Schmidt[1] and Zbyněk Loebl[2]

[1] Fraunhofer Institute for Secure Information Technology SIT,
Dolivostrasse 15, 64293 Darmstadt, Germany
Andreas.U.Schmidt@sit.fraunhofer.de
http://www.sit.fraunhofer.de
http://www.math.uni-frankfurt.de/~aschmidt
[2] Central European Advisory Group, Betlémská 1,
110 00 Prague 1, Czech Republic
ceag@ceag.cz
http://www.ceag.biz

Abstract. Transformations of signed documents raise questions of technical and organisational nature which render the legal security of the transformed document doubtful. In particular, digital signatures of originals break depriving documents of probative force. This report elucidates legal problems, and introduces fundamental concepts of legally secure document transformations in a deliberately generic, application-independent way. A process analysis of transformations of signed documents is carried out to elicit common security requirements. This leads to the solution approach *transformation seal*, a cryptographically secured container used to ensure legal security for transformed documents by securing the content's integrity, attesting a transformation's correctnes, and attributing it to a responsible party.

1 Modern Legislation Governing Electronic Transformation and Archiving

The principal regulation of legal issues related to the electronic transformation and archiving of documents was created as early as 1996 in the UNCITRAL Model Law on Electronic Commerce [1] (UMLEC). Since its inception, the UMLEC has served as an inspiration for the most developed countries in the preparation of national legislation, and the European Commission draws on it in the drafting of European legislation addressing certain aspects of electronic communication, e.g., the recently adopted EU Directive 2004/17/EC uses the UMLEC's definition of a data message (Article I (11)).

1.1 UNCITRAL Model Law

The UMLEC presumes a consistently applied technology neutral approach and distinctions between the notions of 'data message', 'writing', 'original', 'signature', 'legalisation/notarisation', 'legal effect/evidential weight', and 'document

David W. Chadwick and G. Zhao (Eds.): EuroPKI 2005, LNCS 3545, pp. 255–270, 2005.

archiving'. Basically, the UMLEC defines a writing at the lowest requirement level as anything in any form and on any carrier that may be reproduced and read for the purposes of subsequent reference [1, Article 6]. Therefore, any e-mail messages or any texts in electronic form ought to be viewed writing as well as data messages, regardless of the level of security that may apply to them, or regardless of whether their source is apparent, let alone trustworthy.

Further, the UMLEC defines an 'original', not only with regard to the form of the original document, but also with regard to the integrity of content of it [1, Article 8]. Therefore, an n-th electronic copy of a document is deemed to be an original if the integrity of its content from the moment when it was generated in its final form can be proved. In particular, this notion was adopted in the US.

In France and the UK, this provision of the UMLEC was commented to the effect that it is difficult, currently, to talk about an original document in electronic form, and that this notion ought to be abandoned altogether. Contrary to that, it was necessary to focus on stipulating general conditions pursuant to which documents in any form (on paper or in electronic form) have full legal effect/evidentiary weight comparable to that of original documents on paper. If such conditions were satisfied, it would be possible to present the court for instance with an electronic document containing information from a document that was originally on paper, and the court ought to give such document legal effect identical with the legal effect afforded to the original document. The problem with this notion arises in situations where the law expressly requires that an original be submitted. There are only a few such provisions, however, and they can be amended.

The UMLEC also stipulates general conditions that affect the full legal effect/legal force of electronic documents [1, Article 9(2)]. This provision places an emphasis on securing the integrity of information, authenticity of the originator and credibility of the process of generation, storing and communication of data messages. The satisfaction of such conditions is to a significant extent influenced by requirements regarding a credible (authenticated) electronic signature. The UMLEC sets out the requirements applicable to electronic signatures in its Article 7. Owing to the principle of technological neutrality, it was impossible to adopt for general electronic signatures the same concrete presumptions which exist it in the EU Directive 1999/93/EC and national laws in EU member states[1] with respect to authenticated electronic signatures based on asymmetric cryptography. Such legal presumptions concern precisely the equivalent of a handwritten signature (proof of authenticity) and the integrity of content[2]. A part of the professional practise now views their absence as a drawback - e.g., in the USA where the principle of technological neutrality was also adopted.

The UMLEC distinguishes between these general conditions and electronic legalisation/notarisation. It merely recommends in this regard that any obstacles contained in national laws that prevent legalisation through electronic means be removed (e.g., changing the requirement of affixation of an official seal, etc.) [2,

[1] e.g. Czech Act on Electronic Signatures, Act No. 227/2000 Coll., as amended.
[2] Directive 1999/93/EC.

