NISTIR 7224
ISBN 1-886843-38-4

4th Annual PKI R&D Workshop "Multiple Paths to Trust" Proceedings

Clifford Neuman
Nelson E. Hastings
William T. Polk

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7224
ISBN 1-886843-38-4

4th Annual PKI R&D Workshop "Multiple Paths to Trust" Proceedings

Clifford Neuman
University of Southern California

Nelson E. Hastings
William T. Polk
Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology

August 2005

U.S. DEPARTMENT OF COMMERCE
Carlos M. Gutierrez, Secretary
TECHNOLOGY ADMINISTRATION
Michelle O'Neill, Acting Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
William A. Jeffrey, Director

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

Foreward

NIST hosted the fourth annual Public Key Infrastructure (PKI) R&D Workshop on April 19-21, 2005. The two and a half day event brought together PKI experts from academia, industry, and government to explore the current state of public key technology and emerging trust mechanisms, share lessons learned, and discuss complementary topics such as usability. The workshop also served as a forum to review continuing progress in focus areas from previous workshops. In addition to the seventeen refereed papers, this proceedings captures the essence of the workshop activities including the invited talk, three panels, and the work-in-progress session.

The third annual workshop concentrated on public key authentication and authorization technologies; previous workshops focused on PKI theory, architectures, human factors, and deployment. The fourth workshop focused on interactions between PKIs and emerging trust mechanisms, but also addressed security of cryptographic primitives, usability of PKI-enabled applications, standards development, and sector specific deployment issues.

The workshop opened with sessions on the integration of the Shibboleth system with the Global Grid architecture and interoperability of traditional PKI with Shibboleth and other federated models. Sessions on enhancements to traditional PKI architectures and a look at PKI in government completed Day 1.

Day 2 began with Arjen Lenstra's invited talk, *Progress in Hashing Cryptanalysis*, which demonstrated the relevance of recent cryptanalytic results to PKI implementations. By applying recent advances in the cryptanalysis of SHA-1 by Xiaoyun Wang, Lenstra and de Weger have demonstrated that the X.509 certificate structure could be manipulated to create certificates with different keys that are consistent with a single digital signature. Lenstra challenged the traditional Merkle-Damgard construction, suggesting that a stronger basis for secure hash functions is needed.

After the invited talk, the workshop revisited a topic from the second workshop with a session on Usability and PKI. A session on standards activities addressed both X.509 based PKI standards and new standards initiatives based on the Weil pairing – a topic featured in the initial workshop. A session on the scalability and performance of PKI systems and a series of informal presentations of works in progress completed the day.

The workshop's final day began with a look at future PKI development and ended with a discussion of current PKI capabilities. Presentations on language based policy analysis and security mediated PKI described how these techniques may resolve problems in current PKIs. The final session was a panel presentation on PKI and the health care industry describing how current technology is supporting security services for mission critical applications.

The 120 attendees represented a cross-section of the global PKI community, with presenters from the United Kingdom, Canada, Brazil, Germany, and Japan. Due to the success of this event, a fifth workshop is planned for the spring 2006.

William T. Polk and Nelson E. Hastings
National Institute of Standards and Technology
Gaithersburg, MD USA

This page has been left intentionally blank.

2005 PKI R&D Workshop
Multiple Paths to Trust

Gaithersburg, Maryland USA
April 19-21, 2005
http://middleware.internet2.edu/pki05/

(Pre-proceedings were distributed at the workshop)

* Paper was accepted, but was not presented at during the Workshop.

This page has been left intentionally blank.

4th Annual PKI R&D Workshop Summary

Ben Chinowsky, *Internet2*

Note: This summary is organized topically rather than chronologically. See http://middleware.internet2.edu/pki05/proceedings/ for the workshop program, with links to papers and presentations.

The overarching theme of this fourth iteration of the PKI R&D Workshop was *putting PKI to use*. There was much experience to report from new and ongoing PKI deployments.

Deployments

One of the largest PKI projects is the ongoing **Department of Defense deployment;** Rebecca Nielsen brought the group up to date on DoD's experiences. Nielsen noted that PKI cycle times are around 72 months, while hardware cycle times are more like 24 months, so you end up running on obsolete hardware much of the time. Also, while "smartcards are good," reissuance is hard when you have 3.5 million users. Organizational issues loom larger than technical ones; getting buy-in from both users and management is particularly challenging. Secure web server access is the primary application, though signing and encrypting email are also done; user education is a particular challenge with the latter ("this PKI thing is a problem because I can't sign my boss's emails anymore.") Nielsen characterized the DoD rollout as "generally successful."

Rajashekar Kailar presented experiences with **Securing the Public Health Information Network Messaging System (PHINMS),** deployed by CDC and its partners. PHINMS uses certificates for SSL but does not require a specific source, and Kailar noted that other means of secure authenticated transport could also be acceptable. Kailar identified this as a key lesson learned: where multiple credentials and mechanisms for security are acceptable, permit them, rather than trying to impose just one.

The PHINMS presentation was followed by a panel on **PKI In Healthcare.** Richard Guida presented an overview of the sector, noting that it can be divided up into four categories: companies that supply medical devices, equipment and pharmaceuticals; "points of care" (e.g. hospitals and clinics); companies that handle payments and billings (e.g. insurers); and companies or institutions that support or perform medical research, including clinical trials. Guida noted that while the strong focus on patient care at the points of care tends to lead to less of a focus on security, there is lots of growth in this sector, and lots more the research community and vendor community can do to help ensure that privacy requirements set forth under the Health Insurance Portability and Accountability Act (HIPAA) are met. Reducing the burden of paperwork, especially the burden of storing paper and moving paper around, is a particularly great need; reducing the paperwork involved in clinical trials can save time without reducing the quality of the trials. Guida cited Acrobat 6 and 7 as "exemplars" of PKI growing up. Terry Zagar discussed the biopharmaceutical industry's Secure Access For Everyone initiative (SAFE; http://www.safe-biopharma.org/). Zagar noted that PKI is much harder to scale between enterprises than within them, hence the need for common standards such as those being developed by SAFE. Many companies already have PKIs and want them to interoperate so that certificates can be accepted by outside parties; minimizing the need for reinvention is a major consideration. SAFE has embraced the use of a bridge CA as

a means to this end. SAFE plans to have tens of thousands of certificates issued to doctors and other healthcare professionals by the end of the year, and hundreds of thousands by next year. John Landwehr offered details on the Acrobat signing technology cited by Guida and intended for use by SAFE to execute and validate legally binding digital signatures that comply with regulatory requirements (FDA 21 CFR Part 11). Documents let you apply signatures inline — applying the principle of making it as much like paper as possible — and can specify parameters for allowed signatures. The technology has passed the 250+ tests in the NIST PKI test suite and is JITC certified; a FIPS 201 evaluation is underway.

In the discussion, Ken Klingenstein pointed out that the communities that seem to have the most traction in implementing role-based access control are those that have regulatory mandates that lead to common definitions of roles — above all the securities industry. The panelists agreed on the need for RBAC in healthcare, though this is a ways off yet. Guida noted that there are recurring problems with large institutions failing to understand rapid technical changes and accompanying opportunities; having real-world success stories to tell helps a lot in this regard. Landwehr noted that the standardization of smartcards has the potential to make cert deployment a lot easier, and Zagar stressed the importance of avoiding US-centric standards.

Mike Just discussed **Canada's Secure Delivery of E-Government Services,** updating the group on work presented at PKI03. EPass is now a successfully established solution; the current issues are political and legal, e.g. privacy concerns leading to multiple and burdensome enrollments. Just also noted that Canadian law has recently changed to make electronic data acceptable as legal evidence.

A **Works in Progress** (WIP) session by Jeroen Van de Graaf provided an overview of **PKI Projects in Brazil.** As in the US, projects are underway at the national government and pan-Higher-Education levels. There is also a large project in the state of Minas Gerais, driven by the need to fix the presently fraud-prone process of publishing legal judgments in newspapers. The long-term goal is to issue smartcards and certs for all 15 million residents of the state. At all levels, there is a strong bias in favor of open-source solutions, for both financial and strategic reasons. Van de Graaf also noted that a 2001 federal directive gave digital signatures the same legal status as wet signatures.

There were also two sessions on bridge deployments and PKI interoperability. Peter Alterman surveyed **International and Bridge-to-Bridge Interoperability,** including pending cross-certifications between FBCA and Canada, FBCA and Australia, and the DoD PKI and trusted allies. Alterman noted that where PKI-PKI cross-certification is concerned primarily with policies, bridge-bridge cross-certification requires that business practices be commensurate as well. Scott Rea followed with an update on **HEBCA and Phase 4 of the PKI Interoperability Project.** HEBCA grew out of the NIH-EDUCAUSE PKI interoperability pilot, and has since been moved to Dartmouth; production FBCA/HEBCA cross-certification is expected in a few months. Form-signing is the principal application.

A perspective on **Side-Effects of Cross-Certification** was provided by James Fisher: "It is easy to structure unintended and difficult-to-detect consequences." This assertion is amply documented in Fisher's paper and slides. In the Q&A for this session, Fisher noted that the technical aspect of bridging is relatively straightforward; it's getting the

trust path to reflect what was agreed on in human-to-human trust negotiations that introduces most of the complexity.

The deployment-experience portion of the workshop was rounded out by Ken Klingenstein's survey of **Interfederation Interoperability, E-Authentication and Shibboleth.** There are now production federations in several countries, and many campuses and other enterprises which are themselves federal in structure have been creating federations for themselves. Indemnification issues are the biggest obstacle in getting universities to participate in federations. Ken noted that SAML 2.0, ratified by OASIS earlier this year, is likely to prove a high plateau; it's "a fusion product" with Liberty Alliance, which has incorporated most of its functionality into SAML and is moving off in new directions.

Addressing recurring deployment issues

Of the presentations not concerned with specific deployments, many considered developments aimed at solving problems that recur across deployments. Note that in addition to the presentations discussed below, the proceedings include papers in this area from two contributors — Tice DeYoung and Karl Scheibelhofer — who ended up not being able to attend the workshop.

One of the most important areas of cross-deployment development is, of course, **standards.** With respect to IETF, PKIX co-chair Tim Polk noted that no longer is everything happening in PKIX; there are also important developments in PKI4IPSEC and LTANS. Internationalizing domain names is a major focus, and a new version of RFC 3280 with expanded support for this is expected later this year. Overall, Polk characterized the core PKI specs as stable and the supplemental specs as ready for implementation, so standards obstacles to PKI deployment are diminishing. IETF Security Area Director Russ Housley noted the creation of the ENROLL working group; it's likely that considerable research will be needed to create an effective standard in this area.

Eric Norman asked if the IETF is planning to issue standards for digital signatures, given that the courts are likely to decide what's acceptable here. There was general agreement that the technical community — primarily in IETF, but also in government bodies like NARA — still needs to take the lead in guiding implementations. Housley said that social-engineering attacks based on flaws in Unicode are likely to remain a problem for some time; several working groups are studying the complex tradeoffs in this area.

Polk also surveyed the FIPS 201 project at NIST. This is a Presidentially-mandated standard for both physical and electronic access to Federal facilities; public-key cryptography and contactless smartcards are the core technologies. FIPS 201 was published in February of this year. Biometrics introduce new vulnerabilities and can compromise privacy; fingerprint images are big and therefore slow to move on and off cards. Cards were chosen over other hardware such as dongles largely because they can function as old-fashioned photo IDs as well. See http://csrc.nist.gov/piv-project/ for more on FIPS 201.

David Engberg of CoreStreet presented work on **Secure Access Control with Government Contactless Cards,** for FIPS 201 in particular. Engberg noted a prosaic

reason for contactless cards: the contact strips on swipe cards tend to wear out after a few months. On the other hand, there are privacy risks in allowing remote access to contactless cards. Engberg also noted that processing-power limitations on PKI are "starting to melt away."

Jon Callas discussed a hybrid approach to **IBE with conventional PKI.** IBE as first proposed by Shamir requires a master secret on a central server, creating Kerberos-like vulnerabilities. Callas's approach addresses this by removing offline key generation; this system has also been referred to as "attribute-based enrollment". Callas noted that when you have ubiquitous devices that are hard to turn off — as is increasingly the case just about everywhere — the advantages of offline operations are minimal anyway. Callas argued that his hybrid approach can bring the advantages of IBE to existing PKIs.

Marco "Kiko" Carnut presented an IBE-like approach to **Taking Cryptography Out of the Browsers.** This is accomplished by a proxy, called Kapanga, that takes over functions like certificate issuing and webform signing that are often handled badly by browsers. Carnut described his approach as similar to that of Callas's "Self-Assembling PKI" as presented at PKI03: make every application an opportunity for enrollment. Carnut further elaborated his ideas is a WIP session, offering **an IBE-like idea for instantaneous enrollment.** In this approach, certs are issued with no authentication, and trust depends on the client CA instead of the root.

Sang Seok Lim presented a method of improving access to cert repositories via **LDAP component matching.** He noted that while component matching is generally considered to be the approach of choice, it's complex; his work demonstrates that the complexity can be limited enough to ensure deployability.

There were two presentations on delegation of authority. David Chadwick described a **Delegation Issuing Service for X.509.** Advantages of this approach include the managers doing the delegating not needing to have certificates themselves. Chadwick noted that the lack of standard terminology for roles is a big obstacle to any delegation scheme, including this one. Liang Fang presented **XPOLA,** which stands for eXtensible Principle Of Least Authority. XPOLA is motivated by the need to reduce the time needed to get access to Grid services and by the need to improve authorization scalability and fine-grainedness.

Kenji Imamoto offered **One-time ID** as a solution to DoS attacks on the SEM approach to revocation. One-time ID makes use of symmetric key authentication to provide low overhead.

Two talks explored proposed extensions to the Shibboleth federating software. David Chadwick discussed **Adding Distributed Trust Management to Shibboleth** by combining it with PERMIS (PrivilEge and Role Management Infrastructure Standards; see http://www.permis.org/). Chadwick's paper explores several different ways to combine the two. Von Welch described **Integrating Shibboleth and Globus.** The motivation for this work is to get virtual organizations of scientific researchers out of the business of IT support. Integration is based on replacing Shibboleth's handle-based authentication with X.509, offering stronger security while leveraging the X.509 installed base. Working code is expected this summer; see http://grid.ncsa.uiuc.edu/GridShib/.

In a WIP session, Carl Ellison discussed the need for **ceremony analysis** — formal analysis of the human-to-human, out-of-band elements of security processes. Ellison argued that these elements are just as much part of security protocols as are operations that take place inside computers, and need to be taken seriously as such.

BoF on Human-Computer Interaction

More generally, it is widely agreed that **human-computer interaction (HCI)** is one of the areas where much work is still needed if PKI deployments are to thrive. HCI was the main focus at PKI03; an HCI BoF at PKI05 reviewed recent developments in this area. BoF chair Eric Norman has been trying to identify the minimum set of things a PKI user should need to learn, and used a draft list to get discussion started. The consensus from his previous discussion on the HCISec list (see http://groups.yahoo.com/group/hcisec/) is that this list needs to be much, much shorter. The BoF participants concurred in this judgment; Paul Danik asked "How do you teach someone with rudimentary computer skills even one of the things on Eric's list?"

The group discussed Simson Garfinkel's work on HCI (see http://simson.net/). Sean Smith was a guinea pig for Garfinkel's prototype, which he found confusing — "why are these things changing colors?" — although as several people pointed out, Smith might not be the best test subject for a system aimed at the naive user. There was general agreement that it's well worth paying attention to Garfinkel's criticisms of existing PKI user interfaces.

Smith noted that it's only in the last couple of years that phishing and IDN attacks have created broad awareness that spoofing is really a problem, and recommended taking a look at the presentations from the DIMACS Workshop on Theft in E-Commerce: http://dimacs.rutgers.edu/Workshops/Intellectual/slides/slides.html. He also pointed the group to anti-spoofing work presented at Usenix: http://www.cs.dartmouth.edu/~sws/abstracts/ys02.shtml.

The BoF participants discussed several other factors involved in making PKI usable. Carl Ellison stressed the importance of giving the relying party control over what name is used for a trusted entity, as all other information the user learns about that entity is linked to the name; this is an entrenched human behavior that no amount of "user education" can or should try to change. Jon Callas observed that "one of the principles of real human factors is, the user is always right." Callas also related an experience with certs expiring mid-transaction; there was general agreement that "about to expire" warnings are needed. Several of those present spoke well of *The Design of Everyday Things* by Don Norman (no relation to Eric); though the book is more about doors and clock radios than computers, its principles apply to making anything more usable.

Farther out

There were also several talks aimed at solving broader problems with PKI, or at applying it in new ways. Of these, the one with the widest implications was Arjen Lenstra's discussion of **Progress in Hashing Cryptanalysis.** Lenstra discussed the implications of recent discoveries of weaknesses in commonly-used hash functions; his slides offer an overview of the mathematics involved, and of how these weaknesses might be exploited in real-world attacks. This February NIST announced that SHA1 should still be considered secure until it's phased out around 2010. Lenstra's assessment is somewhat

more pessimistic; while there are currently no dangerous attacks based on these recent discoveries, research continues, and such attacks are likely to emerge soon. Lenstra suggests abandoning SHA1 for SHA256 and launching a competition for a replacement for the entire SHA family. On the other hand, Bill Burr noted that encouraging a move to SHA256 in the short term could make it a lot harder to move to the hoped-for SHA replacement in the medium term. NIST doesn't have the resources to develop that replacement. Burr agreed that a global competition is the best way to mobilize the resources needed.

Cliff Neuman presented a WIP session on work by his student Ho Chung on a multidimensional approach to **Modeling Strength of Security.** This work is at an early stage.

A. Prasad Sistla described a scheme for **Language-Based Policy Analysis in a SPKI Trust Management System,** using modal logic to describe roles in SPKI. While citing related work, Sistla claims that this is the first general policy-analysis language, usable e.g. with Keynote or XACML. There is no implementation yet.

Terence Spies discussed **Pairing Standards for Identity Based Encryption.** "Pairings" are a new elliptic-curve crypto primitive. An IEEE study group on pairings and their application to IBE is just getting underway; see http://grouper.ieee.org/groups/1363/WorkingGroup/.

Finally, Meiyuan Zhao presented her simulations aimed at **Evaluating the Performance Impact of PKI on BGP Security.** Russ Housley stressed the importance of this work; he noted that securing BGP is one of his top priorities as an IETF Security Area Director. Housley also observed that memory requirements are a major obstacle to S-BGP deployment, and suggested focusing future research on approaches that require less memory, in particular by using elliptic-curve cryptography.

Conclusions

The PKI0x series has clearly matured, as demonstrated by its emulation in Europe and Asia (see "Related Workshops" at http://middleware.internet2.edu/pki05/), by this year's conference having the largest number of accepted papers yet, and by several of the sessions offering follow-ups on ongoing work presented in previous years.

PKI04 produced consensus on two main ideas: "Understanding and educating users is centrally important" and "The specifics of any particular PKI deployment should be driven by real needs, and should be only as heavyweight as necessary." PKI05 reaffirmed this consensus; it also demonstrated that we are much further along in applying the latter principle than the former.

There was strong general agreement on keeping the workshop's mix of research and deployment topics. Please join us for PKI06 (http://middleware.internet2.edu/pki06/), April 4-6, 2006.

Adding Distributed Trust Management to Shibboleth

David Chadwick, Sassa Otenko, Wensheng Xu
University of Kent, Computing Laboratory, Canterbury, England, CT2 7NF

Abstract

This paper analyses the simplicity of the trust model adopted by the Shibboleth infrastructure and describes an enhanced distributed trust model and authorisation decision making capability that can be implemented by using X.509 attribute certificates and a Privilege Management Infrastructure such as PERMIS. Several different combinatorial approaches can be taken, depending upon the trust models adopted by the Shibboleth target and origin sites, and each of these are described. The paper also discusses whether user privacy, which is strongly protected by Shibboleth, is bound to be weakened by the use of X.509 attribute certificates rather than simple attributes, and concludes that this does not have to be the case.

1. Introduction

Shibboleth [1] is a distributed web resource access control system that allows federations to co-operate together to share web based resources. It has done this by defining a protocol for carrying authentication information and user attributes from a home site to a resource site. The resource site can then use the attributes to make access control decisions about the user. The Shibboleth project is run by the Internet2 consortium in the USA, and universities throughout the USA and Europe (at least) are now starting to build experimental services based upon it.

At the heart of Shibboleth is a trust model that allows the members of a federation to cooperate together. This trust model, whilst functional, is somewhat limited. Basically each Shibboleth target resource site trusts each Shibboleth origin (home) site in the federation, so that whatever assertions – authentication or authorisation – are digitally signed by the origin site, they will be believed and trusted by the target site. There is little scope for differentiation between authentication authorities and attribute authorities, or for allowing more sophisticated distribution of trust, such as static or dynamic delegation of authority.

Another limitation of the Shibboleth infrastructure is that it only provides a basic access control decision making capability. Whilst this is adequate for many use cases, it lacks the flexibility and sophistication needed by many applications, for example, to make access control decisions based on role hierarchies or various constraints such as the time of day or separation of duties.

We realised that both these limitations could be addressed by integrating an X.509 Attribute Certificate (AC) Privilege Management Infrastructure (PMI) [3] with Shibboleth. PERMIS [2] is one such infrastructure that has already been successfully integrated into Grid application target sites [4] to support the distributed management of trust. PERMIS incorporates a sophisticated policy controlled RBAC access control decision engine (also called a policy decision point (PDP)). The PERMIS PMI has been used to implement distributed trust management in Shibboleth.

The rest of this paper is structured as follows. Section 2 provides an overview of Shibboleth. Section 3 introduces the more sophisticated distributed trust model that we

wanted to introduce into Shibboleth. Section 4 describes how the trust model can be implemented using an X.509 PMI such as PERMIS. Section 5 describes the different combinations of X.509 ACs, attributes, and the PERMIS PDP that may be integrated with Shibboleth to provide the desired trust models of the Shibboleth target and origin sites. Section 6 discusses user privacy issues and section 7 discusses revocation and performance issues that arise with using X.509 ACs. Finally Section 8 concludes.

2. Overview of Shibboleth

Shibboleth is a web based middleware layer that currently makes use of SAMLv1.1 [5] for encoding some of its messages. When a user contacts a Shibboleth resource site from their browser, requesting access to a particular URL, Shibboleth single sign on and access control takes place in two stages:

- In stage one the resource site redirects the user to their home site, and obtains a handle for the user that is authenticated by the home site
- In stage two, the resource site returns the handle to the attribute authority of the home site and is returned a set of attributes of the user, upon which to make an access control decision.

In a large distributed open environment stage one has a number of complications. Firstly how does the resource site know where the user's home site is? Secondly, how can the resource site trust the handle that is returned? The answer to these two questions is surprisingly simple, and is part of the Shibboleth trust model. When the user first attempts to access a resource site, he/she is redirected to a Where Are You From? (WAYF) server, that simply asks the user to pick his/her home site from a list of known and trusted home (Shibboleth origin) sites. The target site already has a pre-established trust relationship with each home site, and trusts the home site to authenticate its users properly. This is facilitated by the exchange of public key certificates or the use of a common trusted root Certification Authority. In the latter case both sites will have been issued with a certificate by the root CA (or one of its subordinates). When a digitally signed SAML message[1] arrives from the home site, such as one containing a user handle, this can be validated and trusted by the resource site.

After the user has picked his/her home site, their browser is redirected to their site's authentication server and the user is invited to log in. If a user is untrustworthy and tries to fool the system by picking a home site to which they do not belong, they will have difficulty authenticating themselves to that site's authentication server, since they won't have any valid credentials. However, if they pick their own home site, they should find authentication is no problem. After successful authentication, the home site re-directs the user back to the resource site and the message carries a digitally signed SAML authentication assertion message from the home site, asserting that the user has been successfully authenticated by a particular means e.g. username/password, Kerberos or digital signature. The actual mechanism used is local to the home site, and the resource site simply has to have a prior agreement with the home site which authentication mechanism(s) will be trusted. If the digital signature on the SAML

[1] Note that the connection from the origin server to the target server can also be optionally protected by SSL in Shibboleth, but this is used to provide confidentiality of the connection rather than message origin authentication. In many cases a confidential SSL connection between the origin and the target will not be required, since the handle is obscure enough to stop an intruder from finding anything out about the user, whilst the SAML signature makes the message exchange authentic.

authentication assertion verifies OK, then the resource site has a trusted message providing it with a temporary pseudonym for the user (the handle), the location of the attribute authority at the origin site and the resource URL that the user was previously trying to access. The resource site then returns the handle to the home site's attribute authority in a SAML attribute query message and is returned a signed SAML attribute assertion message. The Shibboleth trust model is that the target site trusts the origin site to manage each user's attributes correctly, in whatever way it wishes. So the returned SAML attribute assertion message, digitally signed by the origin, provides proof to the target that the authenticated user does have these attributes. This message exchange should be protected by SSL if confidentiality/privacy of the returned attributes is required. The attributes in this assertion may then be used to authorise the user to access particular areas of the resource site, without the resource site ever being told the user's identity.