Article 6]. This recommendation was implemented in all the countries referred to below. The UMLEC further expressly regulates electronic transformation and archiving of documents (Article 10), drawing there on the notions of original, data message and full legal effect/evidentiary weight, and in essence merges all the requirements mentioned above. Section (1) of the said provision sets out the following three (sets of) requirements applicable to a data message that ought to meet the requirements for long-term archiving of documents in any form:

- requirements applicable to the data message (information contained therein needs to be accessible so as to be usable for subsequent reference);
- the data message needs to be retained in the format in which it was generated, sent or received (i.e., the original format), or in a format which can be demonstrated to represent accurately the information generated, sent or received *(this provision is expressly directed at transformation of documents on paper into electronic form)*; and
- such information is retained as enables the identification of the origin and destination of a data message and the date and time when it was sent or received.

1.2 USA

In the USA, an equivalent of the UMLEC was prepared in 1999: the Uniform Electronic Transactions Act UETA. The draft law was prepared by the National Conference of Commissioners of Uniform State Laws which prepares uniform laws for the individual US states. In the meantime, the act has been adopted by an overwhelming majority of US states. In 2000, a federal law on electronic signature, the E-Sign Act, was enacted. Both laws draw heavily on the UMLEC, but further elaborate on it. They represent the latest comprehensive provision of law pertaining to electronic transactions, recognised by experts worldwide. The UETA regulates electronic transformation and archiving in Article 12. In the requirements for an original (indent (d)), the UETA includes requirements for transformation and archiving (indent (a)), as well as requirements for the full legal effect/evidentiary weight of electronic documents (indent (f)). A similar provision of law is contained in the federal E-Sign Act, Article 1 (d).

As the UETA and E-Sign Act do not apply to certain special types of documents, in 2004, the Uniform Real Property Electronic Recordation Act (URPERA) was drafted, pertaining particularly to documents related to real estate and property ownership. This is not a valid law, but was drafted by the above-mentioned National Conference of Commissioners of Uniform State Laws and is to be implemented by individual US states. It provides for electronic transformation and archiving in Articles 3, 4 and 5, where it expressly permits electronic transformation and archiving (Article 4), including the replacement of the requirement of submission of an original document with its electronic record (Article 3). Nonetheless, it does not set out general requirements the way the UETA and E-Sign Act do; instead, it refers to special commissions formed at

state level (state electronic recording commissions - Article 5) that are to formulate the standards applicable to electronic administration of documents falling under the URPERA.

1.3 United Kingdom

The UK accomplished a legal status similar to that of the USA, only not pursuant to special laws but rather by virtue of court rulings. As mentioned above, English law virtually abandoned the requirement of original documents. First court rulings have been rendered in the UK that afford electronic documents full legal effect and evidentiary weight[3]. These rulings pursued requirements similar to those contained in the UMLEC. Moreover, the UK adopted laws implementing relevant EU directives, in particular Directive 1999/93/EC (electronic signatures), Directive 2000/31/EC (electronic commerce), and Directive 2001/115/EC regulating invoicing for VAT purposes, including electronic invoicing.

1.4 France

French law underwent significant amendments with regard to these issues in 2000. The French Civil Code again draws on the UMLEC and defines a data message along similar lines (Article 1316). It further focuses on the stipulation of general requirements for the full legal effect/evidentiary weight of electronic documents (Article 1316-1), again corresponding to the requirements of the UMLEC, and in the US legal regulations and UK court rulings, i.e., authenticity of originator and integrity of content. This provision is interpreted in a way allowing for electronic transformation and archiving[4]. Moreover, the Civil Code stipulates that the requirements for full legal effect/evidentiary weight shall be identical for documents on paper and electronic documents (Article 1316-3).

The Civil Code further stipulates, again similarly to the other examples, that notarial acts may be executed or archived in electronic form in accordance with implementing regulation (Article 1317). The implementing decree, which is about to be finalised, will set out detailed standards for electronic transformation and archiving with regard to documents issued by notaries and persons with similar powers (e.g. bailiffs). There is currently an ongoing debate in France whether it would not be appropriate to issue an implementing regulation also with regard to Article 1316-1 of the Civil Code. French law, similarly to UK law, implemented Directive 2001/115/EC permitting electronic invoicing.