The latest version of the Shibboleth specification has introduced a performance improvement over the earlier versions, by optionally allowing stage one and stage two to be combined together, in that the initial digitally signed SAML message may optionally contain the user's attributes as well as the authentication assertion. It is expected that the Shibboleth software will be upgraded to this during 2005.

Shibboleth has two mechanisms to ensure user privacy. Firstly it allows a different pseudonym for the user's identity (the handle) to be returned each time, and secondly it requires that the attribute authorities provide some form of control over the release of user attributes to resource sites, which they term an attribute release policy. Both users and administrators should have some say over the contents of their attribute release policies. This is to minimise the loss of a user's privacy.

3. An Enhanced Trust Model for Shibboleth

As can be seen from the above overview of Shibboleth, its trust model is sound although rather limited. The model is that the target site trusts the origin site to authenticate its users and to manage their attributes correctly whilst the origin site trusts the target site to provide services to its users. The trust is conveyed using digitally signed SAML messages using target and origin server X.509 key pairs/certificates, configured into the Shibboleth software by their filenames. (Note that the private key files were held unencrypted in the Shibboleth software we were using, so this is a weakness in the implementation if not actually in the trust model.) As each site will typically only have one key pair per Shibboleth system, from the recipient's perspective, there is only a single point of trust per sending Shibboleth system. Although it is not difficult to configure multiple roots of trust into a Shibboleth target site – it is, in fact, a matter of updating one XML file only – the issue is one of being able to use a finer grained distributed trust model, and of being able to use multiple origin site authorities (and private keys) to issue and sign the authentication and attribute assertions.

In many origin sites a single back end LDAP server is the sole authoritative source for both authentication and attribute information. Typically Shibboleth sites implement stage one by issuing a Bind operation on their LDAP server, using the username and password provided by the user to the web login prompt. If the Bind succeeds, the user has been successfully authenticated against the password stored in the LDAP server. Stage two is implemented

by searching the LDAP server for the attributes stored in the user's entry, and filtering these against the Shibboleth origin's attribute release policy before returning them to the Shibboleth target site as signed SAML attribute assertions. One can see that in such an implementation, and as a consequence of the Shibboleth trust model, the Shibboleth target site has no choice but to make access control decisions based on these attributes, without knowing who actually issued them to the user, whether they are still valid or not, or whether they are even the correct attributes for the particular user, since the user's name is not provided to the target site for privacy reasons. The Shibboleth origin doesn't trust anyone to see the attributes except the trusted targets, but even they are not allowed to see the binding between the attributes and the owner's identity. (The two reasons given for this in the Shibboleth documentation are user privacy and legal requirements for universities to protect a student's privacy). The target site thus has no option but to indirectly trust the contents of the origin site's LDAP server or other attribute repository, since it trusts the origin site directly. One can further see that the origin site has to strongly protect the attributes in its (LDAP) repository, which means that it is probably restricted to centrally administering these, and so would prefer that they do not change that often. Flexibility and distributed management of the attributes is hard to adopt. Dynamic delegation of authority would be even harder to support.

We propose that an enhanced trust model should have the following features.

- Multiple authorities should be able to issue attributes to the users, and the target site should be able to verify the issuer/user bindings. For example, a manager should be able to assign a project leader attribute to an employee under his control.
- The target should be able to state, in its policy, which of the attribute authorities it trusts to issue which attributes to which groups of users. The target site should be able to decide independently of the issuing site which attributes and authorities to trust when making its access control decisions.
- Not all attribute issuing authorities need be part of the origin site. A target site should be able to allow a user to gain access to its resources if it has attributes issued by multiple authorities, for example, a target site holding statistics on medical data may require a user to have an attribute issued by a medical authority as well as one issued by the university that employs the user as a researcher.
- The trust infrastructure should support dynamic delegation of authority, so that a holder of a privilege attribute may delegate (a subset of) this to another person without having to reconfigure anything in the system. For example, a project leader may wish to assign a role of team leader to one of his team members; he should be enabled to do this dynamically by the infrastructure without having to reconfigure the system. The target site should be able, in turn, to state in its policy whether it trusts these delegated attributes or not, regardless of the delegation policy at the user's site.
- The target site should be able to decide if it really does trust the origin's attribute repository (e.g. LDAP server), and if not, be able to demand a stronger proof of attribute entitlement than that conferred by a SAML signature from the sending Web server.
- Finally, the origin site, if it chooses, should be able to use a Privilege

Management Infrastructure, rather than a strongly protected attribute repository, for allocating attributes to its users. This will allow the origin to distribute the management of attributes throughout its site. Nevertheless, the origin site should still be able to communicate with Shibboleth targets as usual by only sending attributes to them, if the targets are happy to trust these.

4. Implementing the Enhanced Trust Model using an X.509 PMI

X.509 attribute certificates (ACs) provide a convenient, standardised and compact representation of attribute assignments, and satisfy several of the above requirements. The basic X.509 attribute certificate construct comprises: the name of the holder of the attributes, the name of the issuing authority, the set of attributes, and the time that they are valid for. An extension field can optionally be inserted to state that the holder is allowed to dynamically delegate (a subset of) these attributes to another user, and the depth to which delegation can take place. The whole AC construct is digitally signed by the issuer (attribute authority), thus providing integrity control and tamper resistance. Multiple attribute authorities can co-exist, either in a hierarchical relationship or as separate independent authorities. Attribute certificates are typically long lived, and after issuance, the ACs need to be stored somewhere for retrieval by the target's policy decision point (PDP), and LDAP repositories at the AA site are a natural choice for this, although web servers, filestores and other repositories can also be used.

If the ACs are stored in the AA site's LDAP directory or other repository, and transferred from there to the target site's PDP by Shibboleth, then the target site's PDP does not need to indirectly trust the attribute repository or the underlying transport mechanism used to convey them, since it can directly validate the digital signatures on the attribute certificates when it receives them[2]. Furthermore, if the target site's PDP policy is willing to allow dynamic delegation of authority, the PDP can check the attribute certificate chain to ensure that all ACs were properly authorised by their issuing authorities. By using ACs in its authorisation decision making, rather than plain attributes, a target site can support much more sophisticated and finer grained access control policies, for example, by requiring a user to have ACs issued by multiple authorities, from different issuing domains, before they are granted access to particular resources.

The PERMIS X.509 PMI is part of the US NSF Middleware Initiative software release. PERMIS provides a policy controlled role based access control (RBAC) infrastructure, in which the user's roles are stored in X.509 ACs. These ACs are either passed to the PERMIS PDP along with the user's requested action (the push model), or can be fetched from one or more LDAP servers by the PDP (the pull model). The PERMIS PDP then returns a granted or denied response according to the policy in force at that time. The PERMIS policy is written in XML, and is in many respects a simplified alternative to XACML [6], although the PERMIS policy supports dynamic delegation of authority, unlike XACML. The XML policy is itself stored in an X.509 attribute certificate, and is digitally signed by the trusted authority in control of a target resource. This policy certificate is the root of

[2] It is the case with ACs that the holder's identity is revealed in the Holder field of the AC. But the Holder field could still be an opaque string, understood by the Issuer at the Origin, and it doesn't have to be understood by the AC verifier at the Target site. See section 6 for a fuller discussion of this issue.

trust for the access control decision making. When the PERMIS PDP is initialised, it is given the name of the trusted authority, and the ID of the policy to use (each policy has a globally unique identifier). PERMIS reads in the policy certificates from the authority's LDAP entry, checks their signatures, and keeps the one with the correct ID. It now has the correct trusted policy with which to make access control decisions. PERMIS thus forms a good basis for demonstrating the distributed management of trust with Shibboleth.

4.1 The PERMIS PDP Policy

The PERMIS policy contains a list of trusted attribute authorities, the set of attributes they are trusted to assign, and the groups of users they can be assigned to. This is called the *role allocation sub-policy* (RAP). Attribute authorities can be distributed worldwide, and can be trusted to issue ACs to users from any domain, according to the RAP. When the PERMIS PDP is passed a set of attribute certificates by Shibboleth, it can determine from the RAP which are trusted to keep, and which should be discarded as untrusted. All the trusted attributes are extracted and stored for later access control decision making.

The PERMIS policy also contains the set of targets that are being protected by this policy, the associated actions that can be performed on them (along with their parameters), and the attributes (or roles) that a user needs in order to be granted the access. In addition, constraints can be placed on these grants, such as, only between 9am and 5pm, or only if the user holds non-conflicting roles[3], or only if the size is less than 3Mbytes etc. This is called the *target access sub-policy* (TAP). When the PERMIS PDP is asked if a user with the current roles/attributes is allowed to access a particular target resource, it consults the TAP and returns granted or denied based on its contents and the current state of the environment (time of day, resource usage etc.).

Because PERMIS can act in either push or pull mode with attribute certificates, then it is possible for a target site to create a policy that requires a user to have attributes issued by multiple different authorities in different domains, and the PERMIS PDP can then pull these at authorisation time regardless of the origin site that the user has actually authenticated to.

5. Supporting the different trust models of Shibboleth sites

One can immediately see that if Shibboleth and PERMIS are integrated together, then the enhanced distributed trust model that we wish to provide to target and origin sites can be obtained. However, a number of misalignments between PERMIS and Shibboleth need to be addressed first. Either Shibboleth needs to transfer X.509 attribute certificates (ACs) from the origin site to the resource site instead of plain attributes, or PERMIS needs to be modified to accept plain attributes instead of X.509 ACs. In fact, both of these methods have been implemented so as to provide resource sites with the maximum of flexibility. We have modified the Shibboleth origin site to retrieve X.509 ACs from its LDAP directory, and to pass these as text encoded binary attributes within the SAML attribute assertions. This facility should be provided as part of the standard Shibboleth software release during 2005. We have also modified the code that calls the PERMIS PDP to validate the plain attributes from Shibboleth and use these instead of or as well as X.509 ACs.

[3] Separation of duties is currently being implemented but is not in the current NMI release.

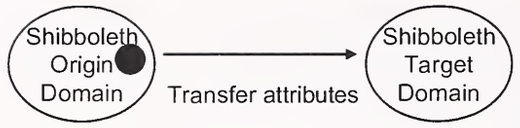
Since it is the target site's resources that are being accessed, we are primarily concerned with the trust that a target site is expected or required to have in the attributes that it receives in order for it to become Shibboleth enabled. An origin site will also have its own preferred trust model for the allocation of attributes, but the target site's trust model must always take precedence since it is the owner of the resources that are being accessed. We can look at trust from two different aspects: the distribution of trust in the attribute issuing authorities (centralised or distributed) and the trustworthiness of an origin site's attribute repository (trusted or not).

Firstly, we consider the distribution of trust. In the simplest case the origin site has a single attribute issuing authority. If the target site trusts the origin site's attribute authority, this authority can issue and sign all the SAML attribute assertions. (This is the standard Shibboleth model.) Alternatively, the origin site may wish to distribute the management of attributes between different trusted authorities in its domain and to allow dynamic delegation of authority. If the target site wishes to distribute its trust to these different authorities, then it can allow (trust) each one of them to issue and sign different attribute assertions, and further decide if it will allow dynamic delegation of authority to take place. Furthermore, in this distributed trust scenario, the target site may be willing to trust, or even require, some attribute authorities that are not even based at the origin site to issue attributes to users. (This is typically the case in today's world when one presents plastic cards from multiple different issuers in order to gain access to a particular resource e.g. access to an airport business lounge may be granted by presenting frequent flyer cards from a number of different airlines or diners clubs.) On the other hand, if the target site is not willing to recognise these multiple authorities, then the origin site will need to (re-)sign all the SAML attribute assertions by the single authority that the target site is willing to trust.

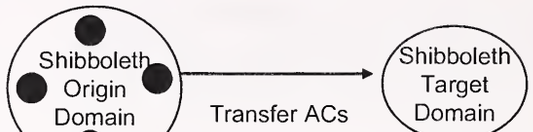
Secondly, we consider the origin site's attribute repository (typically an LDAP server). If either the target or origin site do not trust this to store unprotected attributes securely, then the origin will need to store digitally signed attributes in it, rather than plain attributes. We now consider each combination in turn. Figure 1 pictorially represents each of the trust models shown in the following sections.

5.1 Target trusts origin's attribute repository and origin as a single attribute authority

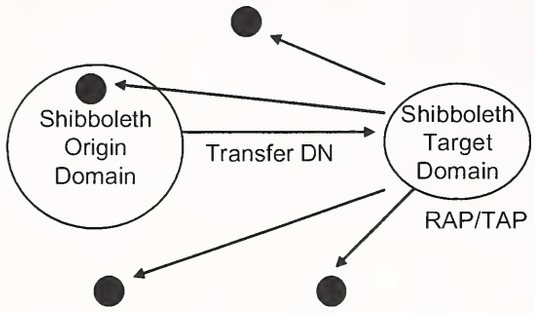
This is the original Shibboleth trust model and both the target site and origin site will use standard Shibboleth. The origin will store plain attributes in its repository, and pass them in digitally signed SAML messages to the target. The target site may use the standard Shibboleth authorisation mechanism, or optionally, for a finer grained and more refined access control mechanism, use a policy controlled PDP to make decisions. When using the PERMIS PDP for authorisation, the PERMIS target access sub-policy (TAP) is used to say which attributes are needed in order to gain access to the targets, and the (unsigned) attributes from the SAML message are passed to the PERMIS PDP.



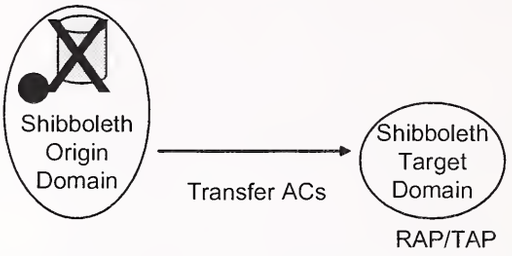
5.1 Standard Shibboleth



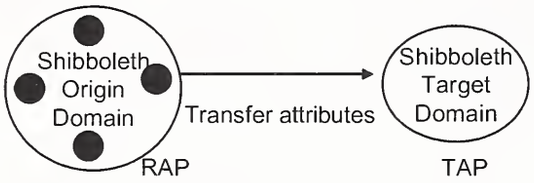
5.2 Multiple ACs at Origin



5.3 Pull ACs from multiple AAs



5.4 Untrustworthy attribute repository



5.5 Multiple AAs at Origin not trusted by Target

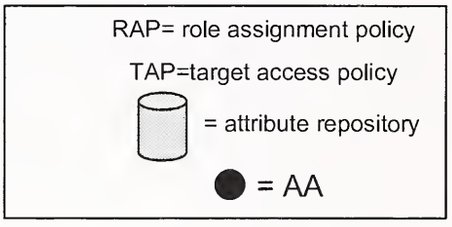


Figure1. Pictorial representation of different trust models

5.2 Origin wishes to distribute attribute assignments and target trusts different attribute authorities at the origin

In this scenario the origin distributes management between multiple authorities and therefore must store attribute certificates in its repository, so that the different attribute authorities can be recognised by the target. The target site uses the role assignment sub-policy (RAP) to describe who it trusts to assign which attributes to whom, and the TAP to determine which attributes are needed in order to access which targets. Note that the target may only trust a subset of the actual attribute authorities at the origin site, according to its RAP, and the policy specification allows for this. Additionally, the target may allow dynamic delegation of authority at the origin site, by specifying this in the RAP[4]. Shibboleth now fetches attribute certificates from the origin site, rather than plain attributes. Consequently the SAML attribute assertions do no need to be signed, though the link will still need to be SSL encrypted if privacy protection is required. In this scenario the origin's attribute repository may or may not be trusted by either the target or the origin, but this is not an issue since it is storing digitally signed ACs in the repository.

[4] Note that the enforcement of dynamic delegation of authority is currently being implemented and will be in a future release of PERMIS.

5.3 Target trusts different attribute authorities at the origin site and elsewhere

In this scenario, the target site wishes to authorise users based on attributes assigned to them by different attribute authorities that are not always co-located with the origin site. In this case, the origin site cannot push all the attributes to the target site (unless the AAs have distributed them to the origin site in the first place, which cannot be guaranteed), so the target will need to operate in pull mode and fetch the ACs that it needs directly from the AAs. The PERMIS PDP can operate in pull mode and fetch all the attribute certificates that are needed from the various distributed (LDAP) repositories. The SAML attribute assertions from the origin site do not need to carry any attribute certificates in this instance. They only need to provide the holder identity of the user, so that the target can know which ACs to retrieve. Of course, each attribute authority will need to let its repository (LDAP server) be accessed by the target site[5]. Once the ACs have been retrieved, the target's PDP will use the RAP to determine which ACs are trusted, and the TAP to determine if the user has the necessary attributes to access the resource.

5.4 Target and/or origin do not trust origin's attribute repository but target trusts origin as a single attribute authority

In this scenario the origin cannot store unsigned attributes in its repository, but rather should store digitally signed attributes in its (LDAP) repository. The exact format of these could be X.509 attribute certificates or (long lived) SAML attribute assertions. These should all be signed by the same organisational attribute authority that is trusted by the target. Shibboleth will then carry either signed attribute certificates or signed SAML assertions to the target site. (Note that the latter is equivalent to the model in 5.1). When the ACs are handed to the PDP, the RAP will check that they have been issued by the sole origin authority. The TAP is then used to determine if the user has sufficient attributes to be granted access to the target or not. When Shibboleth is transferring attribute certificates in the SAML assertions, the assertions do not need to be signed, though SSL encryption will be needed if privacy protection is required.

5.5 Origin wishes to distribute trust to multiple authorities, but target does not recognise them

In this scenario the target wishes to run standard Shibboleth but the origin wishes to distribute the management of attributes to different AAs i.e. to run its own PMI, with all the advantages this brings such as dynamic delegation of authority. The origin will be creating and storing attribute certificates in its AC repository signed by multiple distributed attribute authorities. However, because the target wishes to run standard Shibboleth, and wants a single point of trust at the origin, these ACs cannot be passed to the target. Therefore the origin site should run a PDP with its own RAP to validate that the ACs are issued in accordance with its own policy. This will validate the stored attribute certificates, extract the attributes that are trusted and pass these to the local Shibboleth origin server for transfer in signed SAML attribute assertions to the target. The target site can then run the standard Shibboleth authorisation module, or for finer grained control can run its own PDP and TAP, as in 5.1, to determine if the user is to be granted access or not.

[5] Note that if a site's firewall prevents the LDAP protocol from passing through, there are several http to ldap gateways available that allow the firewall to be tunnelled through on port 80.

6 User Privacy Issues

One of the limiting factors of X.509 attribute certificates (ACs) is that they bind the attributes to the holder, and if the holder is identified by their real name in the AC e.g. {CN=David Chadwick, OU=Computing Laboratory, O=University of Kent, C=GB} then the user's privacy is (at least partially) lost. There are a number of solutions to this problem. X.509 ACs allow holders to be identified in a number of different ways. Firstly, they can be identified by a distinguished name (DN). However, this DN does not need to be the real name of the holder or indeed in any way be similar to the holder's real name. It can be a pseudonym rather than their real name e.g. {CN=123456789}, or even a group name e.g. {CN=Programmer, OU=Computing Laboratory, O=University of Kent, C=GB}. This opaque name only needs to have meaning to the issuing site. The mapping between the user's login/authentication identity and AC holder identity would be performed at authentication time by the origin site's authentication server. It is important to note that the binding between the pseudonym in the AC and the authentication name of the human user is handled not by normal PKI registration procedures, but by the origin authentication system, so that the target site's trust in user authentication has to be placed in the origin site's systems and not in a trusted third party CA. Further, the use of pseudonyms or group names will make it much more difficult for independent AC issuers to participate in distributed trust management, since they will need to liaise with the origin site to know which opaque names have been given to which users.

The difference between using a pseudonym and a group identity is that in the former case the target site would be able to profile the user, without knowing the real physical identity of the user. With a group identity the target site would only be able to profile the whole group, and would not be able to differentiate between different group members, or know how many members were in the group.

Secondly, the holder can be identified indirectly by reference to their X.509 public key certificate. In this case the attribute certificate holds the serial number and issuer name of the user's public key certificate e.g. {x509serialNumber=123456 + x509issuer = {OU=Some CA, O=Some Org, C=US}. The limitations of this method are that the user must be PKI enabled, which of course, many are not; and that, depending upon the contents of the user's public key certificate, the user might be identified via this.

Finally, the holder can be identified indirectly by reference to the hash of a public key that they hold. This is now effectively a random number, giving good privacy protection. The user can prove ownership of the attribute certificate by digitally signing a challenge provided by the origin authentication server, which can then provide this AC to the target site. The restrictions are that the user needs to be using some form of asymmetric cryptography, has generated their own private/public key pair, has created a self signed certificate with a random DN and does not have a corresponding X.509 public key certificate identifying him/her. The main limitation from a privacy perspective is that the target site can profile the user, without knowing the actual identity of the user, since the same public key hash is used each time.

In all these cases there is a trade-off between the "degree of anonymity" and the "quality of issuance". At one extreme we have dynamically generated Shibboleth short lived signed SAML attribute assertions that

provide anonymity but require a trusted directory to store the user's attributes. At the other extreme we have long lived ACs where each attribute authority can issue its own attributes in a controlled manner, but without any privacy protection. At various points in the middle we have long lived ACs with various forms of privacy protection (pseudonyms, group names and public key identifiers) where the AA or authentication system maps the user's name into a privacy protected one.

In addition to protecting the identity of the AC holder, the AC issuer name may also be protected in the same ways as above. Note that the name of the SOA may be privacy protected or pseudonymised in some way, but the target PDP will need to know who this name actually belongs to if it is to be configured as a root of trust. The privacy of the embedded attributes may be protected by encryption, as described in [3] and [7]. Different attributes can be encrypted for different target sites. The main disadvantage of encrypted attributes is that the issuer needs to know in advance, when creating the AC, who the targets are going to be. This of course may not always be possible with relatively long lived ACs, in which case SSL encryption of the communications link is a better option for attribute privacy.

7 Revocation and Performance Issues

The signed SAML assertions of Shibboleth do not need to be revoked due to their short life times. Attribute certificates on the other hand are expected to have a relatively long life time, according to the privileges/attributes being allocated. For example, one might expect a "student" attribute certificate to be valid for an entire academic year. Whilst signed SAML attribute assertions have the performance overhead of requiring a digital signature per message sent by the origin site, long lived ACs may have the overhead of requiring revocation list processing, depending upon how they are stored and distributed. If the ACs are stored in a repository under the control of the issuer, and are retrieved from there by either the Shibboleth origin or target sites, or directly by the target's PDP, then a revocation list may be avoidable providing the issuer deletes the ACs when they need to be revoked, and third parties are not able to surreptitiously write them back again. In this way the revoked ACs will not be available to the PDP when either it or the Shibboleth components try to retrieve them. If on the other hand the ACs are not stored in a repository under the control of the issuer, for example, they are distributed directly to their holders, then standard attribute certificate revocation lists (ACRLs) will be needed, and the issuer will need to periodically update them, in exactly the same way as for public key certificate CRLs. The PDP will need to ensure that it has a current ACRL when validating the ACs that have been presented to it. This will cause some performance overhead at the target site. Short lived ACs on the other hand do not need ACRLs to be published, just as the short lived SAML assertions do no require them. Whilst short lived ACs do not have the distributed trust management benefits of long lived ones (one cannot expect human managers to issue ACs to their staff daily, whilst automated issuing servers have the same trust related problems as existing Shibboleth implementations), they do have a significant performance benefit over signed XML messages [8] [9], so they might still be worthy of consideration in this respect.

8 Conclusions

We have shown how a distributed, finer grained and more functional trust model can be added to Shibboleth, to increase the

latter's flexibility and authorisation decision making capabilities. We have further shown how the model can support target and origin sites using different combinations of centralised and distributed trust models, and different assumptions concerning the trustworthiness of the origin's attribute repository. We have implemented this distributed trust model in Shibboleth by combining it with the PERMIS authorisation infrastructure and X.509 attribute certificates. Finally we have argued that user privacy does not need to be compromised *per se* by using long lived X.509 attribute certificates instead of short lived digitally signed SAML attribute assertions, although it is certainly more difficult to fully protect a user's privacy in the former case.

8 Acknowledgements

The authors would like to thank the UK JISC for funding this work under the SIPS project.

9 References

[1] Scot 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

[4] David Chadwick, Sassa Otenko, Von Welch. "Using SAML to link the GLOBUS toolkit to the PERMIS authorisation infrastructure". Proceedings of Eighth Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Windermere, UK, 15-18 September 2004

[5] OASIS. "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1", 2 September 2003

[6] OASIS. "eXtensible Access Control Markup Language (XACML)" v1.0, 12 Dec 2002, available from http://www.oasis-open.org/committees/xacml/

[7] S. Farrell, R. Housley. "An Internet Attribute Certificate Profile for Authorization". RFC 3281, April 2002

[8] Mundy, D. and Chadwick, D.W., "An XML Alternative for Performance and Security: ASN.1", IEEE IT Professional, Vol 6., No.1, Jan 2004, pp30-36

[9] "Fast Web Services," P. Sandoz and colleagues, Aug 2003, available from http://java.sun.com/developer/technicalArticles/WebServices/fastWS/

Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration

Von Welch,[1] Tom Barton,[3] Kate Keahey,[2] Frank Siebenlist[2]
[1]National Center for Supercomputing Applications, University of Illinois
[2]Mathematics and Computer Science Division, Argonne National Laboratory
[3]Networking Services & Information Technologies, University of Chicago

Abstract

In this paper we describe our work in progress to integrate the Shibboleth SAML-based framework and Globus Toolkit's PKI-based security infrastructure. The result will provide identity federation and attribute-based policy enforcement for Grids that leverages the Shibboleth system being deployed on campuses. We provide an overview of both Shibboleth and the Globus Toolkit, present our motivating use cases, and describe our planned integration work.