2 Technological Development

The most developed countries in the world no longer debate whether and how electronic transformation and archiving ought to be regulated. Rather, it is the

[3] see e.g. R v Spiby (1990) 91 Cr.App. R186; or R v Shepherd (1993) 1 All ER 225.
[4] see e.g. L'archivage electronique, Frédéric Mascré, September 2003.

implementation of general provisions of such legal regulations that now commands attention. The situation in this area is far from standard anywhere in the world. For instance, in the USA in June 2004, the OCC (Controller of the Currency and Administrator of National Banks) issued a recommendation to banks to exercise caution with regard to a fast practical implementation of the provisions of the E-Sign Act, pointing out the absence of standards and court precedents. According to the OCC, the main problems lacking a satisfactory resolution include the electronic transformation of documents on paper signed by hand. Similar problems are addressed by the state commission established with respect to electronic real property recordation, see Section 1.2. In France, the drafting of an implementing regulation setting out detailed requirements for electronic notarial acts, including electronic transformation, is in full swing. In Germany, the government has been funding an extensive project with the aim of electronic transformation of (state) archives, where emphasis is placed on a safe electronic transformation of documents on paper, signed by hand, as well as on genuine transformations of electronic documents [3]. Several European countries already have first standards regulating electronic invoicing, including electronic transformation of older invoices on paper, or archiving. This may be due to the fact that invoices do not need to be signed, and further due to the existence of Directive 2001/115/EC that expressly regulates the electronic invoicing requirement. For instance, the electronic invoicing standards in France are regulated by a special tax instruction of August 2003. Nonetheless, there is still an ongoing debate as to whether the French law permits electronic transformation of invoices on paper into electronic form.

The IETF formed a working group for long-term archiving and notary services, LTANS [4], working precisely on electronic transformation and archiving. It has already published several Internet Drafts for long-term archiving [5]. At the same time, there are several companies worldwide (like AuthentiDate and IXOS in Germany, XEROX, DOCUMENTUM, and many more) that focus on the comprehensive electronic processing of documents. Their systems offer comprehensive solutions including security functions, and the transformation of documents on paper into electronic form. In most cases, such systems enable more functions than expressly provided for in the law. That may increase the legal risk associated with the application of the said technologies, see for instance the recent recommendation of the American OCC. The DMS industry therefore has a genuine interest in the progress and resolution of associated legal issues.

The above shows that it is currently impossible to identify business models and standards acceptable for electronic transformation and archiving. The central problems of secure long-term archiving and secure transformation of digitally signed documents are closely related, the main connection being the latent threat for long-term usability of documents of data formats becoming obsolete. This necessitates proper consideration of transformations, in principle already in the planning of an electronic archive. Legal regulations often demand timespans for document preservation which are well beyond the expected lifetime of

common data formats. After such time-spans, the 'original document' may acquire another important attribute, namely non-reproducibility, e.g., if the original signer is deceased We do not view the two issues as identical or, as is often purported, transformation as a technical sub-domain of archiving. Rather, a more generic approach suggests itself, one which enables a unified treatment, at least on a conceptual level, of transformations of data formats, electronic notarisations, trusted ingestions and issuance of documents by public authorities, and so on (more examples follow below). The UMLEC follows a paradigm of technological neutrality which translates, in the present context, into 'transformation neutrality'. This means that transformations must be enabled and secured by fundamental methodologies which are not bound to specific data formats or underlying technology like document management systems, cryptographic (signature) algorithms, and PKI. The following sections present our contribution to the ongoing technological and scientific effort in the described problem domain.

3 Secure Document Transformations

3.1 Context-Neutrality

We introduce a context-neutral set of basic notions with the aim of defining what a secure document transformation consists of. Such abstract concepts are needed for two reasons. First, many application contexts, in particular the legal domain, possess genuine terminologies from which special criteria for the assessment of document transformations may be derived. Such notions are to be avoided in a generic analysis of transformations. Second, the properties which render a transformation secure vary strongly between application domains and transformation purposes. A transformation may be carried out for reasons of data protection or secrecy. The need for this can arise for instance when government documents are used in court, and parts of them need to be deleted beforehand. An example from the medical sector where data protection necessitates deletions is given below. But the result of a transformation might also be judged from aspects of monetary value of the result (e.g. digital images at different resolutions). The application context determines security in the concrete case. Legal regulations and considerations are of importance in many domains since they pertain to almost every part of human life and in particular to the exchange of signed documents. Thus, to obtain a concept system which is on the one hand flexible enough to span many application domains but on the other hand independent of them, a certain level of abstraction is inevitable. A second goal of these abstractions is to elicit the interface between the 'real world' context, in which humans ultimately interpret and assess documents, paper as well as digital ones, according to their meanings, and the aspects of secure document transformations which are amenable to a formal analysis. This enables the delineation of limits for the formalisation and consequently the automation of transformations.