1 Introduction

As virtual organizations (VOs) [9] increasingly turn into distributed multi-institutional collaborations, secure authentication and authorization become a growing challenge. In the existing Grid [8] infrastructure to support VOs, these mechanisms are typically based on the identities of the interacting entities. While this approach is simple and intuitive, as VOs expand, it becomes impractical to administer. VO membership may change dynamically, rights may be granted to entities on a periodic basis, or a user's role in an organization might dynamically evolve. Such factors make it more practical to express users' rights based on their other attributes, such as institutional affiliation or role in a collaboration, rather than identity alone.

Indeed, it may be desirable to enable anonymous interactions between users, thus protecting individual privacy while still providing basic security services to system owners.

In this paper, we present our work to address this issue by integrating two widely accepted technologies: Shibboleth [20], an Attribute Authority service developed by the Internet2 community for cross-organization identity federation, and the Globus Toolkit's [10] Grid Security Infrastructure (GSI) [26]. Our project, which is funded by the NSF National Middleware Initiative [16], is known informally as "GridShib" [11]. The objective is to provide mechanisms whereby a Grid service can authenticate a user using GSI, determining the address of the Shibboleth attribute service in the process, and then obtain from the Shibboleth service the select user attributes that the Grid service is authorized to see. Attributes obtained in this way can then be used by the Grid service in making authorization decisions.

In Section 2, we describe Shibboleth and the relevant security portions of the Globus Toolkit. Section 3 introduces the use cases we plan to address. In Section 4 we discuss our plans for the Globus-Shibboleth integration, describing modes of usage, technical details, and planned implementation. In Section 5 we compare related technologies. In Section 6 we summarize our plans and conclude the paper.

2 Background

In this section we provide an overview of the two software products relevant to our work: Shibboleth and the Globus Toolkit. A more detailed description of Shibboleth [20] and the Globus Toolkit [10] can be found on the individual Web sites. We also describe the authentication standards used by each of these software systems.

2.1 Shibboleth

Shibboleth is a system that asserts attributes about a user between organizations. More precisely, it asserts attributes between the user's home organization and organizations hosting resources that may be accessible to the user. By using an attribute-based authorization model, the Shibboleth architecture is able to protect user privacy better: identifying information may, but need not, be among the attributes presented about the user.

Shibboleth can be conceptually regarded as comprising three components:

- *Handle Service*: The Handle Service authenticates users in conjunction with a local organizational authentication service and issues to the user a *handle token*. The handle token (comprising a SAML authentication assertion [19]) is a bearer credential containing an identifier, or *handle*. The Handle Service is intentionally neutral to the choice of the organizational authentication mechanism and can function with almost any such service (e.g., LDAP [25]).
- *Attribute Authority*: When a user requests access to a target resource, he presents his handle token. The resource then presents the user's handle token to the attribute authority and requests attributes regarding the user. The attribute authority enforces privacy policies on the release of these attributes, allowing the user to specify which targets can access which attributes. The Shibboleth Attribute Authority retrieves attributes from an organizational authority and provides them in the form of SAML assertions. As is the case with the Handle Service, the Attribute Authority is intentionally neutral to the specific implementation of the organizational attribute service; LDAP is typically used today
- *Target Resource*: The target resource includes Shibboleth-specific code to determine the user's home organization and hence which Shibboleth attribute authority should be contacted for the user, to retrieve attributes regarding the user, and to make authorization decisions based on those attributes.

In normal Shibboleth usage, a new handle token is acquired each time the user accesses a different resource. A handle token can be reused on returning to a resource for a relatively short period of time. Each handle token has a different unique identifier for the user that is meaningful only to the Shibboleth attribute authority. As a result, a target resource cannot rely on the handle to learn the true identity of a Shibboleth user, nor can it correlate subsequent accesses as coming from the same user.

The current implementation of Shibboleth (version 1.2) is primarily designed to function with Web applications (i.e., standard Web servers and browsers). Plans for future implementations (starting with version 1.3) include support for non-Web applications.

Figure 1 shows the typical operation of Shibboleth. A number of steps not relevant to this paper are omitted for clarity.

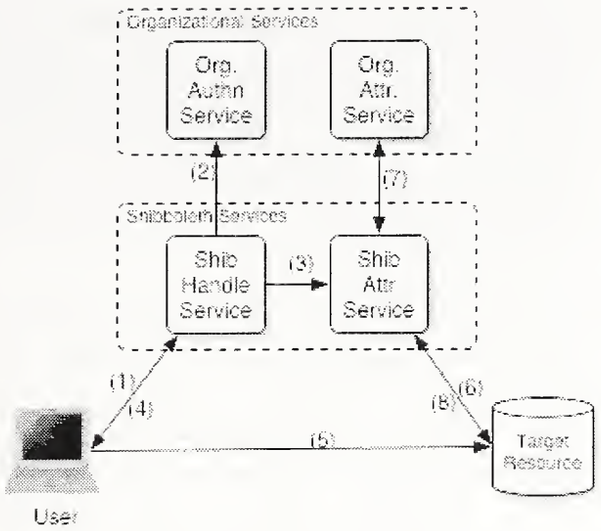


Figure 1: Simplified Shibboleth architecture and typical usage. Steps are described in the text.

The steps shown in Figure 1 are as follows:

1. The user connects and authenticates to the Handle Service.
2. The Handle Service uses a local organizational authentication service to verify the user's authentication information.
3. The Handle Service creates a new handle to identify the user. This handle is registered with the attribute authority so that it can be mapped to the user's attributes when a request from a resource arrives.
4. The handle is placed into a handle token and returned to the user.
5. The user sends a request to a target resource and presents the handle token.
6. The resource examines the handle token to determine which Shibboleth service can provide attributes about the user. It contacts that Shibboleth service and requests attributes, providing the handle token to identify the user.
7. After validity checks have been performed on the handle token and the handle has been mapped to the user's identity, the applicable attribute release policy for that resource is checked whether communication of the requested user attributes is allowed. If so, the requested attribute values are retrieved.
8. The Shibboleth attribute authority casts the attributes in the form of a SAML attribute assertion and returns the assertion to the target resource.
9. (Not shown) After receiving the attributes from Shibboleth, the target resource makes an authorization decision regarding the user's request based on those attributes.

2.2 Globus Toolkit

The Globus Toolkit provides basic functionality for Grid computing [8], with services for data movement and job submission, and a framework on which higher-level services can be built. The Grid in general has been adopting Web services technologies, and this trend is reflected in recent versions of the Globus Toolkit in following the Open Grid Services Infrastructure [24] and now the Web Services Resource Framework [29] standards. This convergence of Grid and Web services was part of our motivation for adopting Shibboleth in our project (which uses the SAML standard).

The Grid Security Infrastructure, on which the Globus Toolkit is based, uses X.509 identity certificates [12] and X.509 proxy certificates [23, 27]. In brief, these certificates allow a user to assert a globally unique identifier (i.e., a distinguished name from the X.509 identify certificate).

We note that in Grid scenarios there is often a clear separation between the certificate authorities (CAs), which are the authorities of identity, and the authorities of attributes or authorization. For example, in the case of the DOE SciDAC program [18], a single CA, the DOE Grids CA [3], serves a broad

community of users, while the attributes and rights for those users are determined by their individual projects (e.g., National Fusion Grid, Earth Systems Grid, Particle Physics Data Grid).

Authorization in the Globus Toolkit is based on access control lists for each resource that specify the identifiers of the users allowed to access the resource. Higher-level services to provide richer authorization exist; we discuss these, and their relationship to this work, in Section 5.

2.3 GridLogon/MyProxy

GridLogon is a credential service being developed at NCSA as an extension to the popular MyProxy service [15]. MyProxy is a credential management service and is the de facto mechanism used to provide security to Grid portals worldwide.

Simply put, GridLogon acts as an online-CA, authenticating the user through a variety of mechanisms and issuing (short-lived) X.509 identity credentials suitable for use with Grid resources. GridLogon will provide a pluggable authentication mechanism to allow for the use of different local authentication systems, such as username/password, One-Time-Password, Kerberos, and Public Key.

3 Motivating Use Cases

In this section, we describe the use cases we wish to support with our work.

3.1 Project Leveraging Campus Attributes

The first scenario, shown in Figure 2, resembles the basic model in which Shibboleth is used today, except that the target resource is a Grid service instead of a Web-based application.

In this scenario, authorized users of the Grid service are located at one or more campuses and can be described by some campus-oriented attribute (e.g., chemistry professor). Verifying this attribute at their home institution authorizes user access to the Grid services.

An example of where this service could be applied in a Grid context is TeraGrid [1]. Each site on TeraGrid could operate a Shibboleth service in order to identify their staff and user community. This would enable TeraGrid resources to be easily available to the entire TeraGrid community without having comprehensive access control lists maintained on the resource.

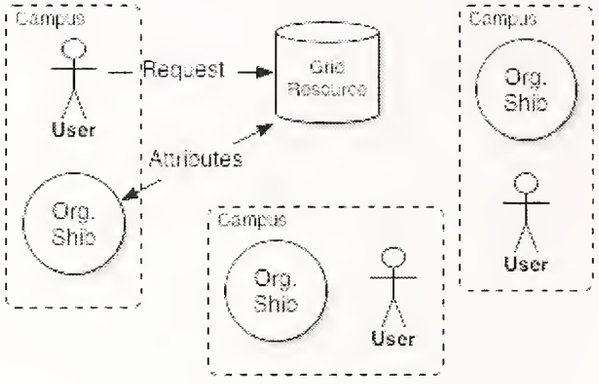


Figure 2: Scenario showing users being authorized based on their campus-assigned attributes.

3.2 Project-Operated Shibboleth Service

Figure 3 depicts a different scenario, where the project deploys and operates its own attribute authority. This scenario has the benefit that the project can freely assign attributes and add users as it wishes, without involving campus administration staff. However, the project must itself operate the Shibboleth service, a critical requirement from the perspective of both security and reliability. This approach is beyond the scope or capabilities of many projects.

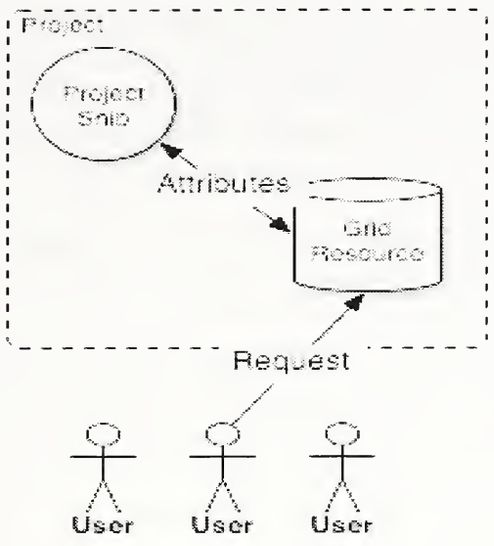


Figure 3: Scenario showing Shibboleth service operated by a project.

3.3 Campus-Operated, Project-Administered Approach

The scenario shown in Figure 4 is a hybrid of the two preceding scenarios. It empowers the project to administer its own attribute space while allowing the Shibboleth service to be maintained by campus staff who are expert in running such services.

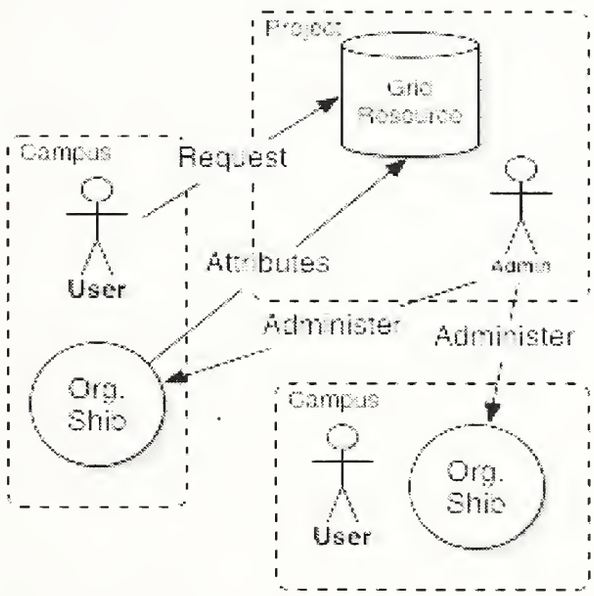


Figure 4: Scenario showing hybrid mode of operation, with the campus operating the Shibboleth service and the project administering its portion of the attribute space.

While we believe this hybrid approach to be the best of the three scenarios, it does require some form of administration delegation capabilities that is not present today. We address this issue in Section 4.3.

4 GSI-Shibboleth Integration

Our work comprises five major components:

- *Assertion Transmission* – to enable the transmission of assertions from the Shibboleth service to the Grid software and ultimately the Grid runtime authorization decision-making component.
- *Attribute Authority* – to enable discovery of the appropriate attribute authority for a user in the Grid context. Since Grid resources serve users from multiple organizations, a mechanism is needed to determine which organization's Shibboleth service is authoritative for a particular user.
- *Distributed Attribute Administration* – to manage subsets of an attribute space served by a Shibboleth service by projects outside the domain operating the Shibboleth service (as described in Section 3.3).
- *Pseudonymous Interaction* – to extend to Grids the pseudonymous interaction provided by Shibboleth.
- *Authorization* – to provide a mechanism that is integrated with the Globus Toolkit and can take advantage of the attributes.

4.1 Assertion Transmission

The fundamental engineering problem that must be solved is how to transmit user attributes from a Shibboleth attribute service to the Grid resource so that an authorization

decision can be made utilizing those attributes.

Two fundamental modes of operation address this problem:

- *Pull mode*: The target Grid service, after authenticating the user, will contact the appropriate Shibboleth service to obtain attributes regarding the user. This is analogous to normal Shibboleth operation today, as described in Section 2.1.
- *Push mode*: The Grid user, before contacting a target service, will contact an appropriate Shibboleth service to obtain attributes and then convey those to the target Grid service at the time of making a request. This is analogous to how other attribute authority systems in the Grid context work today, described in Section 5.

The pull mode of operation has the advantage of being more easily deployed; since clients of services are not affected and do not even need to know Shibboleth is involved in their decision-making. However, as we describe in the subsequent section on Attribute Authority discovery, the push mode has the advantage of allowing a user to select a particular role. Hence we plan on implementing both, to allow for flexible deployment to meet the requirements of different projects.

Regardless of the mode chosen, there exists the issue of federating the Grid identities, which consist of X.509 distinguished names (DNs), with the local identities used by the organization operating the Shibboleth service. This federation will require that a mapping be performed between the DN and the local site identifier. Shibboleth, as described in Section 2.1, already performs a similar mapping from the handle issued by the Handle Service to the local identity, and the upcoming release of Shibboleth (version 1.3) will support a generalized version of this mapping feature capable of supporting DNs, which will solve the basic problem of mapping identifiers.

4.2 Attribute Authority Discovery

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 that can provide attributes regarding the user. This is often referred to as the "Where are you from?" (WAYF) problem.

Shibboleth currently addresses this problem by asking users to identify their home organization when they attempt to access the target resource. In its current model of supporting interactive Web browser-based applications, this approach is acceptable. In the Grid context, however, where the user may be using client software that does not support this level of interactivity or the client may be an unattended batch job, we need a different approach.

We will explore the following possible solutions:

- Use the push mode of operation, described in Section 4.1 and have the user select the attribute authority to contact. This approach has been taken by other systems, such as VOMS described in Section 5.1. The main drawback is that it requires modification of client behavior or software, which can present a deployment challenge.
- Place a pointer (e.g., a hostname) to the attribute authority to contact 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 are not in synch.

- Place a pointer to the attribute authority's location in the user's proxy certificate. Since the user can create proxy certificates fairly easily and with short lifetimes, this approach solves a number of problems with having the issuing CA place information in longer-term identity certificates. The actual placing of the information could probably be automated, and users could select from different attributes authorities, and even multiple authorities, depending on the specific role or roles they want to adopt.

A related challenge that we will explore is a scenario where a user may be acting in a role that combines attributes from multiple organizations or from the project and an organization. In this scenario the user's attributes would come from multiple Shibboleth services. It remains unclear at time whether this is a true requirement for a user community, so our exploration of this problem may be minimal.

4.3 Distributed Attribute Administration

The current Shibboleth attribute management paradigm assumes that the complete attribute space asserted by a particular attribute authority is managed within a single administrative domain. This model makes sense when all the attributes are concerned with the user's role in the single domain; for example, if the administrator works for the user's university and the attributes all concern the user's position at the university.

This attribute management model does not support resource targets wanting to use attributes that are asserted by other authorities. One example is an issue that is already being faced by the Shibboleth community and is known as the "IEEE problem": having universities provide IEEE membership status to allow resource targets to authorize based on their IEEE membership rather than on their campus affiliation. While the authoritative party for the attributes, IEEE in this case, could establish its own Shibboleth service, this approach may not always be desirable because some organizations may not have the resources or skills to operate a highly available secure attribute authority service.

A new privilege management system called Signet [21], which is being developed by a working group of the Internet2 Middleware Initiative, supports the distributed administration of privileges. Shibboleth-enabled access to Signet is planned, which will enable authorities outside of the administrative domain in which a Signet instance is operated to be delegated the ability to manage a portion of the attribute space that can be asserted by that domain's attribute authority. This arrangement has the potential to support the use case described in Section 3.3.

In collaboration with the Signet development team, we will explore the possibility of allowing administrative delegation of the attribute space in a single attribute authority service among multiple organizations as a means to solve this problem.

4.4 Pseudonymous Access

Shibboleth allows for pseudonymous access as part of its normal operation. To provide anonymity in the Grid context, we will integrate the GridLogon service with

Shibboleth and the Globus Toolkit. As we described in Section 2.3, the GridLogon service issues X.509 certificates to authenticated clients. We will implement an extension to GridLogon module that issues an authenticated client a set of credentials with a pseudonym identifier, which will make the GridLogon service essentially act as the Shibboleth Handle Service normally does. GridLogon will register the pseudonym with the Shibboleth attribute service, such that subsequent queries can be mapped to the user's attributes.

4.5 Authorization Mechanism

To allow for the use of attributes in the Globus Toolkit runtime, we need to extend the current ACL-based authorization mechanism to encompass attribute-based policy. We intend to leverage existing standards and implementations here to the greatest extent possible. Our implementation will most likely be very simple at first, for example, using attributes in place of identities to map users to local accounts.

We will explore the integration of XACML [6] with the Globus Toolkit security runtime, with a mapping of SAML attribute assertions to XACML attribute assignments as described in [14]. The result will be a Web services runtime that can make authorization decisions about the user's invocation request of the Web service operations based on the Shibboleth user's attribute and XACML policy rules. The aim is to make the attribute retrieval and the evaluation and enforcement of the authorization policy transparent to the application.

The Globus Toolkit currently supports an authorization callout [28], which allows external services to provide authorization decisions for Globus deployments as described in Section 5.3. Our goal is to provide attributes received from Shibboleth to those external authorization services in order to allow them to incorporate those attributes in their decision-making process.

In parallel with our GridShib effort, the Globus Toolkit team has also started work on a more ambitious authorization-processing framework. As the toolkit is used by many different Grid applications and projects worldwide, it cannot mandate specific security technologies and mechanisms, and has to adopt a modular approach to accommodate the choices made by those responsible for deployment. For example, identity and attribute assertions have to be supported in X.509 Identity and Attribute Certificate, Kerberos, and SAML Identity and Attribute Assertion formats. Furthermore, all these statements can either be available in local storage within the trusted computing base of the relying party, be pushed by other parties via SOAP headers or Proxy Certificate embedding, be pulled from online services, or external attribute and authorization services can be queried through Shibboleth/SAML call-out interfaces.

In the first step of this authorization processing, the received and collected assertions with their associated issuers are verified and validated. The resulting attribute statements with their issuer-subject information are subsequently translated and mapped by mechanism specific Policy Information Points (PIPs) into a common format that is presented to the Policy Decision Point (PDP). Our GridShib effort should be able to leverage this authorization framework development work.

4.6 Planned Timeline

The GridShib project officially began in December 2004. Prior to that date we had identified requirements and made

preliminary project plans. We are now focusing on implementing the pull mode as described in Section 4.1; we expect to have a first release by the summer of 2005 based on the upcoming release of Shibboleth (version 1.3) and a post-4.0 version of the Globus Toolkit. Enabling the push mode and pseudonymous access will follow in 2006.

5 Related Work

Our work is distinguished from related work mainly through the Shibboleth support for pseudonymous interaction, its fine-grained attribute release policy, and its existing broad support base in the Internet2 community.

5.1 VOMS

The virtual organization management service (VOMS) [5] was developed by the European Data Grid project to allow for attribute-based authorization to the Globus Toolkit job management services. It uses X.509 attribute certificates [7] in a push mode to assert attributes in a modified version of the Globus Toolkit.

We believe that Shibboleth, with its use of SAML, will be more easily interoperable with Web services-based technologies emerging in the Grid community. VOMS also does not support a pseudonymous mode, nor does it have any other provisions for privacy support.

5.2 CAS

The Community Authorization Service (CAS) [17] is similar to Shibboleth in its use of SAML assertions. However CAS operates at the level of capabilities rather than attributes; that is, instead of expressing abstractly what someone is, CAS expresses explicitly what actions they are allowed to take. CAS also does not support a pseudonymous mode or have any other provision for privacy.

5.3 Akenti and PERMIS

Akenti [22] and PERMIS [2] are authorization systems that have been integrated with the Globus Toolkit through the use of authorization callouts [13,28]. Both Akenti and PERMIS allow for the use of X.509 attribute certificates to make attribute-based authorization decisions. We envision our work as being complementary to these systems. Our focus falls on the technology to transport SAML assertions from the Shibboleth attribute authority to the Globus Toolkit-based services, whereas these systems are designed primarily as authorization decision makers. We envision a mode of operation in which these systems can be used to provide rich authorization capabilities using Shibboleth-issued SAML assertions in addition to the X.509 attribute certificates they use today.

5.4 Signet

The Signet privilege management system [21] is being developed by a working group of the Internet2 Middleware Initiative. Signet can manage privileges that are expressed as attributes asserted by Shibboleth. Signet itself is planned to be "Shibbolized" to support delegation of privilege management beyond the bounds of a single organization.

As discussed in Section 4.3, we plan to collaborate with the Signet team, in order to enable Signet to manage the access policy to Grid resources.

5.5 ESP-Grid Project

The ESP-Grid project [4] is evaluating how Shibboleth could be used to benefit Grid authentication. We have met with the members of the ESP-Grid project and will

stay in contact, sharing experiences and results of our work.

6 Conclusion

We have described the motivations for identity federation and for attribute-based authorization in Grids. We have described our plans for addressing these motivations through integration of Shibboleth and the Globus Toolkit in order to produce a system capable of enabling attribute-based authorization in Grids, leveraging existing campus Shibboleth infrastructure, and allowing for pseudonymity.

Acknowledgments

This work is funded by the NSF National Middleware Initiative, under award SCI-0438424.

Our work would not be possible were it not for our collaboration with the Internet2 Shibboleth development team chaired by Steve Carmody.

We also thank Ian Foster and Ken Klingenstein for their advice and guidance.

Author Contact Information

Tom Barton
tbarton@uchicago.edu

Kate Keahey
keahey@mcs.anl.gov

Frank Siebenlist
franks@mcs.anl.gov

Von Welch
vwelch@ncsa.uiuc.edu

References

1. Catlett, C. The TeraGrid: A Primer, 2002. www.teragrid.org.
2. Chadwick, D.W. and Otenko, A., The PERMIS X.509 Role Based Privilege Management Infrastructure. 7th ACM Symposium on Access Control Models and Technologies, 2002.
3. DOEGrids Certififcate Service, http://www.doegrids.org, 2004.
4. ESP-GRID – Evaluation of Shibboleth and PKI for GRIDS. http://e-science.ox.ac.uk/oesc/projects/index.xml.ID=body.1_div.20, 2004.
5. EU DataGrid, VOMS Architecture v1.1. 2003. http://grid-auth.infn.it/docs/VOMS-v1_1.pdf.
6. eXtensible Access Control Markup Language (XACML) 1.0 Specification, OASIS, February 2003. http://www.oasis-open.org/committees/xacml/
7. Farrell, S., and Housley, R., An Internet Attribute Certificate Profile for Authorization. RFC 3281, IETF, April 2002.
8. Foster, I., and Kesselman, C. (eds.). *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.
9. Foster, I. Kesselman, C., and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications, 15(3)*, 200–222, 2001.

10. Globus Toolkit. http://www.globus.org/, 2004.
11. GridShib Project Web site, http://grid.ncsa.uiuc.edu/GridShib
12. Housley, R., Polk, W., Ford, W., and Solo, D., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *RFC 3280*, IETF, April 2002
13. Keahey, K., Welch, V., Lang, S., Liu, B. and Meder, S.. Fine-Grain Authorization Policies in the Grid: Design and Implementation. In 1st International Workshop on Middleware for Grid Computing, 2003.
14. Lorch, M., Proctor, S., Lepro, R., Kafura, D., and Shah, S., First Experiences Using XACML for Access Control in Distributed Systems, ACM XML Security Workshop, October 2003.
15. Novotny, J., Tuecke, S., and Welch, V., 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
16. NSF Middleware Initiative (NMI). 2004. www.nsf-middleware.org.
17. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.
18. Scientific Discovery through Advanced Computing (SciDAC), http://www.scidac.org, 2001.
19. Security Assertion Markup Language (SAML) 1.1 Specification, OASIS, November 2003.
20. Shibboleth Project, Internet2, http://shibboleth.internet2.edu/
21. Signet, http://middleware.internet2.edu/signet/, 2004.
22. Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., and Essiari, A., Certificate-based Access Control for Widely Distributed Resources. 8th Usenix Security Symposium, 1999.
23. Tuecke, S., Welch, V. Engert, D., Thompson, M., and Pearlman, L., Internet X.509 Public Key Infrastructure Proxy Certificate Profile, *RFC 3820*, IETF, 2003.
24. Tuecke, S., et. al. Open Grid Services Infrastructure (OGSI) Version 1.0, Global Grid Forum, 2003.
25. Wahl, M., Howes, T., and Kille, S., Lightweight Directory Access Protocol (v3), RFC 2251, IETF, 1997.
26. Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., and Tuecke, S., Security for Grid Services. In *12th IEEE International Symposium on High Performance Distributed Computing*, (2003).
27. Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S. and Siebenlist, F.. X.509 Proxy Certificates for dynamic delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop*, 2004.
28. Welch, V., Ananthakrishnan, R., Meder, S., Pearlman, L., and Siebenlist, F., Use of SAML for OGSA Authorization (work in progress), Global Grid Forum, May 14, 2004.
29. WS-Resource Framework. http://www.globus.org/wsrf/, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf, 2004.

XPOLA – An Extensible Capability-based Authorization Infrastructure for Grids

Liang Fang and Dennis Gannon
Computer Science Department, Indiana University, Bloomington, IN 47405
Frank Siebenlist
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439
{lifang,gannon}@cs.indiana.edu, franks@mcs.anl.gov

ABSTRACT

There is great need for a secure, fine-grained, efficient, and user-friendly authorization infrastructure to protect the services in Grid community. Grid users and administrators still have to deal with authentication and authorization issues in the traditional supercomputer-centric fashion, especially with the host account maintenance and certificate management. This paper proposes a capability-based infrastructure that provides a fine-grained authorization solution to Web service deployments, and also manages to hide complex security issues from regular Grid users. Furthermore, it gives the resource providers and administrators the extensibility, flexibility and convenience to enforce their authorization policies at the resource with minimal efforts.

1. INTRODUCTION

1.1 The Grid and Its Security

Since the end of the last millennium, the Grid technologies have profoundly changed the way scientists compute by supporting collaboration and resource sharing securely across multiple domains. Interests from academia and industry in Grid technologies lead to a series of frameworks and applications, such as the Globus Toolkit and e-Science [6, 11]. The emerging Grid technologies are leveraging the Web services efforts, and are guided by Open Grid Services Architecture (OGSA) at the Global Grid Forum [29].

To realize the resource sharing across multiple domains, the underlying security infrastructure plays a key role. The widely used Grid Security Infrastructure (GSI), first integrated in Globus, provides secure communication, mutual authentication, and coarse-grained single sign-on (SSO) access to remote resources [7, 27]. It allows a user to access her resources across administrative domains through the use of her proxy certificates, and by delegating her rights to remote agents that manage remote resources on the user's behalf. The assumption it is based on is that the user had an account on the remote host and the remote agent is able to locate a mapping from the user's Grid identity to an associated local host account. This model has worked reasonably well for large, centrally coordinated Grids like NASA's Information Power Grid [12] and NSF's TeraGrid [3]. However, as the size and diversity of these research collaborations grow larger, it becomes increasingly difficult and sometimes impossible to stay with this supercomputer-centric model. For example, when a project has an educational outreach program that encourages thousands of grade school students to use services provided by the research Grid, we cannot assume these students all have Grid accounts on central resources. Indeed, for any research collaboration that does not have a strong centrally managed organization that can exert influence over resource providers, creating user accounts and managing Grid tools like Globus can be a serious administrative problem if it involves more than a few machines.

Specifically, in order to obtain access to a specific remote resource, even just temporarily, a Grid user-to-be has to go through the following steps:

First, she needs an X.509 certificate issued by a certificate authority (CA) that is trusted by the remote host. The user is supposed to manage her private key securely, and make those credentials accessible to the Grid-enabled client software.

As an alternative, a Grid portal solution provides the interface to Grid services through web servers and standard web browsers. This is definitely seen as a relief by the users because it frees them from the Grid security configuration pains. However, the user is still responsible for loading her certificate from a Grid-enabled host into a credential server, such as MyProxy, which stores the Grid users' credentials. From the user's credentials, the MyProxy server derives short-lived proxy certificates [28] after receiving requests from Grid portals or other Grid-enabled applications [18].

The configuration complexity is mirrored on the remote host. The user will have to contact the system administrator for an account on the remote machine that she wants to use as a computing resource. After being given an account with a user name, she has to ask the Grid administrator to add the mapping entry of her username and her certificate's X.500 distinguished name (DN) into a gridmap file. At runtime, GSI will authenticate the user and map her Grid identity to the local account according to the gridmap file. In some cases, the user may also be responsible for setting up a separate hosting environment under that account.

The administrators have the issue of maintaining an exploding number of user accounts, of which many will only

be used for a short time, or never be used at all. As a result, a significantly number of allocated resources is wasted in this way.

One of the identified problems is that, in its authorization process, GSI does not follow the principle of least authority (POLA), also known as the principle of least privilege. The gridmap file authorization model is coarse-grained, and in most cases allows the users and the user's delegates more rights than necessary for their jobs. As a consequence, the effect of compromises is more severe than needed [15].

Authorization frameworks, such as the Community Authorization Service (CAS) [18, 19, 26], VOMS [1], Akenti [25], PERMIS [2] and PRIMA [14] provide a set of solutions to some of these problems. Their ability to manage fine-grained access control can limit the risks. Nevertheless, the resource providers are still required to account for the identity and activities of each "user".

Peer-to-Peer (P2P) technologies, such as remote file-sharing systems, present us with another use-case for collaboration at the other end of the spectrum. In these "Grids", users provide resources and services to anonymous clients in exchange for similar access from other users. The only security guarantee the service providers have is the vague assurance that the service they are providing is limited access to network bandwidth and personal file space.

1.2 Goals and Accomplishments

Given that many scientists and educators have access to powerful desktop systems and small servers which they control and are either open to the Internet, or can be made visible via a proxy server, it is natural to consider the problem of running a user-level, peer-to-peer collaboration Grid that allows a group of people to share access to a collection of research resources. Many collaboration activities already operate in this manner, but they use ad-hoc security models.

This paper describes an extension to the Grid Security Infrastructure that exploits the convergence of Grid standards with Web services standards. It enables one to build peer-to-peer Grids with reasonable security and that are scalable and dynamic in structure. The goal of this work is not only to provide a fine-grained authorization solution to Web services in Grids, but also to hide the security issues from the regular Grid users as much as possible, and to give the resource providers and administrators the flexibility and convenience to enforce and modify their authorization policies for the resources at runtime with minimal efforts.

The systems we describe have the following characteristics:

1. Grid resources are made available to users through *capabilities*: signed tokens that allow specific users limited access to specific remote resources. These capability tokens can take the form of documents stating that user X has the rights to interact with service A and the document is signed by Y, the provider of service A.
2. The types of services that are provided to users in this infrastructure are Web and Grid services. For example, the interface to remotely execute a specific application, or an interface to push or pull data from a specific stream, or an interface to see a directory of objects or other services. It is *never* assumed that the user has an account on the computers upon which these remote services execute. Remote services execute under the identity of the service provider. Typically this service provider identity is the identity of the scientist or engineer responsible for running or maintaining the service on behalf of his collaborators.
3. The entire infrastructure is built on a P2P chain-of-trust model: The extend of the capability that I am willing to provide to an unknown remote user is limited by the level of trust I have in providing access to that user. I demand that any user accessing my resource will present a capability token signed by me and that the community authority has authenticated the user that is presenting the capability. If the service that I am providing requires the use of other services to do its job, I am responsible for assuring those service providers that the capability level that I provide to third parties is within the bounds of the capability provided to me by them.

With the problems and goals described, we will first introduce the basis of the capability model in the following section. Then we will discuss several existing authorization systems in Grids. In section 3, our capability-based authorization infrastructure, XPOLA, will be presented in both macroscopic and microscopic views. Next, the application of our capabilities framework is discussed as it applies to our Web services framework, Grid services framework, Grid portals, and application factory services. Lastly, we will list a set of challenges and the corresponding work to do in the second phase.

2. THE CAPABILITY MODEL AND RELATED WORK IN GRIDS

2.1 Access Control Matrix, ACL and Capability

The idea of capability originates from Dennis in 1965 [4]. A capability is an identifier that carries a set of specific access permission policies on the referred objects.

	Resource 1	Resource 2	Resource 3
Alice	**Yes**	**No**	**No**
Bob	**Yes**	**No**	**Yes**
Carol	**No**	**Yes**	**No**

Table 1 An Access Control Matrix

Illustrated in Table 1 is an access control matrix, which shows all the access rights of the users to a set of resources. If we describe the table in the column order, we call it an *access control list* (ACL). For instance, the ACL of "Resource 1" covers Alice and Bob, but not Carol. If we describe the matrix by row, then each row holds the capabilities of a user. For example, the user Bob has the capabilities to access "Resource 1" and "Resource 3", but not "Resource 2". A fine-grained authorization model may choose either way when describing its access control policy definitions.

Coupled with the principal of least privilege, the capability model has the advantage over ACL in that it solves the *Confused Deputy* problem, of which ACL is incapable, as proved by Hardy [10]. The "deputy" here is a program that is being executed on behalf of other programs or people with appropriate permissions. The Confused Deputy problem happens when the program is somehow fooled to apply its permissions to something that it shouldn't do. One of the reasons is the "ambient authority" issue, which means the program need not and cannot name the specific authority that justifies the operations that it takes to sense or modify the world around it.

An old example from the Confused Deputy problem is that a printing program that audits the users' printing activities by outputting the printing information to a log file, purposely, is fooled to overwrite some important system file such as /etc/passwd, which leads to a security hole that allows attackers break in. The problem lies in the fact that the printing service works under two different authorities. One is representing the user to print his files; the other is for the system or system administrator to record users' printing activities. Under each authority, it is able to do anything more than what the user or the system administrator means. ACL does not solve the problem because if it looks up in the ACL, the program that is representing the administrator does have the authority to write /etc/passwd. However, in capability model, the deputy would have been given a set of capability tokens specifying what is allows to do with the printing service and the logging file. Without being granted the capability to write /etc/passwd, it will simply be denied by the system.

The association of access control with objects can be extended by defining a capability as the property of a user or process to carry out a particular operation. This concept was first used in computer architectures such as the Cambridge CAP, the Hydra operating system for C.mmp, and the Intel iAPX 432 architecture (see [13] for an excellent survey). Capabilities have also been embedded in programming languages, such as E [16], and they seem to be especially important for distributed systems. Furthermore, capabilities have been applied to the design and implementation of operating systems like EROS [21].

The Web and Grid services are loose-coupled, highly distributed systems. The different clients and services may be part of different trust domains without any initial trust relationships. The process of trust establishment, unlike the ones in the operating system level security, requires minimal costs on communication as well as service load and maximal flexibility on authorization policy administration. The capability model externalizes the authorization process from the service, and thus reduces a large portion of its load.

For those reasons, many Grid deployments could benefit from the capability-based model. The fact that each Web service has a standard definition document with all its operations, parameters, identifier, and location, allows one to specify detailed authorization policies that could be protected by cryptographic means. HP's E-speak was the earliest capability-related effort on Web services [24]. Unfortunately, Lacking of commercial success, its development was halted in 2001. In the Grid community, authorization frameworks like CAS and VOMS, have also adopted a capability model explicitly or implicitly, as we will discuss in section 2.3.

2.2 ACL-based Authorization Infrastructure in Grids

A cross-domain fine-grained authorization model usually comprises of the nodes of clients, resources, and authorities. In a typical ACL-based model, the authority is commonly on the resource side and part of the same administrative domain as the resources. The resource provider often has no control over the authority. In order to configure the authorization policy, the resource provider collaborates with the administrators for the users. Usually this process is not automated, can be cumbersome, and may involve many steps.

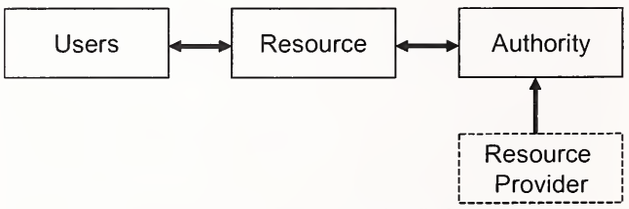


Figure 1 ACL-based Authorization Infrastructure

For now, we assume that the user has acquired her certificate and an account on the remote machine. After receiving a client's resource request, the resource node needs to verify the client's identity, after which it forwards the client's identity to the authorization authority for an access decision concerning the targeted resource. This decision will either permit or deny the client's request. PERMIS and Akenti have adopted the ACL-model that works in the flow illustrated in Figure 1 (note that Akenti can work in a capability-mode too).

Within this ACL-based model, the resource node is explicitly responsible for authenticating and authorizing

every single request. When the requests are repeated in a sequential manner from the same client, the authentication and authorization processes are thus repeated, which makes this system susceptible to scalability problems and possible Denial of Service (DoS) attacks. Akenti uses caching to mitigate the problem, but it hurts those isolated requests at the same time.

2.3 Capability-based Authorization Infrastructure in Grids

Another category of fine-grained authorization frameworks works with a capability model. In such infrastructures, the client's possession of a capability confers an access right to a resource, according to the pre-defined policy embedded in that capability.

The workflow is as follows: the user first sends a capability request to the authorization authority. The resource provider could either approve the request or delegate his rights to the authority to issue the requested capability tokens. Subsequently, the client sends her access requests along with the capability tokens to the resource node. On the resource node, the capability token's integrity is first verified. The policy details of the capability are then extracted as the reference to the final authorization decision. Figure 2 shows a simplified diagram of the capability model.

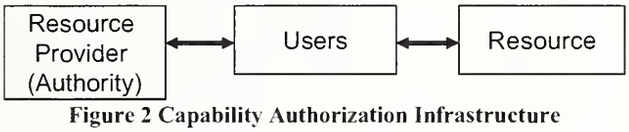


Figure 2 Capability Authorization Infrastructure

Besides the advantages on solving the Confused Deputy problem and externalizing authorization administration as we mentioned in section 2.1, the capability-based authorization infrastructure also has the benefit of reusability. The issued capability tokens are reusable for multiple accesses as long as their policy allows. It saves the repeated authorization evaluation processes that are inevitable in ACL-based systems.

A more practical reason is that some of the Grid users, especially those scientists, have the will to build their extemporaneous experiment services and allow others to use them securely without the lengthy interference of their system or Grid administrators. Capability allows them to distribute their services in a peer-to-peer fashion and remove the system security concerns that hinder their research.

CAS is an example of a capability-based authorization service. In CAS, a user must present a capability issued by user's community's authority to the resource provider to access the resource. The CAS capability token is bound to an extended proxy credential. Instead of the resource provider issuing the capability directly, the provider delegates part of his authorities to a community administrator. Every community has at least one central administrator to define the authorization policies and manage the service. The service authenticates the user with his X.509 certificate and issues capabilities to authorized users. On the resource server side, with the local policies, the resource provider still makes the final authorization decision, even if the user has been given the permission by the CAS server. However, the resource server will allow no more than what the capability allows.

Even though the CAS framework supports fine-grained authorization, resource providers have to verify that the CAS server is to be trusted and that the community's policies match their own local policies through some out-of-band form of configuration, which is not applicable in the case of highly dynamic services like the impromptu experimental ones that are provided by the scientist users.

Moreover, any form of centralized administration presents a single point of failure when being compromised or under attacks. One proposed solution of having multiple replicas of CAS servers to address the single-point-of-failure issue will statistically bring more chances of being compromised. Note that the capability model itself is not perfect either. Capability revocation is a thorny problem in CAS as well as any other capability-based systems.

Other fine-grained Grid authorization infrastructures are similar to CAS or Akenti. The different approaches can be distinguished in the way they bind policies to certificates and their authorities, while the enforcement mechanisms may differ slightly.

As we will show in the next section, XPOLA tries to keep the advantages of the capability model, while solving most of the issues associated with the capability model and other fine-grained authorization efforts. The main difference from any other existing capability-based infrastructures including CAS is its peer-to-peer administration model that securely brings maximal freedom to scientists and researchers. In the extreme case that all services are provided by one single user or a service account, that user becomes the administrator, just like the role in other infrastructures.

3. XPOLA, THE MACROSCOPIC VIEW

3.1 The Big Picture

The name of XPOLA stands for an eXtensible fine-grained authorization infrastructure that strictly conforms to the Principle of Least Authority (POLA). The design picture of XPOLA is shown in Figure 3. The capability manager (Capman) is the platform for resource providers to collaborate with their users. Through this platform, the resource users send requests to the providers for access rights; the provider processes the requests and creates the corresponding capability tokens. The capability manager stores the capability tokens and supports the providers with the functionalities for manipulating the capabilities: capability generation, destruction, update, renewal, request processing, and optionally pushing the capabilities to the users.

The registry service provides registration and discovery of the available Web service instances. It keeps the information of all the registered service instances. Every time a new service is instantiated, it is supposed to register itself by sending a Web Services Definition Language (WSDL) document to the registry. A WSDL document contains the detailed information needed to interact with the service instance.

If the capability of a specific resource for a specific user exists in the capability manager's storage, the user can fetch the capability from the manager directly and stores it in her local token agent. The token agent is an ssh-agent-like client for interacting with the capability manager and caching the retrieved capability tokens.

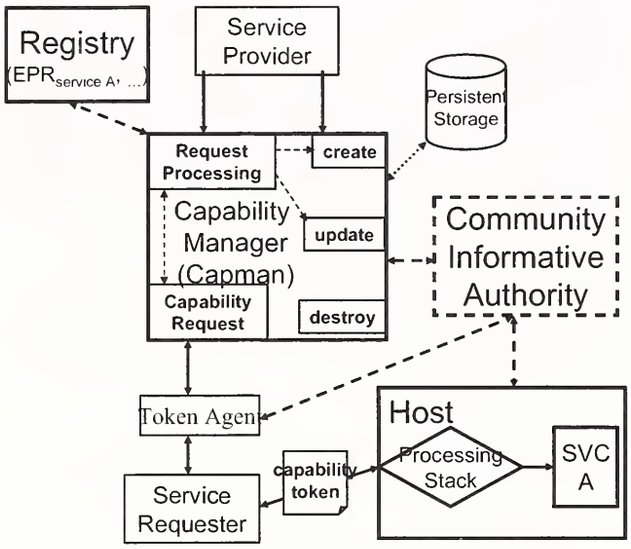


Figure 3 The Big Picture

The Community Informative Authority (CIA) is an optional trusted authentication service for identity verification in the community. It establishes trust relationships, either mutual or unilateral as required, between the users and providers. Note that CIA service does not make any authorization decisions for the providers. It is left to the providers themselves to make their authorization decisions based on the provided authentication information. CIA is not necessary if two parties can mutual authenticate each other by other means, for example, through the portal login authentication of a trusted portal. No matter whether it works implicitly or explicitly, the CIA service is designed to address the trust bootstrap problem in an XPOLA-enabled Grid community.

To make it better understood, let's walk through one of the use scenarios as a provider, *X*, of service *A*, and a service *A*'s user, *Y*. Suppose *X* first creates the service *A* on a remote host. Upon being created, the service A registers itself to a persistent registry. The provider then wants to distribute the service to other people including the user *Y*. Through a Capman service, he generates a set of capability tokens which contain detailed authorization policy and protected by *X*'s signature. We will dissect the capability token later, but here we just need to know that the identity of *Y* is included in the policy as one of the users. Now *X* advertises his service to *Y*, who will to use *A*. When *Y* tries to access *A*, his token agent contacts the Capman service for the required capability token. If available, the token is fetched and sent along with the service request to *A*, where the capability is to be verified. Thus the user *Y* is served by *A*, if the request matches what the capability allows.

3.2 Attack Analysis

Figure 3 can be simplified as Figure 4 to reflect the trust relationships between the main entities, namely the capability authority, the clients, the service instances, and an optional CIA service.

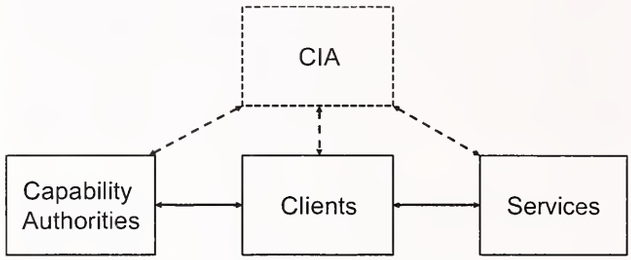


Figure 4 A Simplified Trust Relationship Picture for Attack Analysis

The service might be subject to various forms of illegal accesses and attacks from the clients. Examples of illegal or malicious access attempts are:

1. The user does not have a capability for the remote resource, or has a fake or invalid capability.
2. The capability is valid, but the user applies it beyond its policy definition. For instance, the user tries to access some other resource that is not specified in the capability policy.
3. The access is authorized, but the user orchestrates a Denial of Service attack (DoS) with the valid capabilities.

The first two situations can be handled easily by executing the capability enforcement. The last one is a tricky problem for capability-based systems. We need to revoke the malicious users' capabilities at runtime while the DoS attack takes place.

The capability manager does not need much protection as the capability tokens are inherently protected by the signatures from being forged or tampered. They are totally useless to any user who steals them, as the attackers' identities are not specified in the policies, unless a valid user's private key is available at the same time.

The worst case could happen is when the private key of a provider is stolen; however we can confine the consequences of this compromised authority to the provider

himself, who is merely one of the non-privilege users. Note that the providers won't be able to blame the administrators, because they themselves are responsible for the security of their own resources. XPOLA is flexible in that under some circumstances, administrators can take over the authorization work by running services with special non-privilege service accounts.

4. XPOLA, THE MICROSCOPIC VIEW

4.1 The Capability Tokens

A capability is a policy definition referring to a specific resource object. In XPOLA, a policy definition comprises of a delegation relationship and a set of authorization decisions for the referred object. The capability is protected with signatures and can be encrypted if needed.

Formally we define a capability as follows. Assume a resource provider named X creates a capability C to delegate a set of rights R on the service identified as S to a group of users P after the time of T with the duration Δt. The capability C_X is defined as

$$C = [X, P, R, S, T + \Delta t]$$

XPOLA has adopted the Security Assertion Markup Language (SAML) to express C, but it could be any policy languages, as long as the recipient, who himself set up the policy in most cases, is able to understand it. XPOLA is extensible to use policy definitions expressed in different formats or languages.

The policy definition for the operations of Web services is specified according to their Web Services Definition Language (WSDL) documents. The WSDL document is the standard interface description of a Web service.

The policy must contain the following items, corresponding to the formal definition:

- Parameter X: One or a set of the resource providers' X.500 distinguished names (DN) from their X.509 certificates.
- Parameter P: One or a group of specific users' DNs.
- Parameter R: A set of authorization decisions corresponding to the concerned operations of S.
- Parameter S: A service identifier, which is an endpoint reference (EPR) of a specific Web service.
- Parameter T: The lifetime of the capability.

Furthermore, the assertion will include signatures and verification methods. When privacy is a concern, the capability should be encrypted.

Within the Web services framework, when a user makes secure requests to remote resources, her client program communicates with them by exchanging SOAP messages that comply with Web services security specifications. Web services security is a series of emerging XML-based security standards from W3C and OASIS for SOAP-based Web services. The security related elements are added to the header section of SOAP messages, including signatures, references, confirmation methods, canonicalization methods, etc.