3.2 Purpose and Purport of Transformations

The **transformation**[5] of a signed document is the deterministic conversion of a **source document** with a certain purport into a **target document** with a certain purport. The **purport** of any signed document is to be understood pragmatically as the union of its possible utilisations within the context of the given application domain, i.e. those usages of the document which can be realised in it[6]. In principle the purport of the target document can be larger, smaller or equal to that of the source document, but apart from these exceptional cases their respective purports are rarely directly comparable. The **purpose** of a transformation is to obtain a target document with a certain purport from a given source. In general, the purport of the target will be partly determined by the source, often in a restrictive sense. At this point, we do not yet differentiate between contents and signatures which are both counted as contributing to the purport, and can therefore support the purpose of the transformation as well as determine it.

The three mentioned special cases of transformations are fundamental in the sense that other transformations can be considered as mixtures or combinations of them. They correspond to three fundamental purposes.

1. The target has to convey — as far as possible — an identical purport as the source if it is to be replaced by the former. ***Examples*** of **replacement** documents are attested copies (exemplifications) of paper documents[7]; P→E transformations as pre-processing steps of digital workflows. Ensuring the readability of a document for the addressee can often necessitate transformations of data formats, for instance when a document is submitted to a government authority.
2. If only a **partial copy**, restricted to certain utilisations is required, the purport of the target is less than that of the source. ***Examples*** comprise attested excerpts from official records for designated purposes; anonymised versions of documents for reasons of data protection; health records might be anonymised for usage in medical studies, yet keeping attributability to the attending physician (by his signature of the source).
3. A transformation may entail the **valorisation** of a target document with respect to the source, i.e., enable certain utilisations that are beyond those of the source. A simple, yet practically relevant example is the migration of an electronic document format to a new version by addition of an empty field for later use; by addition of an alternative font or other representation a document can become accessible to handicapped persons.

[5] Here we address not only transformations between electronic documents (E→E), but also those involving paper documents as source or target (P→E and E→P).

[6] If the context could be formalised in the sense of a formal languages, a usage would be a model of it in which the document is a valid expression, and the purport would be the union of all those models — but this is hardly ever feasible in practise.

[7] Note that identical copies make no sense as E→E transformations since it is loss-free and the original digital signature remains valid. Here, replacements of signed digital document are understood as resulting from a nontrivial conversion of contents which are free of losses and additions.

It is desirable to distinguish the notion of transformations against genuine administrative procedures and workflows in which documents are recombined and meaningful contents can be added (e.g. another signature to a file in circulation).

Thus we define that valorisations by adding contents or signatures do not fall into the considered category of transformations.

3.3 Faithfulness, Trustworthiness, and Security

To satisfy a purpose, a transformation must fulfil the appropriate requirements of **faithfulness**, to be understood as 'converting the contents faithfully for the desired purpose' as opposed to one-to-one correspondence of source and target contents. Faithfulness pertains to all relevant parts of source and target, including signatures. The referral to revisable properties of source, target, and transformation is a characteristic of the concept of faithfulness. This is in intentionally in contrast to the differences between the semantic content, i.e. the meaning, of source and target, which can hardly be grasped formally. Which properties *must* be inspected to assess faithfulness depends on the purpose of a transformation. What *can* be inspected, depends on the source and target documents as such.

Examples: Faithfulness can be assessed on very different levels. It can be sufficient to check resolution and colour depth of a scanner, or necessary to compare source and target letter-wise. The adequacy of the source's data format can be as relevant as its printing quality if it is a paper document. Properties of the transformation may be important, e.g., that the conversion algorithm eventually deletes all personal data in a document that is to be anonymised.

Essential for faithfulness is also the question whether the data formats of source and target are appropriate to present all signed contents correctly. This is necessary to enable forensic inspection of a transformation and requires a proper consideration of the presentation problem[8] for signed digital data.

Without appropriate security measures, the *a posteriori* survey of faithfulness cannot be carried out. Thus, in order to arrive at a **secure transformation**, a record must be kept which asserts that the target has the right faithfulness to the source according to the purpose of the transformation. The **trustworthiness** of this assertion means that it can be retraced at later times, what kind of transformation has taken place and how, that faithfulness has been assessed and by whom, and finally who is responsible for the transformation and the assessment of the result. The necessity to enable forensic inspection sets high standards for trustworthiness in that the target must serve its purpose *even if the source is no longer available*, the most important examples being non-reproducibility and obsolete data formats.

Several instances can assess the faithfulness of a transformation and attest it at the end of the process. In a large-scale application, the transformation system itself can affirm that a certain algorithm was applied for data conversion and,

[8] Also termed 'What You See is What You Sign' (WYSIWYS) problem [6,7].

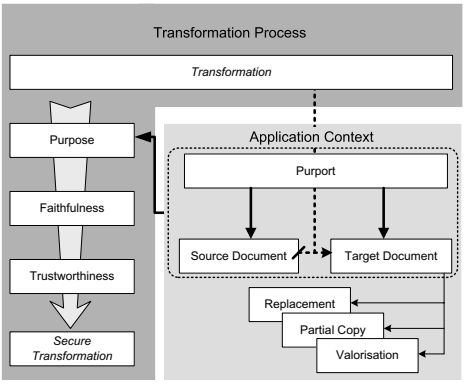

Fig. 1. Fundamental concepts of secure document transformations

e.g., that source signatures have been verified successfully, whereas in the case of notarisations it is necessary that an authorised person inspects faithfulness, and establishes trustworthiness by noting the inspection result and confirming it with his signature. Figure 1 compiles the notions introduced, and is to be understood as follows. A secure transformation is ensured through the trustworthiness of faithfulness for a given purpose. In turn, the purpose is the conversion between source and target with their respective purports. A central result of this system of concepts is that application context and transformation process are linked exactly where the purpose is determined by the desired changeover of purports. Here lie the main difficulties for the formalisation and practical realisation of secure transformations.

4 Process Analysis of Document Transformations

To fulfil the requirements of a secure transformation laid out in section 3, it is not enough to convert the contents of a document 'in all conscience'. Even in the case of simple format conversions, additional process steps and organisational measures are required to ensure security. To elicit these requirements, a procedural analysis of a generic transformation process is helpful. In the following, a transformation is presented as a sequence of phases, independently of the kind of transformation (P→E, E→E, or E→P). This yields the maximal set of high-level **transformation phases**, with the understanding that in special cases some of them can be less important, be subsumed under or combined with another, or be parallelised if they become logically independent. Figure 2 gives an overview.

4.1 Pre-processing

Classification. In an initial step the source document is inspected to determine the purpose of the transformation. Apart from ascertaining source formats like

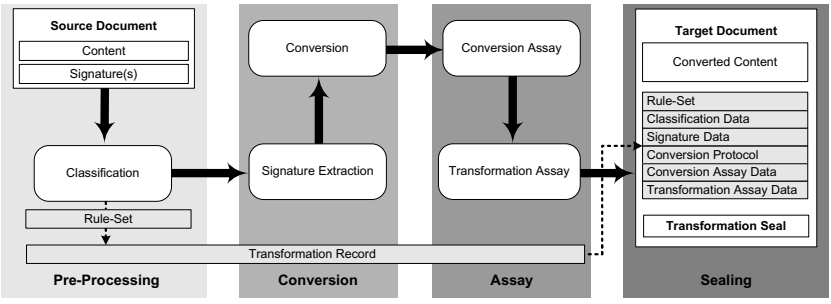

Fig. 2. Processual view on secure document transformations

'Word document' or 'technical drawing on paper', this is essential for the whole following process. The classification does not only determine the relevant properties of the source but also those of the target and the transformation, which have to be satisfied to achieve the desired faithfulness. From this classification follow the rules that govern the way in which trustworthiness is to be established, in particular which data from the process must be kept for forensic inspection and which checks have to be performed on the target (see below).

Examples: Assume the source is a construction drawing that is to be replaced for electronic processing of an application by a city building authority. Then, dimensional accuracy and preservation of colours are essential criteria for faithfulness. Faithfulness and the possibility to revise it afterwards must here be ensured through usage of devices and software of appropriate quality, suitable target data formats (JPEG, e.g., is known not to be adequate for line drawings due to artifacts), and a proper choice of settings. In a secure e-government workflow, the requirements on the target with respect to cryptographic security and signature level might be low, and an automated signature applied by the transformation device can suffice. Classification can be carried out by a human inspector who checks, e.g., a paper drawing for qualitative defects, or in a highly automated way by checking that an XML document satisfies the syntactic rules of a certain schema.

The classification also determines if the source is at all appropriate for the desired purpose. An important example is the mentioned presentation problem of signed digital data, which can raise requirements for presentation components in the transformation system, set limits on automation, or even completely prohibit a secure transformation. Even for signed paper documents the proper handling of marginal notes, cancellations, and corrections of text must be completely and clearly prescribed. On a higher level, (legal) formalities that have to be satisfied by the source, for instance the presence of a certain number of signatures in the right places, can be criteria to decide if it is admissible for transformation.