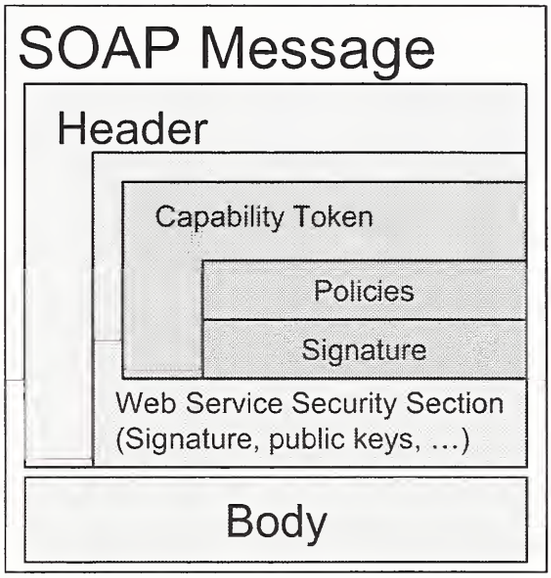


Figure 5 A Capability Bound to a SOAP Message

Similarly, capability tokens are embedded into the header section of users' SOAP messages, as showed in Figure 5. The SOAP message is then signed with the user's private key and the generated signature inserted into the Web service security section to protect the integrity of the SOAP message, along with the user's public key and identity, which will be verified against the capability token later.

4.2 The Capability Enforcement in Web Services

The enforcement is done when the capability-enabled SOAP message arrives at the remote resource service. We will identify the actual resource owner as A^*, the actual resource with the identifier S^*, the actual resource access attempt E that happens at the time T^*. The original capability C_A arrives at the resource as

$$C' = [X', P', R', S', T' + \Delta t']$$

The final authorization decision is made upon

$$C = C', X = X^*, S = S^*, E \subseteq R, T^* < T + \Delta t$$

and

$$\prod_{j \in C, C \subseteq R} r_j \sum_{r_i \in R} r_i \geq F_s. \quad r_i \in R, r_i \in \{0,1\}$$

Here, F_s is the authorization threshold of the resource S and $F_s = 1$. When $r_i = 0$, it means "denied"; while $r_i = 1$ means "granted". The set C is the crucial authorization decisions in R, which means any element r_j in the set C must be

"granted", in order that the access be authorized. The above process can be generalized as $r_i \in [0,1]$ and $F_s \geq 0$, when the provider is not so sure about his decision.

In XPOLA-enabled Web services, when a user makes a remote call from the client side, the client program first performs a preliminary authorization check by matching the capability policy with the SOAP body content and the user's identity. If nothing violates the policy, it inserts the appropriate tokens into the SOAP header, including the signatures; otherwise, the client is refrained from invoking the request.

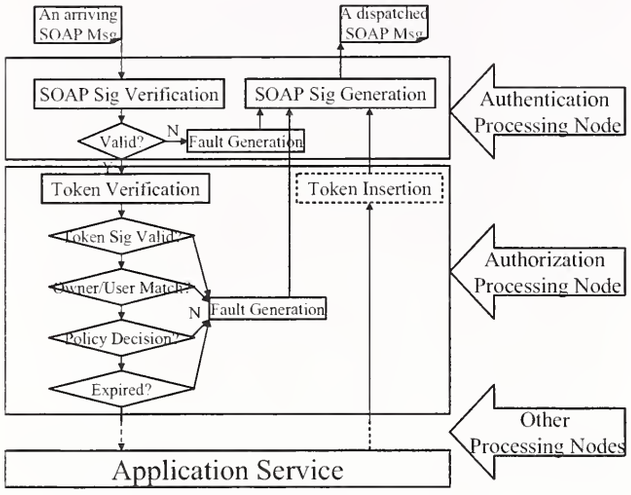


Figure 6 The Processing Stack on the Service Side

On the resource side, when the Web service receives a remote call through a SOAP message, the processing node first authenticates it by checking the validity of the user's signature against the whole message. If nothing is wrong, the authorization processing node verifies the signature of the embedded capability token. It then does a series of matching operations, such as the capability issuer's DN and the actual resource provider's DN, the actual caller's DN and the allowed users' DNs, the operations specified in the capability and the ones in the SOAP body. It also checks whether the capability has expired against the time. At last, it peels off the capability token and passes the rest of the SOAP message on to the next processing node. The detailed diagram is showed in Figure 6.

The client side policy checking prevents those unauthorized requests from reaching services and is able to prompt appropriate error information immediately to the users. Services thus need not waste their resources on processing these requests. The capability checking at the service side is mainly for guarding services from malicious users who elude the provided authentic clients to send the invalid requests directly. Such attack scenarios are discussed in section 3.2.

5. THE XPOLA IMPLEMENTATION AND APPLICATIONS

5.1 The Implementation

The implementation work includes the development of the core package of capability tokens with an application programming interface (API), the capability management toolkits, and their applications.

The capability's policy core adopts the popular XML-based security language, Security Assertion Markup Language (SAML) to express policy definitions [31]. SAML addresses three different use-cases.

1. **Single Sign-On.** The ability of a user (or subject in SAML terms) to authenticate in one security domain and have that authentication respected in another domain.
2. **Authorization.** The ability to ask questions about and describe the authorization of an actor to access a resources.
3. **Transactions.** The ability to pass the authority to complete a task from one security domain to another.

At its most basic level SAML is a language for making assertions about authentication, attributes, conditions and authorization decisions. For example, an authentication assertion typically takes the form "subject *S* was authenticated by means *M* at time *T*." Attributes are simply name-value pairs associated with facts about a subject. Authorization decisions take the form "It has been decided that subject *S* is authorized to perform action *A* on resource *R*. SAML provides a simple protocol to ask questions like "What authorization assertions are available for this subject?", "What are the values associated with these attributes for this subject?", "Is this subject allowed to access this resource in the following manner?".

In SAML terms, each capability policy document is a standard SAML assertion. The capability tokens are signed with the provider's private key. The public key is attached for verification.

XPOLA is designed with extensibility in mind. If needed, we can plug-in other XML-based policy languages such as eXtensible Access Control Markup Language (XACML) [32], or eXtensible rights Markup Language (XrML) [33]. The bottom line is, the provider can use any policy language he wants as long as he understands his own policies. Neither does XPOLA mandate the use of PKI to protect the capabilities. The provider may use symmetric keys to sign the capabilities, though in that case he needs some other ways like Kerberos, to authenticate the users and assertions.

XPOLA relies on XSUL to bind the capabilities to SOAP messages and to enforce the capability-based authorization. XSUL is a light-weighted SOAP engine and a general Web services framework [23]. Web service developers can write their own capability-enabled Web services with XSUL with limited effort. We have also integrated the XPOLA into GSX for building capability-enabled Grid services. GSX is an Open Grid Services Infrastructure (OGSI) [30] and Web

Services Resource Framework (WSRF) –compliant [34] Grid service framework based on XSUL [20]. The capability-based Grid service was assigned as homework in a distributed computing class. Turning an insecure Grid service into a capability-enabled one, was so simple that few student ever complaint.

The fact that our capabilities are inherently protected by themselves, allows us to use alternative ways to distribute them even through unprotected means. For example, the resource provider can use email, shared file system or even fax to deliver capabilities to the user.

To help the user, we provide a portlet-based capability manager and token agent build on the core capability APIs. We believe a user-friendly human-computer interface is vital in any security infrastructure. Many security systems fail, commercially or technically, simply because the users would not or could not follow the complicated rules. The capability manager as an independent Web service that shares the same database is also available.

5.2 XPOLA in Grid Portals

Grid portals are becoming very important in Grid deployments. They provide Grid community with a user friendly and familiar web-based interface to access their Grid resources. A Grid portal is made up of a cluster of customizable display units, called portlets. A dynamically generated web page in a Grid portal is the aggregation of a group of portlets in a personalized layout. The Grid portlets could act as the clients of the remote Grid services.

A Grid portal provides its own authentication mechanism through portal accounts bound with the users' proxy certificates. Both resource providers and users, as the portal users, share the same trusted portal context.

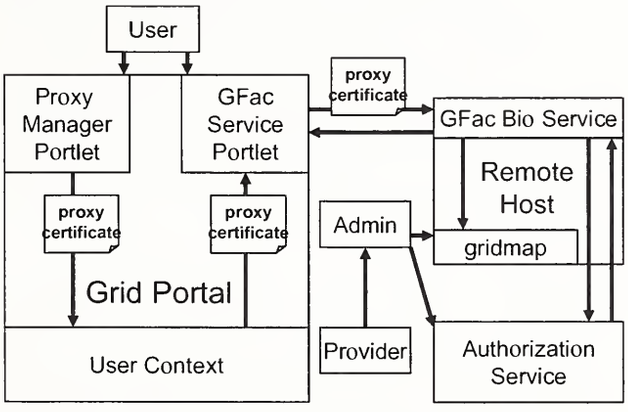


Figure 7 Authorization in an ACL-based Grid Portal

Traditionally, when a user logs into a Grid portal with her account, the portal requests a proxy certificate from a credential server, such as the MyProxy server [17], which returns a proxy certificate under the request. Usually the proxy certificate has a very short period of lifetime. The portal server stores the proxy certificate in the user context. While the proxy certificate is valid, the portal can fetch it when the proxy certificate's owner, the portal user, wants to invoke a remote Grid service through a Grid portlet. The proxy certificate goes along with the remote service call to authenticate the user, according the gridmap mechanism. The authorization steps, if available, are done in the ACL style, as shown in Figure 7.

One example of a Grid portal is the GFac portlet that works as the client of a GFac service. The GFac service is a Grid-enabled application factory service that encapsulates non-Web services, launches them on remote hosts at run time, and presents them as Web service instances [8]. GFac has been applied to launch complicated jobs remotely in the scientific domains of biology and meteorology.

Originally, GFac was a typical non-capability-based Grid service. After a GFac Bio service is launched remotely by its provider, a Bio service user can access the service from the Grid portal by contacting the remote service with her proxy certificate stored in her portal user context. On the remote host, her Grid identity is first mapped to a local account, where the authorization decision lookup is done by the GFac Bio service. With no choice, the GFac service provider has to delegate his authority to the administrator such that he can pre-configure the gridmap file and the authorization service, before allowing the designated users to access his service.

The XPOLA integration moves GFac's authorization process to the external Grid portal and avoids the administrator's authority brokering. After launching the service, the provider creates capabilities with capability manager portlet and stores them in portal user contexts. Later, when a portal user logs in, he can simply access the remote Bio service resource through the portlet client. The portlet automatically fetches the required capabilities from the user's context if they are available. The remote service authorizes the user according to the capabilities it receives.

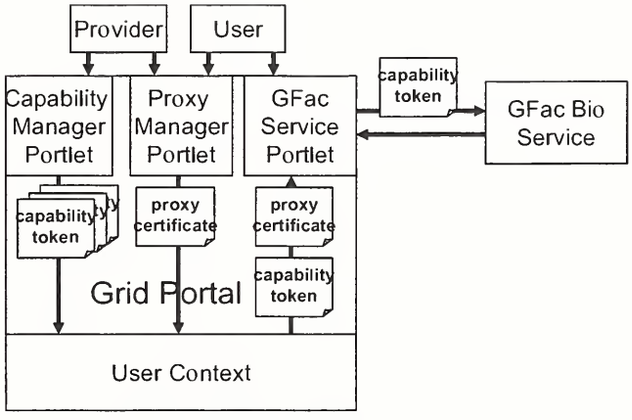


Figure 8 Authorization in an XPOLA-enabled Grid Portal

Because Grid portal is a trusted domain for all users, it implicitly plays the role of CIA by authenticating users with their portal accounts. Grid portal makes it possible to

transparentize the authorization process to portal users, as illustrated in Figure 8.

To summarize the scheme depicted in Figure 8: in a capability-enabled GFac service, the provider first launches a Bio job remotely through a GFac portlet. Then he simply creates a series of capability tokens for those authorized users with the capability manager portlet and stores them in the user contexts. After that, he can invite the users who have been endowed with the capabilities to use his service. The authorized user logs into the portal and she will be able to access the remote Bio service with the capabilities automatically fetched from her user context in the Grid portal; while as a naïve user, she may never know the existence of capabilities.

5.3 Performance

The capability-based authorization infrastructure is efficient as it allows for the reuse of capabilities and user-level administration. An authorization decision, once made, can be reused for multiple times.

However, if we only look at the narrowest definition of performance, it takes about 1000 to 2000 mille-seconds for a roundtrip communication between a client and a service, as shown in Figure 9. As the number of invocations grows, we see a slightly better throughput, probably due to the internal optimization of Java Virtual Machine itself, but it still falls in the same range. Considering the bulky SOAP header with security-related SAML assertions, signatures, and public keys, we are not too surprised about these observations. The reason seems to be a general problem of the implementations on Web services security nowadays: the underlying XML-related operations and processing, especially the canonicalization operations, are very expensive [22].

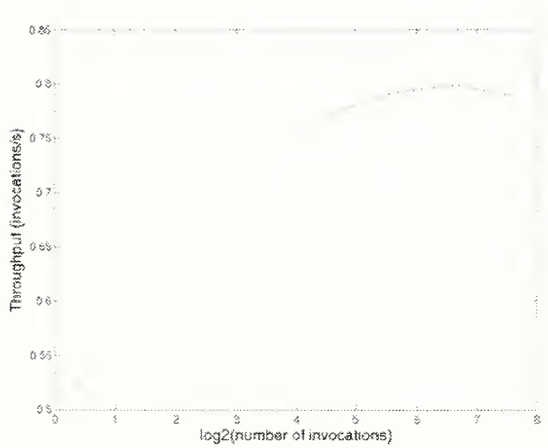


Figure 9 Throughputs of an XPOLA-enabled Web service

6. CHALLENGES AND FUTURE WORK

6.1 Session-based Communication

Performance is a big challenge in XML-based security implementations. Fortunately there are often possible workarounds to address the high overhead of the XML processing. The initial results of some informal tests, gives an indication that a session-based communication may be a promising solution. With session-based protocols integrated in our communication stack, we expect them to lower the XML security processing overhead significantly. In the second phase of our XPOLA work, we plan to implement a session-based secure communication specification, based on WS-Trust, WS-SecureConversation, and on the implementation work that was previously done [5].

6.2 Revocation

Revocation is a challenge for all capability-based systems. After the users acquire the capability, the provider does not have any effective approach to revoke it. In most capability-based systems, a capability's lifetime is so short that revocation is not practical, while the short lifetime may also help limit the amount of damage in the case of compromise.

One possible solution that we are investigating is to tie the capability to established sessions. This would allow us to revoke a specific capability immediately, by simply invalidating the underlying session.

6.3 Denial of Service Mitigation

Capability itself is a good mechanism for dealing with Denial of Service (DoS) attacks for providing fine-grained authorization. One of the principals of DoS attack prevention and mitigation is using the cost model – a client has to pay proportionally to the amount of the accesses to a service. A capability can be regarded as a resource "currency note" to be used to pay for the service it refers to. Capability tokens also make it possible to load balance the authentication and authorization work to other machines. A more sophisticated method will be to issue or request for dynamic capability tokens in the situation of an overwhelmed service, which would match the session-based capability system design.

7. CONCLUSIONS

In this paper, we introduced an efficient and user-friendly capability-based authorization infrastructure, XPOLA. It is based on user-level, peer-to-peer trust relationships, and used for building secure Web services and Grid services with the support of fine-grained authorization policy enforcement. We presented both microscopic and macroscopic views of XPOLA, and discussed its applications. Lastly, we brought up some challenges and tentative solutions that are to be addressed in future work.

8. ACKNOWLEDGMENTS

We would like to thank Gopi Kandaswamy, Aleksander Slominski, and Yogesh Simmhan for their discussions and collaboration on the integration work of the factory service, XSOAP and GSX. We also appreciate Markus Jakobsson for his invaluable suggestions to improve the paper.

9. REFERENCES

[1] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lorentey, and F. Spataro. VOMS, an Authorization System for Virtual Organizations. In European Across Grids Conference, February 2003.

[2] D. W. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure. *Future Generation Comp. Syst. 19(2), pp. 27-289, 2003.*

[3] C. Catlett. TeraGrid: A Primer. http://www.teragrid.org/about/TeraGrid-Primer-Sept-02.pdf.

[4] J. B. Dennis and E. C. Van Horn. Programming Semantics For Multiprogrammed Computations. MIT Technical Report MIT/LCS/TR-23, 1965.

[5] L. Fang, S. Meder, O. Chevassut, and F. Siebenlist. Secure Password-based Authenticated Key Exchange for Web services. *ACM Workshop on Secure Web Services, Oct.29, 2004, Fairfax, VA.*

[6] I. Foster, C. Kesselman. *Intl J. Supercomputer Applications, 11(2):115-128, 1997.*

[7] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A Security Architecture for Computational Grids. *Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.*

[8] D. Gannon, S. Krishnan, L. Fang, G. Kandaswamy, Y. Simmhan, and A. Slominski. On Building Parallel & Grid Applications: Component Technology and Distributed Services. *In CLADE 2004, Challenges of Large Applications in Distributed Environments. IEEE Computer Society Press, Jun 2004.*

[9] D. Gannon et al. Building Grid Portal Applications from a Web Service Component Architecture, 2004.

[10] N. Hardy. The Confused Deputy, http://www.cap-lore.com/CapTheory/ConfusedDeputy.html.

[11] T. Hey and A. E. Trefethen. The UK e-Science Core Programme and the Grid, *Future Generation Computing Systems, Vol 18 page 1017, 2002.*

[12] W. Johnson, D. Gannon, B. Nitzberg. Grid as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid, HPDC 1999.

[13] H. Levy, Capability-based Computer Systems, Digital Press, 1984, now available at http://www.cs.washington.edu/homes/levy/capabook/

[14] M. Lorch, D. B. Adams, D. Kafura, M. S. Koneni, A. Rathi, and S. Shah. The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments. *Proceedings of the IEEE 4th International Workshop on Grid Computing, 2003.*

[15] K. Keahey, V. Welch. Fine-Grain Authorization for Resource Management in the Grid Environment. *Proceedings of Grid2002 Workshop, 2002.*

[16] M. Miller, M. Stiegler, Open Source Distributed Capabilities, http://www.erights.org/index.html.

[17] J. Novotny, S. Tuecke, V. Welch. An Online Credential Repository for the Grid: MyProxy. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.*

[18] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A Community Authorization Service for Group Collaboration. Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[19] L. Pearlman, C. Kesselman, V. Welch, I. Foster, S. Tuecke,The Community Authorization Service: Status and Future, *CHEP03, March 24-28, 2003, La Jolla, California*

[20] Y. L. Simmhan, Grid Service Extensions, http://www.extreme.indiana.edu/xgws/GSX/

[21] J.S.Shapiro, J.M.Smith, and D. J. Farber. EROS: A Fast Capability System. SOSP, 1999.

[22] S. Shirasuna, A. Slominski, L. Fang, and D. Gannon. Performance Comparison of Security Mechanisms for Grid Services, *the 5th IEEE/ACM International Workshop on Grid Computing,* Pittsburgh, Nov. 8, 2004.

[23] A. Slominski, M. Govindaraju, D. Gannon, and R. Bramley. Design of an XML-based interoperable RMI system: SoapRMI C++/Java 1.1. *In Proceedings of the 2001 International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV, Jun 2001* http://www.extreme.indiana.edu/xgws/xsoap/

[24] Spring, Buranarach, Schrubb, and Zatuchnaya. E-speak Revisited. *Forum on Design Languages Sep 3-7, 2001, Lyon, France.*

[25] M. Thompson, et al. Certificate-based Access Control for Widely Distributed Resources. In Proc. 8th Usenix Security Symposium. 1999.

[26] V. Welch, R. Ananthakrishnan, S. Meder, L. Pearlman, F. Siebenlist, Use of SAML in the Community

Authorization Service, http://www.globus.org/security/CAS/Papers/SAML%20Feedback-aug19.pdf

[27] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke. Security for Grid Services, *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press, June 2003.*

[28] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, F. Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. *3rd Annual PKI R&D Workshop, 2004.*

[29] Open Grid Services Architecture, OGSA working group at Global Grid Forum (GGF), https://forge.gridforum.org/projects/ogsa-wg

[30] Open Grid Services Infrastructure, OGSI working group at Global Grid Forum (GGF)

[31] OASIS, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1 http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf

[32] OASIS, Core Specification: eXtensible Access Control Markup Language (XACML) V2.0 http://docs.oasis-open.org/xacml/access_control-xacml-2_0-core-spec-cd-02.pdf

[33] ContentGuard, eXtensible Rights Markup Language (XrML) 2.0 http://www.xrml.org

[34] OASIS, "Web Service Resource Framework", March 2004.

Design and Implementation of LDAP Component Matching for Flexible and Secure Certificate Access in PKI

Sang Seok Lim
IBM Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
slim@us.ibm.com

Jong Hyuk Choi
IBM Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
jongchoi@us.ibm.com

Kurt D. Zeilenga
IBM Linux Technology Center
Carson City, NV
zeilenga@us.ibm.com

Abstract

Lightweight Directory Access Protocol (LDAP) is the predominant Internet directory access protocol and hence so is its use in the Public Key Infrastructure (PKI). This paper presents the design and implementation of LDAP component matching which enhances flexibility and security of the LDAP directory service when it is used for the PKI certificate repositories. The component matching together with the prerequisite ASN.1 awareness enables matching against arbitrary components of certificates and enables matching of composite values at the abstraction layer of the underlying ASN.1 type definition. This allows searching for certificates with matching components without the need of providing syntax specific parsing and matching routines (flexibility), without the need of extracting the certificate components and storing them into separate attributes which become searchable but mutable (security), and without the need of restructuring Directory Information Tree (DIT) to support multiple certificates per subject (manageability and performance). In this paper, we describe the architecture, key data structures, and the proposed methods of enhancing interoperability and performance of our component matching implementation in the OpenLDAP open source directory software suite. We also propose the use of component matching in on-line certificate validation and in Web services security. Through performance evaluation of the OpenLDAP component matching, we show that our LDAP component matching implementation exhibits the same or higher performance compared to the previous approaches.

Keywords

PKI, X.509 Certificate, Certificate Repository, Component Matching, LDAP

1 Introduction

The certificate repository in Public Key Infrastructure (PKI) is a means of distributing certificates and Certificate Revocation Lists (CRL) to end entities. It stores certificates and CRLs and provides efficient access methods to them by harnessing storage means with communication mechanisms. The directory technology stands out as the most befitting approach to implementing certificate repositories because the X.509 [14] PKI has been standardized in the context of the X.500 recommendations as the public key based authentication framework on the X.500 directory.

Lightweight Directory Access Protocol (LDAP) [10] renders lightweight directory service by providing direct mapping onto TCP/IP, simple protocol encoding, reduced number of operations, and string-based encoding of names and attribute values (hence of assertion values). However, these simplifications come at a price. Because the string-based encoding in LDAP generally does not carry the complete structure of abstract values, adding support for new syntaxes and matching rules requires ad-hoc developments of syntax parsing and matching routines. X.500 protocols, on the other hand, avoid this problem by use of ASN.1 (Abstract Syntax Notation One) [13] encoding rules, in particular, the Basic Encoding Rules [12].

Though these limitations were not viewed as a significant problem during LDAP's early years, it is clear that a number of directory applications, such as PKI, are significantly hampered by these limitations. For instance, in PKI, a certificate needs to be located based upon the contents of its components, such as serial number, issuer name, subject name, and key usage [14]. LDAP search operations do not understand ASN.1 types in the definition of the certificate attribute and assertion [14], because attributes and assertions in LDAP are encoded in octet string with syntax specific encoding rules. Not only would it require exceptional effort to support matching rules such as *certificateExactMatch* and *certificateMatch* as defined in [14], that effort would have to be repeated for each matching rule introduced to match on a particular component (or set of components) of a certificate. Because of the large amount of effort each server vendor must undertake to support each new rule, few new rules have been introduced to LDAP since its inception. Applications had to make due with existing rules.

Foreseeing the need to be able to add new syntax and matching rules without requiring recoding of server implementations, the directory community engineered a number of extensions to LDAP to address these limitations. The Generic String Encoding Rules (GSER) [17] was introduced to be used in describing and implementing new LDAP string encodings. GSER produces human readable UTF-8 [32] encoded Unicode [28] character string which preserves the complete structure of the underlying ASN.1 type and supports reuse of the existing LDAP string encodings. Provided that an LDAP server is ASN.1 aware, i.e. it can parse values in ASN.1 encoding rules into its internal representation of ASN.1 value and can perform matching in that abstraction layer, it is possible to support matching of arbitrary types without needing ad-hoc developments of parsing and matching routines.

The component matching [18] mechanism was also introduced to allow LDAP matching rules to be defined in terms of ASN.1. It introduces rules which allow arbitrary assertions to be made against selected components values of complex data types such as certificates. For example, the component matching enables matching against the selected components of certificates without the need to define a certificate component specific matching rule and without

requiring custom code to implement that matching rule for the certificate attributes.

Though the directory community saw GSER and component matching as an eloquent solution to the LDAP syntax and matching rule limitations, there were some concerns, as most LDAP server implementations were not ASN.1 aware, that its adoption would be slow. To fulfill immediate needs of PKI applications, another solution based upon attribute extraction (or "data de-aggregation") has been being utilized as a practical remedy. The attribute extraction method decomposes a certificate into individual components and stores them into separate, searchable attributes. Certificate Parsing Server (XPS) [2] automates the attribute extraction process. Although this approach has filled the interoperability gap between LDAP and PKI, it is considered to be not a workable solution for PKI applications (and certainly not a workable general solution to the component matching problem), because it introduced a number of security and management issues.

In the spring of 2004, IBM undertook an engineering effort to provide ASN.1 awareness (with GSER, BER, DER support) and component matching functionality for the OpenLDAP Project's Standalone LDAP Daemon (*slapd*), the directory server component of OpenLDAP Software [26]. To our knowledge, this is the first implementation of the component matching technology in a pure LDAP directory server (second to the View500 [29] directory server from eB2Bcom [8] which is X.500 based). This paper presents a detailed and comprehensive description of the design and implementation of the LDAP component matching for improved PKI support, extending our previous work [19] which had described component matching in the context of WS-Security [24]. Another contribution of this paper is that it proposes key mechanisms to improve performance and interoperability – attribute / matching rule aliasing, component indexing, and selective component caching. This paper will also present a preliminary performance evaluation result which convinces us that the performance of component matching is on par with or better than those of the syntax specific parsing and attribute extraction approaches if the optimization mechanisms proposed in this paper are used. This in fact provides a strong evidential answer to the debate in the PKI standardization community on whether the component matching technology can be implemented in LDAP directory servers timely and efficiently. This paper also discusses on the possibility of using the component matching for CRL in order to support on-line certificate status checking using LDAP. It also discusses on the feasibility of using LDAP component matching for PKI in Web services security.

This paper is organized as follows. Section 2 introduces the interoperation of LDAP and PKI and describes the deficiencies of LDAP when it is used for PKI. Section 3 describes the component matching technology and its use in PKI enabling secure and flexible certificate access. It also discusses on the possibility of certificate validation against CRL using LDAP component matching. Section 4 introduces GSER (Generic String Encoding Rules) which facilitates the ASN.1 awareness in LDAP when it represents the attribute and assertion values. In Section 5, we present the design and implementation of the ASN.1 awareness and the component matching in the OpenLDAP directory server. Section 6 demonstrates the application of the component matching for PKI to the security of Web services. Section 7 shows experimental results of our prototype implementation of the LDAP component matching and proves that the component matching can be accomplished without any loss in performance. Section 8 concludes the paper.

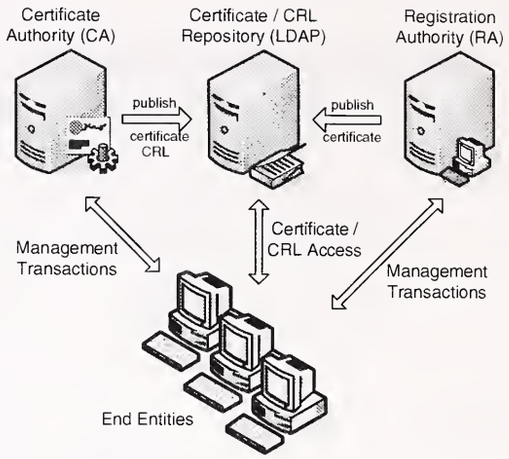


Figure 1. The Architecture of Public Key Infrastructure.

2 LDAP in PKI

2.1 LDAP Certificate Repository

X.509 certificates and CRLs are commonly distributed by the certificate repositories. LDAP directories are the most versatile mechanism of implementing the certificate repositories. Figure 1 illustrates the conceptual interoperation of four entities in PKI. In the public key registration phase, an end entity sends its identity as well as its public key to a Registration Authority (RA). If the identity is validated by the RA, the Certificate Authority (CA) will publish the end entity's certificate, storing it in the LDAP directory. After that, the published certificate can be retrieved by any properly authenticated LDAP client. If the issued certificate is revoked by any reason, the CA is responsible for revoking the certificate by publishing CRLs to the LDAP directory. LDAP directories serve as the central place where the end entities not only can download certificates of others in order to send encrypted messages or verify digital signatures but also can be informed of the latest certificate revocation information by downloading CRLs.

2.2 Deficiencies of LDAP Certificate Access

An end entity should be able to send a request to the LDAP certificate repository searching for a certificate having matched values in specific components of the certificate. As a principle example, when it wants to retrieve the certificate having a specific serial number and issued by a specific CA, it will send an assertion against *serialNumber* and *issuer* components as specified in *certificateExactMatch* of X.509 [14]. However, the need for matching is not limited only to these two certificate components. An end entity may want to search for certificates which belong to a subject. It may also want to restrict the scope of the search for the subject's certificates to those having a specific key usage, e.g. *nonRepudiation*, by using the *keyUsage* certificate extension. Because LDAP stores attribute and assertion values in LDAP-specific octet strings which do not generally preserve structural information of the underlying ASN.1 types, however, it is far from trivial to provide this component level matching in a generic and flexible way.

X.500 [15] satisfies this demand for component level matching by allowing matching to be defined at the ASN.1 layer. For instance, [14] defines *certificateExactMatch* and *certificateMatch* matching

```
(userCertificate:certificateExactMatch:=12345$o=IBM,c=US)
```

(a) Syntax Specific Parsing.

```
(&(x509SerialNumber=12345)(x509KeyUsage=010000000))
```

(b) Attribute Extraction.

```
(userCertificate:componentFilterMatch:=
  and:{
    item:{
      component "toBeSigned.subject",
      rule distinguishedNameMatch,
      value "cn=John Doe,o=IBM,c=US"
    }
    item:{
      component "toBeSigned.extension.*.
                          extnValue.(2.5.29.15)",
      rule bitStringMatch,
      value '010000000'B
    }
  }
)
```

(c) Component Matching.

Figure 2. Three LDAP Certificate Access Methods.

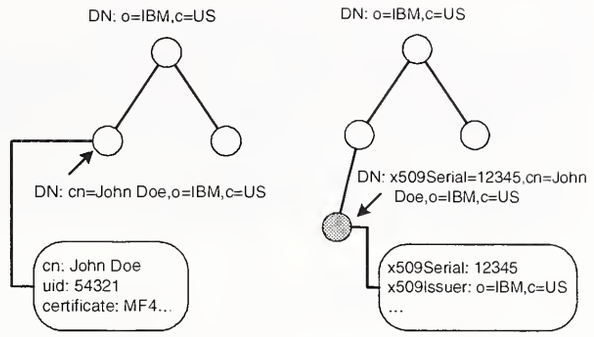
rules by specifying them in ASN.1 data type representations. The use of ASN.1 specifications is beneficial in the following respects: 1) parsing and matching can be automatically accomplished from the given ASN.1 type specification without providing ad-hoc routines; 2) simple but powerful matching rules are derivable from the strong expressive power of ASN.1, as exemplified in the use of *OPTIOANL* in *certificateMatch*; 3) new matching rules can be easily provided by specifying them in ASN.1.

The rest of this section explains how the current workarounds try to provide solution to the above mentioned interoperability gap between LDAP and PKI and introduces the component matching approach focusing on its advantages over the earlier workarounds.

2.3 LDAP Certificate Access Methods

2.3.1 Syntax Specific Parsing

A brute force approach to providing matching for arbitrary components of a certificate against an assertion is to provide certificate-syntax specific matching rules. For example, it is possible to manually write a special matching routine that matches the certificate attribute against the assertion value consisting only of *serialNumber* and *issuer* to implement *certificateExactMatch* which, in case of X.500 [15], is meant to be derived automatically from its ASN.1 specification [14]. In OpenLDAP, *certificateExactMatch* is implemented by using certificate decoding libraries provided by OpenSSL [27]. Figure 2 (a) shows an example filter in which the predetermined token '$' is used to separate the serial number 12345 and the issuer distinguished name `o=IBM,c=US`. The server can recognize the serial number and issuer name by reading two strings separated by '$'. The downside of this approach is obvious. It is too costly to define syntax specific matching rules for all possible components and their combinations. It is also difficult to cope with the extension mechanisms such as a certificate and CRL extensions.



(a) DIT. (b) DIT of Attribute Extraction.

Figure 3. Example Directory Information Tree (DIT).

2.3.2 Attribute Extraction

To address these deficiencies, Klasen and Gietz [22] proposed an alternative solution, based on a practical workaround that PKI administrators have been using. A set of attributes are extracted from the certificate and stored as simple, searchable attribute together with the certificate in a newly created entry which is subordinate to the original one. For this purpose, they defined a set of 30 attributes [22] for the X.509 certificate. Matching is performed on the extracted attributes. The example DIT with extracted attributes is illustrated in Figure 3. DIT (a) in Figure 3 consists of person entries under the base `o=IBM,c=US` each of which contains a certificate attribute. After attributes are extracted, the person entry will have a new subordinate entry whose DN (Distinguished Name) becomes `x509Serial=12345,cn=John Doe,o=IBM,c=US`. The attribute extraction mechanism not only makes the end entity's view of a DIT different from the CA's who published the certificates but also doubles the number of entries at minimum.

With the attribute extraction mechanism, performing matching against components is identical to performing matching against attributes as depicted in Figure 2 (b). Although attribute extraction facilitates matching against components of a complex attribute, it can be considered as a suboptimal approach in the following respects. First, matching is performed on the extracted attributes, not on the certificate itself. Because the contents of the extracted attributes are mutable, there is non-zero chance of returning a wrong certificate to a client if the extracted attributes were maliciously forged. It is strongly recommended for the client to verify the returned certificate again to ensure strong security. In the server side, on the other hand, the server administrator must ensure the integrity of a certificate and the corresponding extracted attributes in order to minimize this security vulnerability. Second, when there are more than one certificates in a directory entry, one per key usage for example, it is not possible to pinpoint and return the certificate having the matching component (i.e. key usage for example again) since the searched-for attribute is different from the to-be-returned attribute. The matched value control [4] does not solve this problem, because an LDAP attribute is set of values, not sequence of values. Therefore, it is inevitable to transform the DIT structure in designing a certificate DIT to avoid the need for an additional searching step in the client [3]. Third, the attribute extraction does not facilitate matching against a composite assertion value as in X.500. It is not possible to support a flexible matching as in X.509 *certificateMatch* without making LDAP directory servers ASN.1 aware.

2.3.3 Certificate Parsing Server

An automatic attribute extraction mechanism was recently proposed. The Certificate Parsing Server (XPS) designed by the University of Salford [3] extends the OpenLDAP directory server in order to automatically extract and store the certificate components. Although it can significantly relieve the PKI administrator's burden, it does not improve the attribute extraction mechanism not to suffer from the three disadvantages of described above.

2.3.4 Component Matching

Component matching is recently published in RFC 3687 [18] in an effort to provide a complete solution to the LDAP - PKI interoperability problem. All attribute syntaxes of X.500 and LDAP are originally described by ASN.1 type specifications [15, 10]. However, LDAP uses LDAP specific encodings which does not generally preserves the structural information in the original ASN.1 type, instead of relying on an ASN.1 encodings. The component matching defines a generic way of enabling matching user selected components of an attribute value by introducing a new notion of component assertion, component filter, and matching rules for components. With component matching, it becomes possible to perform matching of an assertion value against a specific component of a composite attribute value. For example, infrastructure is provided to perform matching against an arbitrary component of an X.509 certificate, such as *serialNumber*, *issuer*, *subject*, and *keyUsage*. Technical details of the component matching will be explained in the following sections. Compared to the attribute extraction approach, component matching has the following advantages:

1. It does not extract and store certificate components separate from the certificates themselves. Therefore, it does not increase storage requirements and does not open a potential to the compromised integrity between a certificate and its extracted attributes.
2. Matching is performed not on the extracted attributes' contents but directly on the certificate's content. It can return only the matched certificate out of multiple certificates in a user's entry if it is used in conjunction with the matched values control [4].
3. It becomes convenient to provide a complex matching flexibly because matching between attribute and assertion values is performed at the ASN.1 layer.

3 Component Matching for PKI

3.1 Component Matching and Its Usage

The attribute syntaxes of X.500 are defined in ASN.1 types. The type is structurally constructed from basic types to composite types just like C struct definition. Every field of an ASN.1 type is a component. Based on ASN.1 types, component matching [18] defines how to refer to a component within an attribute value and how to match the referred component against an assertion value. Matching rules are defined for the ASN.1 basic and composite types. It also defines a new assertion and filter tailored for a component, or each field of the ASN.1 type. These definitions are based on ASN.1 so that they can be applied to any complex syntax, as long as it is specified in ASN.1.

The search filter for component matching is a matching rule assertion [10] whose matching rule is *componentFilterMatch* and whose

```
Certificate.toBeSigned :: = SEQUENCE {
    version          [0] EXPLICIT Version DEFAULT v1,
    serialNumber     CertificateSerialNumber,
    signature        AlgorithmIdentifier,
    issuer           Name,
    validity         Validity,
    subject          Name,
    subjectPublickKeyInfo subjectPublicKeyInfo,
    issuerUniqueID   [1] IMPLICIT UniqueIdentifier OPTIOMAL
    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL
    extensions       [3] EXPLICIT Extensions OPTIONAL
}
```

GSER encodings

```
{ version 2,
  serialNumber 12345 ,
  signature { algorithm 1.2.840.113549.1.14, parameters NULL},
  issuer {{type o, value IBM},{type c, value US}},
  validity {notBefore {2004 01 13 18 59}, notAfter {2005 01 13 18 59} },
  ...
}
```

Figure 4. Certificate ASN.1 Specification and GSER Encoding.

assertion value is a component filter. The search filter for component matching consists of three parts.

- *Component Reference*: specifies which component of the attribute value will be matched against the assertion value.
- *Matching Rule*: specifies which matching rule will be used to perform matching on the values.
- *Value*: An assertion value in GSER.

3.2 Certificate Access

Component matching, as introduced in Section 2.3.4, enables matching of an assertion value against a specific component of a certificate such as *serialNumber*, *issuer*, *subject*, and *keyUsage*. If a client receives a reference to a certificate consisting of the name of the issuing CA and its serial number, the client has to search for the certificate having matching *issuer* and *serialNumber* components in a certificate repository. Alternatively, a client may want to retrieve communicating party's certificates, not all of them, but only the ones for the non-repudiation purpose, by matching its distinguished name and key usage against the *subject* and *keyUsage* components of the certificate. For instance, the client can make an LDAP search request having the search filter illustrated in Figure 2 (c) to search for the certificates of `cn=John Doe,o=IBM,c=US` that are arranged to be used for non-repudiation. The example component filter of Figure 2 (c) contains two component assertions, one for *subject* and the other for *keyUsage*. The component references to these components begin with *toBeSigned* which is a sequence of certificate components to be digitally signed for immutability. *toBeSigned.serialNumber* refers to the *serialNumber* component of a certificate while *toBeSigned.extension.*.extnValue.(2.5.29.15)* refers to any extension of *keyUsage* type. In the latter example, *(2.5.29.15)* is the object identifier (OID) of the *keyUsage* extension. It is contained in *OCTET STRING* of the *extnValue* of any components of *extensions* certificate component. In other words, the reference means identifying all *keyUsage* extension components.

The component matching rule specifies which matching rules will

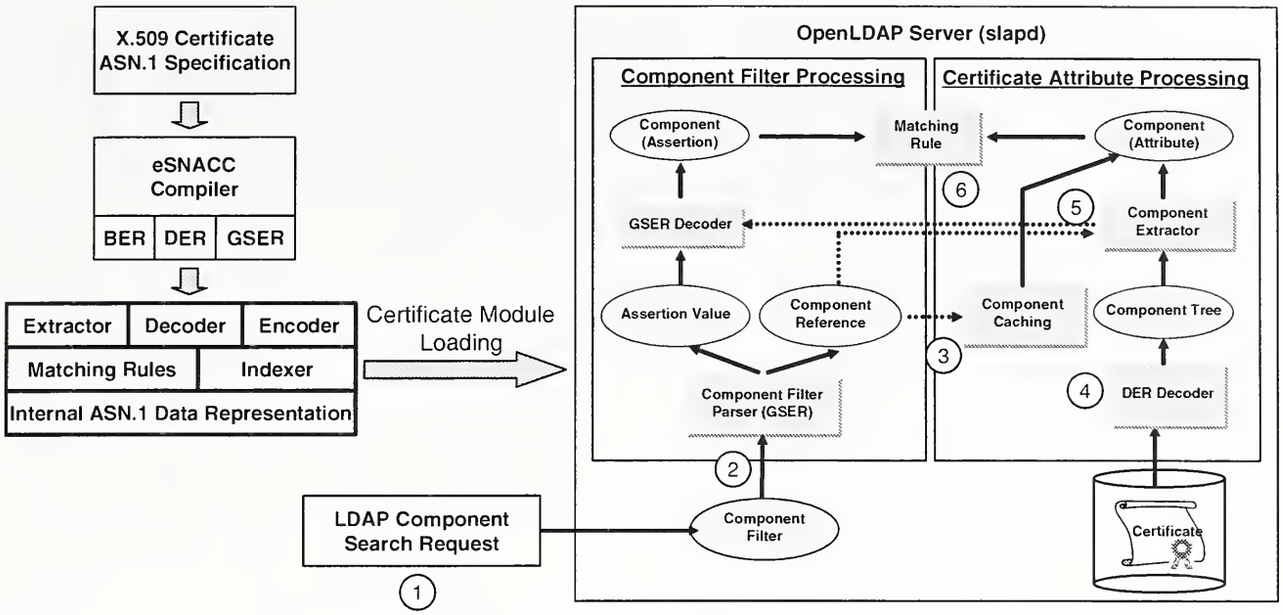


Figure 5. Architecture of Component Matching in OpenLDAP.

be used to perform matching a certificate's component against an assertion value. Either existing matching rules or newly defined matching rules can be used as the component matching rules. Matching rules for composite types can be provided by combining those of their subordinate types. *allComponentsMatch* implements matching at the ASN.1 layer whereas derived matching rules can be defined to override it with specific syntaxes.

RFC 3687 [18] defines which matching rules can be applied to each of the ASN.1 types. In the example component filter in Figure 2 (c), *distinguishedNameMatch* is used for *subject* and *bitStringMatch* is used for *keyUsage*. The component assertion value is a GSER-encoded value asserted against the component selected by the component reference. In Figure 2 (c), the value `cn=John Doe,o=IBM,c=US` is the GSER encoded ASN.1 *UTF8 STRING* and the value '010000000'B is the GSER encoded ASN.1 *BIT STRING* value.

The client sends a search request containing the component filter to a component matching enabled LDAP directory server. In response, the client will be returned with those entries having the matching certificate if there is any. After checking the authenticity and integrity of the returned certificate, the client can extract the public key out of the certificate for further use.

3.3 Certificate Revocation List (CRL) Access

Although certificates were valid at the time when they were issued, CA must revoke certificates occasionally because the key pair for a certificate can be compromised or the binding between an identity and a certificate become invalid. As a result, a certificate should be validated when it is used. Otherwise, the client might incur not only incomplete, but also insecure transactions. The CRL mechanism [11] provides a means of performing validation of certificates against periodically published list of revoked certificates, or Certificate Revocation List (CRL). A CRL is periodically generated by the CA and is made available through certificate repositories such as LDAP directories. Although the CRL mechanism has been carefully revised to reduce the CRL download traffic which can degrade the scalability of PKI significantly, it still requires the end entities to store CRLs in its local storage in order to facilitate efficient and off-line operation. On the other hand, on-line certificate validation protocols are also proposed in order to cope with the on-line end entities which need more fresh information on certificate validity. Because the on-line end entities need not store the certificate status information in its storage, the on-line protocols also eliminate the requirement for the hefty local certificate storage. Online Certificate Status Protocol (OCSP) [21] and Simple Certificate Validation Protocol (SCVP) [9] are two examples of the on-line certificate validation protocols.

We conceive that component matching enabled LDAP can also be used as an on-line certificate validation protocol. CRL is a sequence of pairs of a revoked certificate's serial number and revoked time [11]. In order to check status of the certificate, the client needs to make a component assertion against the serial number of the certificate under scrutiny. Then, the LDAP server will perform component matching on the CRL against the assertion to find the asserted serial number in the CRL. This is possible with component matching, since the LDAP server understands the structure of the CRL and is able to compare specific components of the CRL against the component assertion. In the attribute extraction approach, however, the serial numbers of all the elements of the revoked certificate list must be extracted as separate attributes which need to be stored in the individual subordinate entries. This not only increases the amount of storage and increases the complexity of managing directory significantly, but also makes the server vulnerable to malicious attacks as explained in Section 2.3.4.

With component matching, the whole CRL does not necessarily have to be downloaded to the client and scanned by the client so as to save the network bandwidth and the client's computing power significantly. Especially for the clients which have limited computing power and low bandwidth such as mobile devices, component matching will be very efficient solution for the client to access PKI. Furthermore, an LDAP server already has been widely used

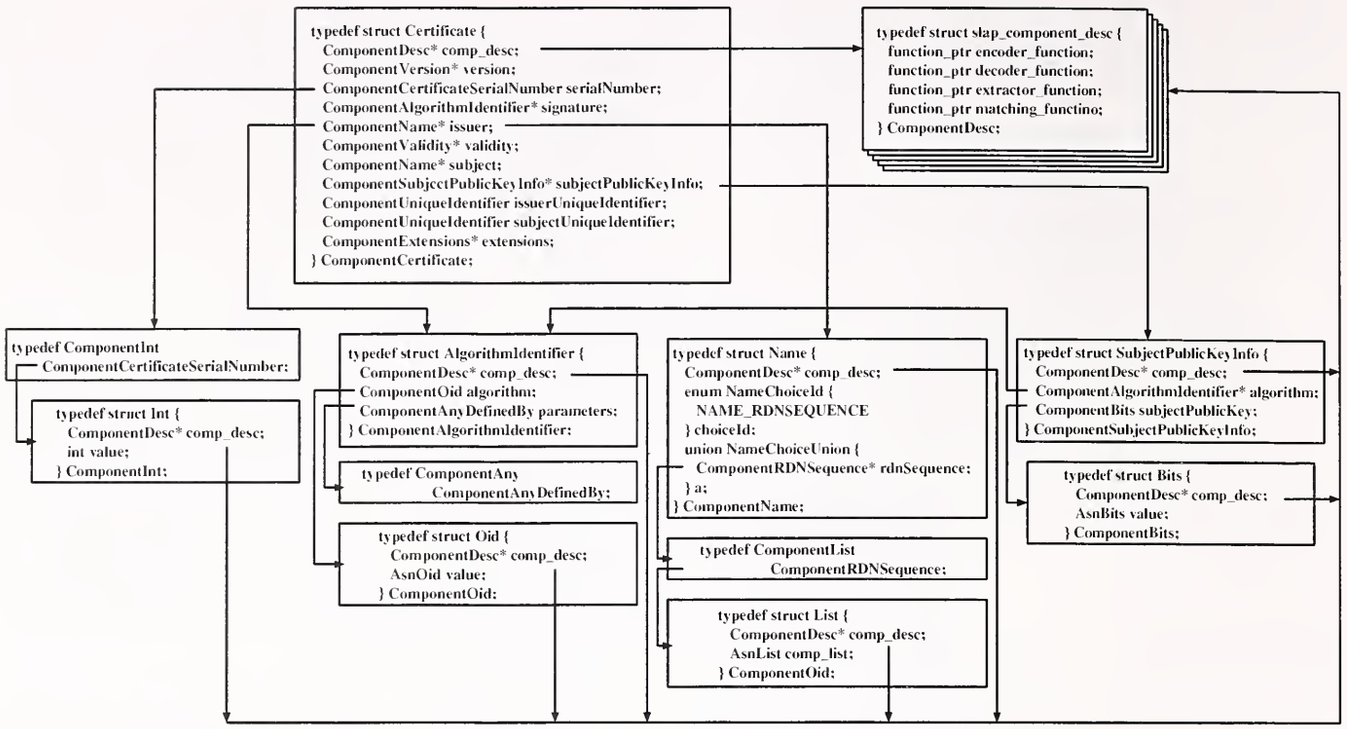


Figure 6. Certificate Component Tree.

for distributing CRLs and certificates. Hence, if the server can perform on-line validity checking over the CRL as well, it will be very practical and efficient alternative to OCSP which needs additional software, or an OCSP responder.

In [6], we also propose to structure the internal representation of CRL as an authenticated data structure such as the Certificate Revocation Tree (CRT) [16] and the authenticated 2-3 tree [23]. Together with the component matching, it makes certificate validation result from an LDAP server unforgeable while not requiring to have the LDAP server as a trusted entity nor to sign every LDAP response on the fly as in OCSP.