Before delving into the transformation process proper, two basic data structures must be introduced.

The **rule-set** is, along with the classification data describing the source and the transformation purpose, a central result of the classification. It is an abstract term for the comprehensive set of rules governing all the following phases. Yet, it has a very concrete meaning in every special application case, comprising organisational provisions, technical measures, attributions of responsibility, rules for signature verification and generation, etc. Generically, the rule-set consists of a combination of machine-processable instructions, with normative prescriptions understandable only by humans. In some cases, like notarisations, the latter can already be implicated by existing legal regulations. Since application scenarios with transformations for special purposes abound, it is pragmatic to define generic rule-sets for particular domains and adapt them through *profiles* which reference the generic rules and specialise them appropriately. It is desirable that the generic rule-sets be extensible, modular, combinable, and parameterisable.

As a data container which carries the information compiled during the transformation, the **transformation record** is useful not only for purely technical reasons, but also to establish security by conserving relevant meta data and, e.g., protocols of the conversions and inspections carried out in later phases. The first item in the transformation record is the rule-set.

The transformation record serves also to ensure the proper binding of relevant data with each other. In particular: 1. The source contents must be uniquely identified throughout the whole process, for which the record carries an identifier. 2. Likewise, the rule-set must be unique during the process. 3. The integrity of the target's contents and their association with the source's must be ensured. 4. Protocols and meta data generated must be kept unique for the process and unadulterated.

While in closed transformation systems the record may be a simple data container storing the objects mentioned, distributed processing or genuine security requirements can necessitate that portions of it are cryptographically secured to maintain the mentioned bindings and data integrity.

4.2 Conversion

Signature Extraction. During signature extraction, the signatures of the source are gathered from it and added to the transformation record as source signatures. The rule-set determines whether digital signatures must be verified or the signers of handwritten ones be authenticated. In this case, it also prescribes validation policies and names the signature data to be carried to the record (e.g., time stamps, attributes, etc.), and the result of the validation is also added to the record.

Conversion. In this phase the proper conversion of source to target contents takes place according to the rules of the rule-set. Apart from the target contents, a conversion protocol and error log is filed in the record.

4.3 Assay

Conversion Assay. In many, but not all cases it is possible to include two steps of *ex post* inspection into the transformation process to raise the level of trustworthiness. To indicate that these steps can comprise a mixture of human inspection of the converted contents, automated comparisons with the target, consistency checks on the data of the transformation record, we use the not very common word 'assay' for them[9].

The first step assays the results of the conversion of the contents by any means possible, and as prescribed by the rule-set. As mentioned, this can mean anything from a person comparing source document and converted contents using a trusted viewing component, to merely checking the syntactic compliance of the converted contents with a specific data format (e.g., an XML Schema). Similar checks, if they have not already been implicitly applied during signature extraction, can take place for signature data. Most importantly, the source can at this point be discarded from the transformation process if this is allowed and the conversion assay leads to a positive result — both criteria being specified, again, by the rule-set.

Transformation Assay. A final assaying step can inspect the correctness of the whole transformation process. For instance in distributed transformation systems, it can be necessary to ascertain that all necessary phases have been traversed, or to counter-check the hash values associated with certain parts of the transformation record.

4.4 Sealing

After the two assaying steps have obtained a positive result, the transformation record is complete and the transformation as such is ended. It remains the task of securing these results to achieve the ultimate goal of a secure transformation. For this, a **transformation seal** is attached to the transformed document and signed by the transforming entity.

The result of a transformation needs to be secure even if the source is not available for later comparison. This is the main reason why relevant data produced in the transformation process must be persistently and trustworthy bound to the target to enable a thorough forensic inspection. This possibility to assess the quality of a transformation *a posteriori* is an important building block for the probative force of the target. It is embodied in three subordinate goals, which describe the essential purpose of the transformation seal. 1. Securing the integrity of the transformed document, and other recorded data. 2. Attestation of the correctness of the transformation according to the specified rule set. 3. Attribution of the transformation to the transforming entity and non-repudiation of that fact.

[9] This is also to distinguish it from the notion of verification which is often understood as being specifically bound to digital signatures.

In general, the rule set will contain instructions on which parts of the transformation record need to be transferred to the transformation seal. The seal attests that the transformation process was carried out correctly according to the rule set and that the desired faithfulness between source and target is thus achieved. Technically, the transformation seal can be realised as a cryptographically secured data container and selected data from the transformation record and other relevant meta data. It always has to be (digitally) signed by the entity or person that performed the transformation.

In particular in the assaying and sealing steps specific need for human inspection can arise for technical, security, and legal reasons. A regular inspection of the transformation system and probing of results is perhaps a standard organisational requirement for secure transformation systems. If the content of the original is not structured, the transformation itself must in part be carried out by hand and consequently the result should be (independently) inspected by a human. Legal responsibilities borne by the person sealing the target may, as is likely to be the case for notaries, entail the necessity of inspecting the target. Needless to say, human interference always introduces its own risks into technical processes. Since it is unavoidable in general though, technology must support secure and failsafe means for it, e.g., trusted viewing components in view of the mentioned presentation problem, and trusted signature terminals. The remaining risk of malicious behaviour is, however, already covered by civil and criminal legislation, for instance with regard to the liability in case of negligent or fraudulent use of notary or official seals.

5 Case Study: Secure Translations

With the structure of the transformation processes at hand, we present a final ***example*** to exhibit their scope, which by no means is restricted to conversions between data formats. It also sheds some light on the limits of automation of legally secure doument transformations. Authorised translations[10] are essential for transnational document exchange. Let us, as an abstract exercise, describe an authorised translation as a document transformation — classically between paper documents. The translator classifies the source by checking it for illegibility or other severe defects that would forbid performing a translation, and also inspects the contents to avoid becoming involved into obviously illegal proceedings. After conversion of the contents by translating it to the target language, she attaches her seal to source *and* target in a way which makes them inseparable as items of probative force — by stapling the source with the target and stamping the seal over the staple. She finally applies her written signature to the target to authenticate the target and the seal. The purports of source and target

[10] Under German law, an authorised translation is performed by a professional translator, sworn, registered with a certain judicial circuit, and equipped with a special seal for the purpose of translations. The special prerequisites for authorised translators vary within Germany between federal states. In most states a translator is required to sit a state exam before being able to apply for authorisation.

are not bound to special area, and therefore the purposefulness of the translation depend on the competence of the translator and is subject to human error. Thus, a secure transformation can only be achieved through the organisational requirement that the translator possesses a certified credential (the seal). The seal ensures trustworthiness in two senses. It attests the capacity of the translator to convert the document contents faithfully, and ensures the possibility to forensically compare source and target.

Interestingly enough, a counterpart for authorised translations is completely missing for electronic, digitally signed documents, though it would be very useful. We offer some thoughts on it at a high level, based on the concepts developed, and following the process structure worked out above.

The classification of the source in the case of paper documents is rather simple for the translator who essentially checks if the document is written completely in the proper source language and is free from qualitative defects which would bar him from reading and translating it. The latter is a bit more complicated for electronic documents since the presentation problem has to be taken into account. The translator must be sure to be shown all signed contents. This leads to the first organisational requirement of the rule set, namely that the translator possesses a trusted viewer for the source signed document format This implicitly entails that he has access to the PKI in the source country that was used to create the source's signatures, which can be utilised in the next step.

Signature extraction for E→E translations clearly offers more possibilities and variants than for paper documents with handwritten signatures, where signer authentication hardly ever takes place and targets mostly carry a note 'illegible signature' in the approximate place of the original ones. In contrast to that, the translator can verify the digital signatures in the source and carry that verification data into the target in some form. Depending on the level of PKI interoperability between the two countries in question, the translator could either — if the two respective CA domains are not connected — serve as an independent authentication instance for signatures from the source country and attest their validity through his transformation seal. For this, bilateral accords and a special authorisation of the translator from the target country (where the signatures are to be accepted) would be necessary. Or, in the more preferable case where the CAs are bridged by a transnational infrastructure like that envisaged in the European Bridge CA project [8], source signature verification can be directly, and without special organisational prerequisites, be carried out by the translator as well as by the target's recipient (i.e., where the document is to be utilised). In both cases however, it makes good sense to carry the original signatures completely to the target for forensic inspection. If the source consists of signed and unsigned data and/or carries more than one signature over different portions of it, the pertinent associations must be recorded.

Clearly automation of content conversion will not be possible in the foreseeable future, and can only be carried out by a responsible human being. This simplifies the transformation process. In particular, conversion and transforma-

tion assay become implicit steps carried out by the translator while translating the contents.

Three simple rules govern the sealing of the target. 1. The certificate of the translator must be issued by an appropriate authority of the target country and identify him as an authorised translator. 2. The translators signature authenticates at least source and target contents, and if desired also the signatures of the source, to enable forensic inspection. 3. If there is a many to many association between source signature(s) and portions of signed data, this must be re-traceable in the target by introducing appropriate meta-data structures.