4 GSER (Generic String Encoding Rules)

A native LDAP encoding does not represent structure of an ASN.1 type. Instead, it is either in octet string or in binary. With the LDAP encoding, as a result, it is difficult to contain the structural information of ASN.1 type in its representation. In order to solve this problem, S. Legg [17] recently proposed GSER (Generic String Encoding Rules). Component matching uses GSER as its basic encoding for the component assertion value. GSER generates a human readable UTF-8 character string encoding of a given ASN.1 specification with predetermined set of characters to keep the structure such as '{', '}', and ','. It defines UTF8 string encodings at the lowest level of the ASN.1 built-in types such as INTEGER, BOOLEAN, and STRING types and then it builds up more complex ASN.1 types such as SEQUENCE and SET from the lowest level by using the characters. Thus, the structural information of an ASN.1 specification is maintained in encodings so that it can be recovered in the decoding process easily. By using GSER to store attribute values instead of the native LDAP encoding, an LDAP server is capable of identifying the structure of ASN.1 specification of the attribute. Furthermore, the component filter itself is also encoded in GSER. Hence, GSER is an essential mechanism to ASN.1 awareness and component matching.

Figure 4 shows the ASN.1 type specification of a *toBeSigned* and its GSER encodings. The certificate is SEQUENCE so that there are curly braces at the beginning and at the end of its GSER encodings. It has *version*, *serialNumber*, etc. as its components inside of SEQUENCE. Within the braces, there is *version* and *2*, or its value, followed by comma which separates the subsequent field encoding. GSER defines each basic type's encoding and then combines them structurally to a more complex one by using "{", "," and "}". On the other hand, a native LDAP encoding does not have any systematic rule to construct the structure information of attribute value in it.

5 Component Matching Implementation in OpenLDAP

The overall conceptual architecture of the component matching in the OpenLDAP *slapd* directory server is illustrated in Figure 5. Given the ASN.1 specification of the X.509 certificate as an input, the extended *eSNACC* ASN.1 compiler generates the *slapd* internal data representation of the X.509 certificate and their encoding / decoding routines. We extended the *eSNACC* ASN.1 compiler [7] to support GSER in addition to the originally supported BER and DER [7]. It also generates component equality matching rules, component extract functions, and component indexer functions which will be discussed later in this section in detail. In order to facilitate the integration of the newly defined syntaxes without the need of rebuilding the *slapd* executable, the generated data structures and routines are built into a module which can be dynamically loaded into *slapd*. The overall flows of LDAP component search is explained as follows;

1. On the client side, a search for components of X.509 certificate is initiated by the inclusion of the *ComponentFilter* in the filter of the search request. A *ComponentFilter* consists of *ComponentAssertion*s each of which is in turn comprised of *component*, *rule*, and *value*.
2. On the server side, whenever *slapd* detects that the search request contains *ComponentFilter*, it parses the incoming component filter to obtain assertion values and component references. The assertion values are also converted to the ASN.1 internal representation by the GSER decoder.
3. Retrieve the entry cache to see if the target certificate's decoded component tree is cached. If so, skip the following steps upto the step 6.
4. If it is not cached, by using an appropriate ASN.1 decoder, *slapd* decodes the *certificate* attribute into the component tree, the ASN.1 internal representation, when loading the candidate entries containing certificate for matching. Because a certificate is DER encoded, DER decoder is used to construct a certificate's component tree.
5. The component reference is fed into the component extractor to obtain the component subtree which is referenced by *component reference* out of the attribute component tree.
6. The assertion component and the extracted attribute component are then matched together by the matching rule corresponding to the component which is generated also by the extended eSNACC compiler. Matching is performed at the abstract level using the internal ASN.1 data representation.

The rest of the section provide detailed description of the component matching in two steps. After first describing how to make the OpenLDAP directory server ASN.1 aware in detail, component filter processing, aliasing, component indexing, and component caching will be described.

5.1 ASN.1 Awareness

5.1.1 eSNACC Compiler

Figure 6 shows the internal data representation of the *toBeSigned* ASN.1 type along with the representations of some of its key components. The data structures for this ASN.1 data representation are automatically generated by the eSNACC compiler from the given ASN.1 specification of *toBeSigned*. The generated data structure for the *toBeSigned* has data fields corresponding to components of the *toBeSigned* ASN.1 type. Once the internal data structure for *toBeSigned* is instantiated, it can be converted to DER by `DEnctoBeSigned()` and back to the internal representation by `DDectoBeSigned()`.

Component matching can be performed for any composite attributes which are encoded as one of the ASN.1 encoding rules. In addition to the DER used for a certificate, we have implemented a GSER backend in the extended eSNACC compiler. GSER can be used as an LDAP-specific encodings for newly defined attribute types. With GSER, string-based LDAP-specific encodings can maintain the structure of their corresponding ASN.1 types. The assertion values in the component filter are also represented in GSER and the extended eSNACC compiler is used to decode them into their internal representations.

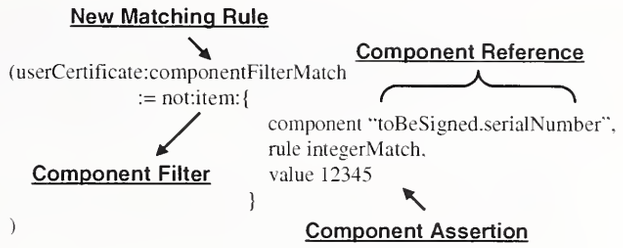


Figure 7. Example Component Filter.

5.1.2 Internal Representation of ASN.1 Type

A new data structure of *slapd* is needed to represent an attribute value as its components because the original data structure for attribute types does not contain the structural information of an ASN.1 type in its representation. Every field of an ASN.1 type is a component which is addressable by a component reference. In our implementation, the component data structure consists of two parts: one to store the value of the component; the other to store a component descriptor which contains information on how to encode, decode, and match the values.

The data structure of a component appears as a tree which keeps the structural information of the original ASN.1 specification using nodes and arcs. Each component node of the tree not only has data values but also represents the structural information of the given ASN.1 specification by having links to subordinate nodes. In the tree, any node can be referenced by a component reference in order to perform matching on the corresponding component. Hence, we need a function to traverse the tree and locate the referenced node. The ASN.1 compiler also generates component extractor routines for this purpose.

Figure 6 illustrates the component data structure for *Certificate.toBeSigned*. For the convenience of illustration, only *serialNumber*, *signature*, *issuer*, and *subjectPublicKeyInfo* are shown with their component subtrees among ten components of *Certificate.toBeSigned*. Let's look at *subjectPublicKeyInfo* in more detail. Its component data structure, *ComponentSubjectPublicKeyInfo*, contains a pointer to its component descriptor and its own subordinate components, *algorithm* and *subjectPublicKey*. *Algorithm* is represented by *ComponentAlgorithmIdentifier* and *subjectPublicKey* is of the ASN.1 *BIT STRING* type which is represented by *ComponentBits*. Leaf nodes of the component tree, such as *ComponentBits* and *ComponentInt*, contain the values of the ASN.1 basic types.

5.1.3 Syntax and Matching Rules

An attribute is described by an attribute type in LDAP. An attribute type contains two key fields which help to define the attribute as well as the rules that attribute must follow. The first field is syntax which defines the data format used by the attribute type. The second field is matching rule which is used by an LDAP server to compare an attribute value with an assertion value supplied by LDAP client search or compare operations. Attributes must include the matching rules in their definition. At least, equality matching rule should be supported for each attribute type. From the viewpoint of an LDAP server, an ASN.1 specification defining a new attribute type requires a new syntax and its matching rule to be defined. To fully automate the component matching in which the

Table 1. Attribute Aliasing Table.

Alias Attribute	Aliased Attribute	Component Reference	Matching Rule
x509certificateSerialNumber	*userCertificate*	*toBeSigned.serialNumber*	*integerMatch*
x509certificateIssuer	*userCertificate*	*toBeSigned.issuer*	*distinguishedNameMatch*

composite attribute types are defined in ASN.1, we extended the *eSNACC* compiler to generate the basic equality matching rule of a given ASN.1 type, or *allComponentMatch* matching rule specified in RFC 3687 [18]. *allComponentMatch* matching rule evaluates to true only when the corresponding components of the assertion and the attribute values are the same. It can be implemented by performing matching from the topmost component which is identified by the component reference recursively down to the subordinate components. The generated matching function of each component can be overridden by other matching functions through a matching rule refinement table. Therefore, it is possible that a syntax developer can replace the compiler-generated matching functions with the existing matching functions of *slapd* which might be more desirable. In order to support this refining mechanism, *slapd* checks the refinement table whether it is overridden by looking up the table, whenever a matching functions are executed.

5.2 Component Matching

5.2.1 Component Assertion and Filter

RFC 3687 [17] defines a new component filter as the means of referencing a component of a composite attribute and as the means of representing an assertion value for a composite attribute types. Component assertion is an assertion about presence or values of components within an ASN.1 value. It has a component reference to identify one component within an attribute value. Component filter is an expression of component assertion, which evaluates to either *TRUE*, *FALSE*, or *Undefined* while performing matching. Figure 7 illustrate the example component filter. The component reference or *toBeSigned.serialNumber* identifies one component in the certificate attribute value. In the component reference, "." means identifying one of components subordinate to the preceding component. In the component assertion, *rule* is followed by an *integerMatch* matching rule [15] which will be used to compare the following assertion value with the referenced component of the attribute value. The routines required to support the component filter and the component assertion were hand-coded while the routines for the component assertion values are automatically generated from a given ASN.1 type.

5.2.2 Attribute / Matching Rule Aliasing

To enable component matching, clients as well as servers need to support GSER and new component matching rules. However, the client side changes will be minimal if at all, because the component filter can be specified by using the existing extensible matching rule mechanism of LDAPv3 and the component assertion value is represented as the text centric GSER encoding rules. Especially, the clients that accept search filters as strings require no changes to utilize component matching other than filling in the necessary component filter as the search filter. However, for those clients who have search filters hard coded in them, we propose an attribute aliasing mechanism which maps a virtual attribute type to an attribute component and a component matching rule and a matching rule aliasing mechanism which maps a virtual matching rule to a component assertion.

Table 2. X509 Certificate Decoding Time.

	d2i_X509() OpenSSL	ASN.1 Decoder
Time (usec)	32.74	40.20

Attribute alias registers a set of virtual attributes to an LDAP server. The virtual attributes themselves find corresponding matching rules and component references by looking up an attribute alias table. The example attribute alias table is shown in Table 1. *X509certificateSerialNumber* attribute is aliased to "*userCertificate.toBeSigned.serialNumber*" with the *integerMatch* matching rule. Hence, the filter "(*x509certificateSerialNumber=12345*)" is considered equivalent to "(*userCertificate:ComponentFilter:=item:component userCertificate.toBeSigned.serialNumber, rule caseExactMatch, value 12345*)". With the attribute aliasing, clients only have to form simple assertions to utilize component matching. Matching rule alias works in a similar way. An alias matching rule is mapped into the corresponding component reference and matching rule.

5.2.3 Component Indexing

The maintenance of proper indices is critical to the search performance in the Component Matching as much as in the conventional attribute matching. In *slapd*, the attribute indexing is performed by generating a hash key value of the attribute syntax, matching rule, and the attribute value and maintain the list of IDs of those entries having the matching value in the set of attribute values of the indexed attribute.

The component indexing can be specified in the same way as the attribute indexing, except that the component reference is used to specify which component of a composite attribute to be indexed. If the referenced component is a basic ASN.1 type, the indexing procedure will be the same as the attribute indexing. The indices for the referenced component are accessed through a hashed value of the corresponding syntax, matching rule, and value in the index file for the referenced component of the composite attribute. In OpenLDAP, the indexing of the composite component is centered on the GSER encoding of the component value. The hash key of a component value is generated from its GSER encodings together with its syntax and matching rule. For the SET and SET OF constructed types, it is required to canonicalize the order of the elements in the GSER encodings before generating the hashed key value. For <all> component reference of SET OF and SEQUENCE OF constructed types, its is needed to union the indices for each value element of SET OF and SEQUENCE OF.

5.2.4 Component Caching

Whenever a certificate is matched against an incoming component filter, it is repeatedly decoded into the internal representation from DER. This requires non-negligible CPU cycles as presented in Table 2.

In order to eliminate the repeated decoding overhead, we decided to cache certificates in their decoded form, i.e. in the component

```
<ds:KeyInfo>
  <wsse:SecurityTokenReference>
    <ds:X509Data>
      <dsext:GenericCertificateReference xmlns:dsext="..." EncodingType="...#XER">
        <dsext:CertificateAssertion>
          <dsext:serialNumber>8fb2adb53a9056a511d356947cedeec0</dsext:serialNumber>
          <dsext:issuer>o=IBM,c=US</dsext:issuer>
          <dsext:keyUsage>0</dsext:keyUsage>
        </dsext:CertificateAssertion>
      </dsext:GenericCertificateReference>
    </ds:X509Data>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

(a) XER.

```
<ds:KeyInfo>
  <wsse:SecurityTokenReference>
    <ds:X509Data>
      <dsext:GenericCertificateReference xmlns:dsext="..." EncodingType="...#GSER">
        { serialNumber "8fb2adb53a9056a511d356947cedeec0", issuer "o=IBM,c=US" , keyUsage '010000000'B }
      </dsext:GenericCertificateReference>
    </ds:X509Data>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

(b) GSER.

Figure 8. Example GenericCertificateReference.

tree structure explained in Section 5.1.2. In the OpenLDAP directory server, the entry cache is provided to store frequently requested entries to enable very low latency access [5]. We extended the current entry cache to store decoded certificates along with other attributes of the entry. We devised various caching policies for the entry cache. In early implementations, we decided to cache a decoded certificate as a whole, along with its entry in the entry cache. The size of a certificate is 899Bytes and that of the corresponding component tree is 3KBytes. Caching all the decoded component tree consumes more than three times as much memory compared to the base entry caching. To reduce the memory requirements, we devised an indexing based caching policy. Since it is a common practice to index those attributes that are likely to be asserted, caching only those indexed components is a very practical solution to reduce the memory requirement. In our experiment, the serial number component was cached which takes only 148Bytes of memory.

6 Component Matching in WS-Security

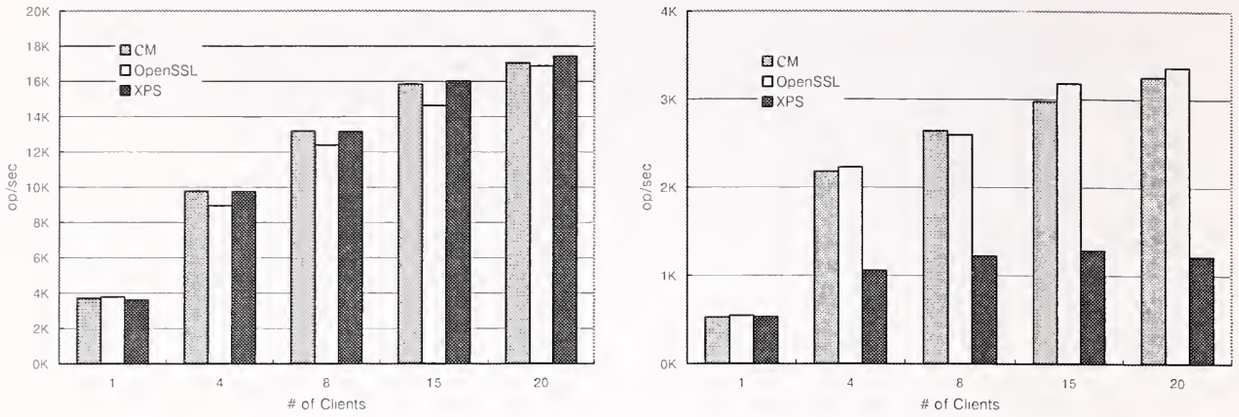
SOAP (Simple Object Access Protocol) is a protocol for invoking methods on servers, services, components, and objects [1]. It is a way to create widely distributed, complex computing environments that run over the Internet using existing Internet infrastructure, enabling Web service developers to build Web services by linking heterogeneous components over the Internet. For interpretability over heterogeneous platforms, it is built on top of XML and HTTP which are universally supported in most services. WS-Security is recently published as the standard for secure Web Services [24]. It provides a set of mechanisms to help Web Services exchange secure SOAP message. WS-Security provides a general purpose mechanism for signing and encrypting parts of a SOAP messages for authenticity and confidentiality. It also provides a mechanism to associate security tokens with the SOAP messages to be secured. The security token can be cryptographically endorsed by a security authority. It can be either embedded in the SOAP message or acquired externally. There are two types of PKI clients in WS-Security: one directly accesses PKI; the other indirectly accesses it by using service proxies such as XML Key Management System (XKMS) [30] which provides clients with a simple-to-use interface to a PKI so as to hide the complexities of the underlying infrastructure.

In the X.509 token profile of WS-Security [25], it is defined that the following three types of token references can be used:

1. Reference to a Subject Key Identifier: value of certificate's *X.509SubjectKeyIdentifier*.
2. Reference to a Security Token: either an internal or an external URI reference.
3. Reference to an Issuer and Serial Number: the certificate issuer and serial number.

Because it is defined as extensible, any security token can also be used based on schemas. It is shown in Figure 8 that the *<ds:X509Data>* element of *<ds:KeyInfo>* is used as the security token. *<ds:X509Data>* defined in [31] contains various references such as *X509IssuerSerial*, *X509SubjectName*, *X509SKI*, and so on. With the ASN.1 awareness and the component matching support in the OpenLDAP directory server, these references can be used without the need of implementing syntax specific matching rules for various types of references. It is also possible in *<ds:X509Data>* to use elements from external namespace for further flexibility.

Figure 8 shows one such example. Here, *GenericCertificateReference* element from *dsext* namespace is used to provide a generic reference mechanism which implements *CertificateMatch* in the X.509 recommendation [14]. The reference consists of a sequence of certificate attributes, *serialNumber*, *issuer*, *subjectKeyIdentifier*, *authorityKeyIdentifier*, *certificateValid*, *privateKeyValid*, *subjectPublicKeyAlgID*, *keyUsage*, *subjectAltName*, *policy*, *pathToName* each of which is defined optional. By using the example reference, it would be possible to perform security key reference in a very flexible way. It would be possible to search for a certificate having



(a) 100K Entries (No Memory Pressure). (b) 500K Entries (High Memory Pressure).

Figure 9. The Performance of Three Approaches.

a *subjectAltName* with a specific *keyUsage*. Figure 8(a) shows that the reference is encoded in XML while Figure 8 (b) shows that the reference is encoded in GSER.

With the component matching enabled LDAP server, the GSER encoded reference value can be used as an LDAP assertion value in a component filter. With the ASN.1 awareness support, the LDAP server is now capable of understanding the structure of the *CertificateAssertion* type when configured with its ASN.1 definition. Because encoders / decoders for various encoding rules (GSER, DER, XER ...) are automatically generated and integrated into the LDAP server, it is possible to use ASN.1 values encoded in those encoding rules as an assertion value in an LDAP search operation.

With the ASN.1 aware and component matching enabled LDAP server, flexible reference formats for X.509 certificates can now be defined in ASN.1 to configure the LDAP server to understand the reference. The required matching rules, encoders, and decoders for the reference type will be automatically generated and integrated to the LDAP server. This increased flexibility will foster the flexible use of security token references in the LDAP server by making it easy to create and update references.

7 Experimental Results

We used MindCraft's DirectoryMark [20] tools to generate the directory entries and client scripts containing a list of LDAP operations. The client scripts were run on an 8-way IBM xSeries 445 server with Intel Xeon 2.8GHz processors and the directory server was run on an IBM xSeries 445 server with 4 Intel Xeon 2.8Ghz processors and with 12GB of main memory running SUSE SLES9 (Linux kernel version 2.6.5). We used the transactional backend (back-bdb) of OpenLDAP version 2.2.22 together with Berkeley DB 4.3 and OpenSSL 0.9.7 for the evaluation. Two different size DITs with 100K and 500K entries were used for evaluation. Our intension of using two different size DITs was to observe the throughput of *slapd* with and without memory pressure. With 100k entries, all the entries was able to be cached into the DB cache. On the other hand, with 500k entries, we observed that the server experienced a number of memory swapping and disk I/O due to memory shortage. The directory was indexed for *cn*, *sn*, *email* of *inetOrgPerson* and for *serialNumber* and *issuer* of *userCertificate* (or the corresponding extracted attributes in the case of attribute extraction mechanism).

In the experiment, OpenLDAP stand-alone directory server, *slapd*, was used as an LDAP certificate repository testbed for all three methods. *Slapd* as of OpenLDAP version 2.2.22 supports both the component matching and the certificate specific matching. The attribute extraction mechanism was tested by using the XPS patch to OpenLDAP which was contributed to the OpenLDAP project by University of Salford. XPS was used to automatically generate the DIT for the attribute extraction. The same version of *slapd* was tested for all three mechanisms for the LDAP certificate repository.

Fgure 9 (a) shows the throughput of three approaches, varying the number of clients. With 100k entries, the peak throughput of component matching and attribute extraction mechanisms are almost the same. The certificate-syntax specific matching (*OpenSSL* decoder) exhibits slightly lower performance than the other two methods. We attribute the reason of lower throughput to longer code path of *slapd* such as normalization and sanity checks of assertion values when it uses the OpenSSL library. In order to observe the behavior of the three methods in the presence of memory pressure, we increased the number of entries to 500K and the database cache size is reduced from 1GB to 200MB. With this configuration, only small portion of the entries can be cached and hence the system suffers from frequent memory swapping. Figure 9 (b) shows that the throughput of all three methods are degraded significantly compared to Figure 9 (a). The peak throughput of component matching is 3250 ops/sec, significantly degraded from 17,057 ops/sec with no memory constraint. The attribute extraction mechanism is hit by even further performance degradation than the other two mechanisms. This is because the number of entries becomes doubled by extracting attributes and by having them as separate entries subordinate to the original ones. This results confirms that the component matching is a superior approach to the attribute extraction with respect to performance as well as to security and manageability.

8 Conclusion

Although it is a general consensus in the PKI standardization working group that the component matching is a complete solution to the LDAP - PKI interoperability problem, it was under debate that its adoption in LDAP servers might be slow and an alternative solution needed be pursued in the interim. In this paper, we have presented the design and implementation of the component matching in OpenLDAP *slapd*. Our work provided a strong evidence that the

component matching can be implemented in pure LDAP-based directory servers without exploding complexity and degrading performance. Our work also proposed a number of enhancements to the component matching technology to improve performance and interoperability with legacy clients. In this paper, we also proposed the use of the component matching enabled LDAP as a secure on-line certificate validation protocol. We further demonstrated the usefulness of the component matching in WS-Security as a key application. As PKIs are being adopted in larger scale and in more critical deployments, it becomes more important to provide as complete a solution as possible, especially when it comes to security. The component matching technology enables more secure and flexible implementation of LDAP certificate repositories for PKI without compromising performance.

9 Availability

The component matching software is included in OpenLDAP release as a module and can be downloaded at `http://www.openldap.org/software/download/`. The eSNACC ASN.1 compiler can be obtained from DigitalNet at `http://digitalnet.com/knowledge/download.htm`.

10 References

[1] D. Box and D. Ehne. Simple object access protocol (SOAP). W3C Note, May 2000.

[2] D. W. Chadwick. Deficiencies in LDAP when used to support PKI. *Comm. of the ACM*, 46(3), March 2003.

[3] D. W. Chadwick, E. Ball, and M. V. Sahalayev. Modifying LDAP to support X.509-based PKIs. In *17th Annual IFIP WG 11.3 Working Conference on Database and Applications Security*, August 2003.

[4] D. W. Chadwick and S. Mullan. Returning matched values with LDAPv3. RFC 3876, September 2004.

[5] J. H. Choi, H. Franke, and K. D. Zeilenga. Performance of the OpenLDAP directory server with multiple caching. In *Proceedings of International Symposium on Performance Evaluation of Computers and Telecommunication Systems*, July 2003.

[6] J. H. Choi, S. S. Lim, and K. D. Zeilenga. On-line certificate revocation via LDAP component matching. To be presented in DIMACS Workshop on Security of Web Services and E-Commerce, May 2005.

[7] DigitalNet. Enhanced SNACC ASN.1 software. `http://www.digitalnet.com/knowledge/snacc_home.htm`.

[8] eB2Bcom. `http://www.e2b2com.com`.

[9] T. Freeman, R. Housley, A. Malpani, D. Cooper, and T. Polk. Simple certificate validation protocol (SCVP). `<draft-ietf-pkix-scvp-18.txt>`, Feburuary 2005.

[10] J. Hodges, R. Morgan, and M. Wahl. Lightweight directory access protocol (v3): Technical specification. RFC 3377, September 2002.

[11] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 public key infrastructure certificate and CRL profile. RFC 2459, January 1999.

[12] ITU-T Rec. X.690, ASN.1 encoding rules: Specification of basic encoding rules (BER), canonical encoding rules (CER), and distinguished encoding rules (DER), 1994.

[13] ITU-T Rec. X.680, Abstract syntax notation one (ASN.1): Specification of basic notation, December 1997.

[14] ITU-T Rec. X.509, The directory: Public-key and attribute certificate frameworks, March 2000.

[15] ITU-T Rec. X.500, The directory: Overview of concepts, models and service, February 2001.

[16] P. C. Kocher. On certificate revocation and validation. In *Proc. of the 2nd Int'l Conference on Financial Cryptography (Lecture Notes in Computer Science, Vol. 1465)*, pages 172–177, 1998.

[17] S. Legg. Generic string encoding rules. RFC 3641, October 2003.