Requirement 1. is in fact paradigmatic for the transformation seal. In analogy to paper documents which are signed *and* sealed, two authentication characteristics will generally be required for a legally secure seal. An electronic signature identifying the transforming person (or entity, where such is admissible), and a means to authenticate his/her role as a person authorised to carry out the transformation. Technically, a solution through the use of attribute certificates in transformation seals can be envisaged. Since the issuance of 'seals' to notaries, authorised translators, public officials, etc., is governed by detailed legal regulations, and organised in highly heterogeneous and de-centralised administrative infrastructures, the actual implementation of this sound technological solution approach still poses a non-trivial organisational problem. Questions to be resolved include decisions on the carrier and operator of the certificate infrastructure, cost-sharing between administrations, guaranteed service availability, regulation and of certificate revocation[11]

6 Conclusions

Assuring legal security of document transformations is a demanding task necessitating an interdisciplinary approach. Such an approach must combine organisational measures with technical solutions to meet the legal requirements pertaining to a concrete application case. In some cases it may be doubtful whether legal existing regulations, e.g., those of EU member states that realise the Digital Signature Directive and subsequent regulatory statutes, suffice to achieve the necessary security for transformed documents. Concrete solutions for a broad application spectrum should be devised on the basis of the present concepts. These comprise organisational guidelines, process and technical prescriptions, as well as generic software components which perform the transformation process, and realisation of the transformation seal. This work programme is at the core of the project TransiDoc [3].

A point for current research is the instantiation of rule-sets. The general concepts presented above cover a wide range of transformations, too wide to be covered by uniform rules which do not remain on the level of commonplaces. The central idea here is to come to concretely usable rule-sets by profiling along the

[11] In the German Signaturgesetz, the relevant regulations governing issuance and revocation of, and authority over attribute certificates are stipulated in §§5, 7, and 8, respectively.

two axes of application and legislative domain. The latter regards the projection of organisational and technical guidelines to national legislations, whereas the former concerns those rules which are determined by the purpose of a transformation and entail, e.g., machine-processable rules for transformation systems. These rules delineate the boundaries of the notion of transformation, for instance the classification P→E, E→E, and E→P, replacement, partial copy, and valorisation, and more specifically changes of data format, formatting, and layout. The two axes are clearly not orthogonal. An intermediate aim is to create a methodology and standard data structures for profile creation and recording.

Besides the problem of long-term conservation of digitally signed data, which is addressed, for instance by LTANS, see also [9,10,11,12], legally secure transformations may be the most pressing issue for business and legal relations based on signed electronic documents.

Acknowledgements

The parts of this report authored by A. U. Schmidt are results of the project TransiDoc — Legally Secure Transformations of Signed Documents, funded by the German Ministry of Economy and Labour under contract no. 01 MS 401. Full responsibility for the contents of this report resides with the authors. A. U. Schmidt wishes to thank Stefanie Fischer–Dieskau, Thomas Kunz, and Ursula Viebeg for discussions.

References

1. The UNCITRAL Model Law on Electronic Commerce. www.uncitral.org/english/texts/electcom/ml-ecomm.html.
2. Guide to Enactment of the UNCITRAL Model Law on Electronic Commerce. www.uncitral.org/english/texts/electcom/ml-elecsig-e.pdf
3. Website of the TransiDoc project. www.transidoc.de
4. Long-Term Archiving and Notary Services Working Group of the IETF. ietf.org/html.charters/ltans-charter
5. Wallace, C., and Chokhani, S., 2003. Trusted archive protocol (TAP) IETF Internet draft. www.ietf.org/internet-drafts/draft-ietf-pkix-tap-00.txt.
6. Landrock, P., Pedersen, T.: WYSIWYS? What you see is what you sign? Information Security Technical Report, **3** (1998) 55–61
7. Schmidt, A. U.: Signiertes XML und das Präsentationsproblem, Datenschutz und Datensicherheit **24** (2000) 153–158
8. Website of the European Bridge CA project. www.bridge-ca.de
9. Ansper, A., Buldas, A., Roos, M., Willemson, J.: Efficient long-term validation of digital signatures. In: Proceedings of 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2001), Korea; 2001, 402–415.
10. Dumortier, J., van den Eynde, S.: Electronic signatures and trusted archival services. In: Proceedings of the DLMForum 2002, Barcelona 6–8 May 2002, Luxembourg, Office for Official Publications of the European Communities, 2002, 520–524.
11. Lekkas, D., Gritzalis, D.: Cumulative notarization for long-term preservation of digital signatures. Computers & Security **23** (2004) 413–424
12. Website of the ArchiSig project. www.archisig.de

Author Index