[18] S. Legg. X.500 and LDAP component matching rules. RFC 3687, February 2004.

[19] S. S. Lim, J. H. Choi, and K. D. Zeilenga. Secure and flexible certificate access in WS-Security through LDAP component matching. In *ACM Workshop on Secure Web Services held in conjunction with the 11th ACM Conference on Computer and Communications Security*, October 2004.

[20] Mindcraft. DirectoryMark. `http://www.mindcraft.com/directorymark/`.

[21] M. Myers, R. Ankney, A. Malpani, and C. Adams. Internet X.509 public key infrastructure online certificate status protocol - OCSP. RFC 2560, June 1999.

[22] N. Klasen and P. Gietz. Internet X.509 public key infrastructure lightweight directory access protocol schema for X.509 certificates, `<draft-ietf-pkix-ldap-pkc-schema-01.txt>`, October 2004.

[23] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proc. of the 7th USENIX Security Symposium*, pages 217–228, January 1998.

[24] OASIS. Web services security: SOAP message security 1.0 (WS-Security 2004). OASIS Standard 200401, March 2004.

[25] OASIS. Web services security: X.509 certificate token profile. OASIS Standard 200401, January 2004.

[26] OpenLDAP. `http://www.openldap.org`.

[27] OpenSSL. `http://www.openssl.org`.

[28] The Unicode Consortium. *The Unicode Standard, Version 4.0*. Addison-Wesley, Boston, 2003.

[29] View500. `http://www.view500.com`.

[30] W3C. XML key management specification (XKMS). W3C Standard, March 2001.

[31] W3C. XML - signature syntax and processing. W3C Standard, February 2002.

[32] F. Yergeau. UTF-8, a transformation format of ISO 10646. RFC 3629, November 2003.

PKI without Revocation Checking

Karl Scheibelhofer
Institute for Applied Information Processing and Communications
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
Email: karl.scheibelhofer@iaik.tugraz.at

Abstract

Current X.509-based PKIs are typically complex. One of the most critical parts is revocation checking. Providing revocation information is a big effort for CAs, and for clients it is even more costly to get all required revocation data. This work provides a performance calculation of two CA setups with CRLs, delta CRLs and OCSP as revocation checking mechanism. The enhancement of this work is the proposal to avoid revocation checking by issuing certificates on-line and only retroactively. An analysis shows that this approach performs at least as good as OCSP and much better than CRLs. In addition, this solution reduces the complexity of PKI significantly.

Introduction

Today, most computers are connected to a network. Usually, this is a corporate network, a home network or a campus network, and this network is commonly connected to the Internet through firewalls and routers. Pure off-line systems are the exception rather than the rule nowadays. Even if many computers are only on-line from time to time, they attach to a network on a regular basis. This holds for notebooks, PDAs and smart phones. However, with most of these devices, it is easy to go on-line. The Internet becomes ubiquitous: Ethernet at the office, cable modem or DSL at home, WLAN at campus, airports and rail stations, 3G in cities and more to come. Network connectivity is no scarce resource any longer, as well as performance. Even smart phones can create and verify digital signatures in much less than a second. Certificate based PKIs have been designed based on the assumption that network connectivity is a limited and expensive resource (see III in [13]).

This paper considers current certificate and revocation checking practice in the light of ubiquitous access to the Internet. The focus is on X.509 certificates [1] and current revocation checking techniques like CRLs and OCSP [2]. Based on this analysis, we propose a simple way to make certificate handling and validation easier in a networked environment. The proposed approach is an on-line certification service. The CA issues a new certificate each time the key owner uses the key. We will show that this requires not more effort than OCSP, neither on the CA side nor on the client side. Moreover, it reduces the overall complexity of the system and provides up-to-date status information about the key. This can be seen as a special version of short-lived certificates, which have been proposed several times (e.g. [9]). Unlike short-lived certificates, here, the validity time degrades to an instant in time which lies in the past. Thus, the statement made by the CA through a certificate can be definitive, which makes revocation a less critical issue in many cases. There is no need for revocation checking of such certificates; the certificate already contains the status information.

Certificate Validity

Each X.509 certificate contains a validity period, which is a time interval determined by a start date and an end date. The start date is normally the time at which the CA issued

the certificate. The end time is often calculated as a fixed offset from the start time, e.g. start time plus one year. One year is a typical validity frame for a public-key certificate. The certificate validity is often interpreted as "*this certificate is valid during this time interval unless it is revoked*". However, PKIX [1] defines it differently: *The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate.* On the first look, the two interpretations seem to be similar, but there are subtle differences. The PKIX definition does not imply any validity information. A certificate can still be valid even beyond its validity period. The CA simply does not guarantee to provide information about the validity of this certificate outside this time frame. The first interpretation does not accept a certificate which has been used outside its validity period, no matter, if other status information is available. In practice, the application of both interpretations ends in the same results. The reason is that most CAs do not provide positive status information for certificates. This means, the CA tells clients what certificates are definitely revoked, but only for certificates which are within their validity period. For other certificates, the clients do not get any revocation information. If a certificate runs out of its validity period, we call it expired. In case a considered certificate expired, clients have no means to find out if the CA no longer provides revocation information or if the certificate has not been revoked. In other words, a client may or may not find out if an expired certificate has been revoked before it expired. The client has no guarantee that such certificates are listed on the CRL.

The situation is similar for OCSP. Since OCSP responders may cache old CRLs, they may provide certificate revocation status beyond certificate expiration. A responder can indicate such support by adding a special extension to its response messages, the archive cutoff extension. Thus, OCSP responders may extend the period through which status information about certificates is available. However, it does not really extend the validity period in the sense defined by PKIX because in practice, a certificate owner cannot revoke a certificate which has already expired. In consequence, relying parties can get information about revocation issues which happened within the certificate's validity period, but they cannot get revocation issues beyond this period. The CA does not offer such service for expired certificates.

Revocation Checking

In general, a certificate which is within its validity period can be valid or revoked. A client cannot find out the status of the certificate off-line. Finding out if a certificate is valid requires access to some network resource. This can be the current CRL on a web server or an OCSP responder.

CRLs

CRLs are a problematic mechanism for revocation checking. For the CAs they provide a simple means with low requirements concerning signature creation and security. For the relying parties who need to verify certificates, CRLs are a pain. They are often quite large, they often contain close to 100% irrelevant information and they usually provide no current revocation information. Consider SSL [4] server certificates from Verisign, Inc.. They contain a CRL distribution point referring to the location of the current CRL. However, this CRL is about 750 kByte. In consequence, most browsers work without revocation checking for server certificates. Downloading the CRL would take considerable time and would delay the HTTP transfer unacceptably long.

CRLs have a validity period similar to certificates. It is determined by the time at which the CRL has been issued, called *thisUpdate*, and the time at which the next CRL will be issued latest, called *nextUpdate*. According to PKIX, a client can cache a CRL until *nextUpdate* and rely on this information. However, it is apparent that a CRL cannot provide status information newer than indicated in its *thisUpdate* field. Caching the CRL is nice and provides a benefit in some special applications, but for many applications, it does not make much sense. System designers using cached CRLs should be aware of the fact that relying on cached CRLs may produce non-deterministic behavior for revocation checking. Consider a web server certificate. The mentioned CRL has a validity period of two weeks. Thus, in case of certificate revocation, clients will notice this revocation up to two weeks later and one week later on average if they cache the CRL as long as possible. Until the cached CRL expires, the client will consider the certificate as not revoked, even if it has already been revoked. This information simply did not yet propagate to the client because the cached CRL has been used. Thus, the client will get a different result for revocation checking the same certificate twice, even if checked with respect to the same time. This non-deterministic behavior can be a show-stopper for certain applications; for instance consider applications where relying parties transfer noticeable volumes of money based on a positive signature verification (including a certificate validation with revocation checking).

Usually, administrators would switch to a new certificate even before they revoke the old certificate. As long as there has not been a key-compromise, revocations may not be that critical anyway. These cases can often be handled by organizational means. For example, the certificate can be updated, or the server could be replaced by a backup system. In case of a key compromise, however, the administrator would need to revoke the certificate immediately and inform all clients quickly. A delay of several days appears intolerable. CRLs do not provide an efficient solution in this situation. Reducing the caching time of CRLs does not offer a significant improvement either. The client may download the latest CRL for each certificate validation issue, but this is impractical. It wastes bandwidth and introduces delays, which are unacceptable for many applications. Moreover, the server which provides the CRL would break down under the load if all clients would download the current CRL every time, at least in environments with a considerable number of certificates and entries in the CRL. The waste of bandwidth in such configurations would be enormous.

Almost all data in a CRL is of no interest for the client. This is easy to see. The client is usually only interested in the status of a single certificate. The CRL returns a list of all revoked certificates in which the client can look up the concerned certificate. In most cases, the certificate of interest will not be listed in the CRL. Revocation is more an exceptional case than a common case. Common rates for certificate revocation are 5 to 10 percent, i.e. about 5 to 10 percent of all issued certificates get revoked before they expire. This would mean that a client would find the certificate of interest on the CRL in at most 1 of 10 (10 percent) or 1 of 20 (5 percent) cases. *At most* because a revoked signature or authentication certificate will typically only be used after revocation in case of a key compromise. If the certificate owner loses the private key, the certificate is useless and cannot be used any longer. Likewise, the legitimate owner would not use a certificate which has been revoked because of changed owner information. Key compromise will naturally be a

scarce reason for revocation compared to key-loss and change of subject information. As a result, a CRL will contain the concerned certificate even less frequently as calculated before. In effect, during a single certificate validation procedure, a CRL carries only a single bit of information for the client. The client wants to know if the considered certificate is valid (or was valid at a certain time). The answer can only be *yes* or *no* (*unknown* can be considered as a third alternative). Compared to such large CRLs as mentioned before, which contain over 20000 entries, the waste of resources becomes obvious.

The PKIX standards also define so-called Delta CRLs. Their main goal is to reduce bandwidth demand. Subsequent CRLs differ only slightly in their contents compared to their predecessor. Some new entries are added and some old may be removed, but the vast majority of the entries will stay the same between two subsequently issued CRLs. Thus, the idea is to transfer only changes in the CRL most of the time. A delta CRL refers to a complete CRL and just mentions the differences to this CRL, i.e. the added entries and the removed entries. Thus, the client has the same information as if it would have two complete CRLs. Delta CRLs are rarely used in practice. The added complexity is one of the reasons.

Revocation checking does only look that simple on the first look. If one implements the complete PKIX specification and considers all (or at least all practical) use-cases, revocation checking based on CRLs can get complex. Here are some reasons:

- There could be a different CRL for each revocation reason. In consequence, the client would have to check each of these CRLs.
- The issuer of the CRL could be different to the issuer of the certificate. It may be difficult to make a trust decision and no user wants to do this manually (most users do not even have the required background knowledge to do it).
- If the application has to verify a certificate (which may already be expired) with respect to a time in the past, it would need an old CRL. There is no standardized way to get an old CRL. The standards only specify how to get the latest CRL.

This list can be extended by various bullets. These cases do not frequently appear in practice, but they are perfectly valid according to the underlying standards. If a developer has to implement a system, which supports this standard, the system needs to be able to handle these special cases as well, even though they might never appear in practice. As a result, developers may end up with a final system where many parts of the certificate validation system are never really used.

For completeness, we should mention that there are use-cases where CRLs are sufficient and the required bandwidth is not that critical. If the CRL does not contain many entries, the lavished bandwidth is not that significant either. The size of the CRL can be decreased using different approaches. A simple one is to use partitioned CRLs. Using this method, the CA splits the set of all issued certificates into groups of similar size. The CA will issue an individual CRL for each group of certificates. Thus, the certificates, even though issued by the same CA, will point to a different CRL location, depending on the group they belong to. For CAs which issue only a small number of certificates, the CRL will generally stay small enough without segmentation.

OCSP

CRLs have been designed as an off-line mechanism for revocation checking. An off-line mechanism cannot meet certain requirements for timeliness and efficiency. So what about on-line protocols? Does OCSP perform better? It may. At least, it does not waste that much bandwidth in most cases.

The basic idea of OCSP is simple. It is an on-line service which provides revocation information about certificates to clients. When the client needs to validate a certificate, it can send a request to an OCSP responder. The OCSP server answers with information about the revocation state of the certificate. Unfortunately, there are some details which can make OCSP harder to use than necessary. First, the client needs the issuer certificate of the certificate of interest because the hash of the issuer certificate's public key is required in the OCSP request message. Second, the response of the server may not be as concrete as the client would need it to be. A normal OCSP response does not provide positive certificate status information. OCSP responses have several properties which degrades their effectiveness in many use-cases:

- The response does not say if the certificate is valid, it merely says that the certificate is not revoked. However, this can even mean that the certificate has never been issued.[1]
- The revocation information provided by an OCSP server may not be up to date.

At a minimum, a client cannot expect to get more information from an OCSP responder, as it can get from the latest CRL. This is nice for OCSP service providers because it allows very simple implementations of an OCSP responder. In this case, the OCSP service is nothing more than a front-end for the latest CRL. Consequently, an OCSP service implemented this way inherits the problems inherent to CRLs except the bandwidth consumption. Remarkably, the bandwidth consumption of CRLs is the biggest cost factor for big CAs as documented in [11].

Revocation Checking in Practice

In practice, many systems do not perform revocation checking at all. This has various reasons. Most web browsers do not check server certificates for revocation. The main reason seems to be the potentially long delay due to downloading the CRL. Many email clients often perform no checking per default either. The long delay seems to be the reason once again. However, most applications at least offer options to enable revocation checking. Some of them, for instance Microsoft Outlook 2002 or later, perform revocation checking per default. It is easy to identify such email clients with enabled revocation checking because there often is a long delay if the user opens a signed or encrypted email. Downloading the CRL takes most of this time.

Only a few systems support OCSP. Mozilla is one of these few exceptions. Most web browsers and email clients do not support it, at least not without additional third-party tools. On the CA side, the situation is similar—not many of them run OCSP servers. Those who run an OCSP server often implement them based on CRLs rather than to base them on current status information. Third-party OCSP responders usually have no other means than to base their response information on CRLs. Third-party OCSP responders have the advantage that

[1] The German ISIS-MTT [12] profile defines the CertHash extension for OCSP responses. A response containing this extension can provide a positive status information.

they ean provide revocation information for certificates from various CAs. On the other hand, users must configure their clients manually to use external responders. In sensitive environments, it might be impossible to rely on another external party; additional contracts have to be prepared, liability issues have to be clarified, and so on.

OCSP is capable of providing a better service to clients than CRLs can do. For the provider side, OCSP requires additional resources. Since OCSP is an on-line service, it requires a reliable server system with a high level of security. The server has to sign its responses with a private key. This key requires the same security level as the key for CRL signing does because it serves to protect the same kind of information—revocation information for the same certificates. Some simple investigations and calculations show that OCSP shifts the demands from network bandwidth to processing power.

Calculations can show some values for the performance estimation of CRLs and OCSP for revocation checking. First, we sketch rough values for the required bandwidth in certain environments. In addition, we consider the number of required signature creations on the server, which differs significantly between CRLs and OCSP. On the client side, CRLs and OCSP have also others impacts. If a client uses CRLs, it needs to cache CRLs. Without caching, CRLs are a real performance and resource killer because the client would download the complete CRL for each revocation checking. Some implementations do this intentionally to ensure that they have the latest available revocation information. This practice cannot be recommended in general. If there is a demand to get always the latest revocation information, OCSP is a better choice. In addition, processing a CRL is usually more costly than processing an OCSP response, unless the CRL is very small which is typically not the case. Verifying the signature of the status information—the signature on the CRL or on the OCSP response—is effectively the same effort for the client. The timeliness of the status information may also differ between CRLs and OCSP. An OCSP responder may provide real-time revocation information, while a CRL always provides aged information, though, many OCSP servers do not provide more current information than the CRLs do.

	Small CA	Big CA
Certificates	1 000	1 000 000
Certificate Users	1 000	1 000 000
Certificate Validity [years]	1	1
Revocation Probability	10%	10%
Certificate Validations per Certificate [day^{-1}]	10	10

Table 1: Properties of the considered configurations

Table 1 shows the properties of two CA configurations which we will use as a basis for performance estimations. There are two configurations—a small CA configuration with thousand issued certificates, and a big CA with one million issued certificates. We set the validity period of the issued certificates to one year, which is a typical value. In addition, we assume that the number of certificate users (users who validate certificates from this CA) is roughly the same as the number of issued certificates. However, these two numbers can vary significantly in practice; a CA issuing only server certificates for a few big web servers like Amazon, eBay and Google would have a huge number of users validating the certificates while the number of issued certificates is small. For the probability of certificate revocation, we take 10%, i.e. the probability that a certificate will be revoked before it expires. This is

also a quite common value in practice, but it may even be much lower, e.g. one percent. We assume, that a user validates about 10 certificates per day, for example, verify 5 signed emails and encrypt 5 emails per day. These numbers set the frame for our short investigations. They are typical values, even though there are configurations for which these values would look very dissimilar and would result in quite different performance estimations.

	Small CA	Big CA
CRL Validity [hours]	24	24
Signature Creations [day^{-1}]	4	3 014
CRL Size [Byte]	2 250	2 000 250
CRL Downloads [day^{-1}]	1 000	1 000 000
Bandwidth [MB/day]	2,1	1 900 000

Table 2: Expected CRL Performance

Table 2 shows the expected performance data for the two CAs using CRLs for revocation checking. We took a CRL validity of 24 hours. This is the time between the *thisUpdate* and *nextUpdate* values in the CRL. The number of signature creations required on the CA side is comprised of certificate and CRL issuing—one signature per each certificate and each CRL. For instance, the small CA issues 3 certificates per day on average and 1 CRL, which requires 4 signature creations in total per day. Here, the certificates account for most of these signatures because the CA issues CRLs infrequently. For calculating the CRL size, we took 40 Byte as the size of each CRL entry[2]. Despite the entries of the revoked certificates, each CRL contains additional data like the signature, *thisUpdate*, *nextUpdate* and the signer name. For this additional data, we assumed 250 Byte. If we take the small CA, for example, the CRL would contain 50 entries on average[3], which will result in a size of 2250 Byte (50.40 + 250). The total number of certificates and the probability of certificate revocation determine the number of entries. Taking the 10 certificate revocation checks per user and per day, every user downloads each issued CRL. This ends up in a network transfer volume of 2,1 MB per day for the small CA and 1900 GB per day (about 22 MB per second) for the big CA. This is a noteworthy volume. Only big companies can afford such a network bandwidth, and often merely in a local network and hardly via the Internet. It is also a matter of costs. Would you afford this for revocation checking only if there are cheaper alternatives?

Moreover, these numbers can easily increase. The current numbers are based on an issuing interval of one day. In the worst case, clients get one-day-old revocation information. If the managers of a system decide to increase the CRL issuing frequency to get a better timeliness, the increase of network transfer is inverse proportional. Reducing the CRL validity to the half (i.e. to 12 hours) doubles the transfer volume. The peaks of bandwidth consumption may be even much higher because we assumed evenly distributed download requests over time. In practice, peaks are very likely, for instance, in the morning when people start working there will be a peak, while the download rate will drop during the night.

[2] This consists of at least 13 Byte for the revocation date, plus the serial number (e.g. a SHA-1 hash value), plus an overhead for the DER encoding of at least 6 Byte. In addition, there may be extensions for the reason code or invalidity date.

[3] We get 100 entries in the worst case, which occurs if the CA issues all certificates at the same time (e.g. at the beginning of the year). If it issues the certificates continuously over the year, we could expect 50 entries in each CRL, because on average a certificate would be revoked after half of its validity period.

	Small CA		Big CA	
Base CRL Validity [days]	7	7	7	7
Delta CRL Validity [hours]	24	2	24	2
Signature Creations [day^{-1}]	4	15	3015	3025
Base CRL Size [Byte]	2250	2250	2000250	2000250
Avrg. Delta CRL Size [Byte]	288	288	38606	38606
Base CRL Downloads [day^{-1}]	143	143	142857	142857
Delta CRL Downloads [4][day^{-1}]	1000	10000	1000000	10000000
Bandwidth [MB/day]	0,58	3,1	310 000	640 000

Table 3: Expected Delta CRL Performance

Delta CRLs can reduce the bandwidth consumption significantly. Table 3 shows that the CA can issue delta CRLs even more frequently while the bandwidth consumption is still much lower than for full CRLs. To get comparable results, we included a delta CRLs case with the same timeliness constraints, which we had for the full CRL case; i.e. there is one column for the small CA and for the big CA for which the delta CRL validity is 24 hours—the same value as the CRL validity in Table 2. To get an impression how the results would change due to a reduction of the delta CRL validity, we added another column for each CA with smaller delta CRL validity—two hours in this case. In contrast to the full CRL case, with delta CRLs, the CA has to create a similar number of signatures. Only a few delta CRL signatures are required additionally, while the number of signatures for base CRLs decreases at the same time because the CA issues them only once a week.

As a result, we can see that the bandwidth consumption decreased significantly. In the case where delta CRLs are issued once a day—which provides the same timeliness as the full CRL case considered before—the transferred data volume dropped to 0,58 MB per day for the small CA and to 310 GB per day for the big CA. This is a reduction of about 73% and 84% respectively. Even if we reduce the validity period of the delta CRLs (and increasing the timeliness of the information at the same time) in the Big CA setup, the load on the network remains lower than with full CRLs. Only for the small CA, the network load is higher than in the full CRL case, 3,1 MB per day compared to 2,1 MB per day.

	Small CA	Big CA
Signature Creations [day^{-1}]	10 003	10 003 014
Request Size [Byte]	250	250
Response Size [Byte]	1 250	1 250
Bandwidth [MB/day]	14,3	14 305

Table 4: Expected OCSP Performance

Table 4 gives a performance estimation for an OCSP responder for our two CA configurations. This table is simpler because there is nothing such as a validity period. Even if the OCSP responder gets its status information from CRLs, it makes no performance difference, which would reflect in the table. If the OCSP responder gets its status information from CA database directly instead of from CRLs, the CA could abandon CRLs completely. In addition, it would enable the OCSP responder to provide real-time status information and positive status responses (this requires a non-standard extension like the CertHash extension in [12]). In contrary to the CRL case, for OCSP the bandwidth consumption

[4] Note that the number of delta CRL downloads per day will not be higher than the overall number of certificate validations per day, which is the product of certificate validations per day (per certificate) and the number of certificates.

increases linearly with the number of certificate validations. With CRLs, the client downloads a CRL and does all revocation checking based on it until it expires. With OCSP, the client has to get a separate OCSP response for each certificate validation. The caching of responses gives no benefit unless the client always validates the same certificates.

As we can see in the table, the bandwidth consumption is quite affordable. While it is a little higher for the small CA, it is smaller by an order of magnitude for the big CA. For the small CA, the OCSP responder produces 14,3 MB traffic per day compared to 2,1 MB with full CRLs and 0,58 MB and 3,1 MB with delta CRLs. In large environments, the difference is tremendous. Here, the OCSP responder causes about 14 GB network traffic per day compared to 1900 GB with full CRLs and 310 GB and 640 GB with delta CRLs. In addition, an OCSP responder can give real-time status information—CRLs cannot. These 14 GB per day (0,17 MB per second) are an affordable bandwidth, not only for a corporate network but also for Internet connections.

These simple calculations already give a clear impression of the scalability of the different revocation mechanisms used in practice. OCSP scales much better than CRLs. As well, OCSP can provide more current revocation information than CRLs. For small CAs, CRLs may cause slightly less network traffic, but also provide less timely status information. OCSP can show its advantages in large environments, where it can reduce the network traffic massively.

However, all these calculations are valid only for two simple but reasonable setups. The results can vary depending on various factors of the environment, including the number of issued certificates, the number of certificate users, the number of certificate validations and the timeliness requirements for revocation information. Up to now, timeliness requirements seem to have a low priority for CAs because there are many CAs which do not offer OCSP responders and those of which who implement their servers to provide no more current revocation information than the latest CRL does. It is unclear if there is not enough demand from the customers for up-to-date status information, or if the CAs are reluctant to implement better OCSP servers.

Proposal for an on-line PKI

Certification Authorities issue certificates. Users who receive such a certificate need to find out, if the data in the certificate is valid with respect to a certain time. For digital signatures, this is the time of signature creation. In case of encryption, it is usually the current time. The application performs revocation checking to find out if the certificate is valid. Other steps are also required, but applications can typically process them with locally available data. Certificate chain building, for example, is often rather easy because most protocols like secure email and SSL/TLS recommend sending the complete certificate chain anyway. Fortunately, most implementations stick to this recommendation.

Is revocation checking always required? If we assume that the start of the validity period of a certificate corresponds to the time when the certificate has been issued, an application would not need to check a certificate for revocation with respect to this time. In fact, a certificate does not say anything about its validity. Even the validity period is only defined to specify the time interval during which the issuing CA guarantees to provide status information. Currently, there are new standards for certificate validation on the way. One of the most prominent ones is the Simple Certificate

Validation Protocol, short SCVP. The PKIX working group develops this standard. It should enable clients to off-load the complete task of certificate validation. The client sends a request to the service for validation of a certain certificate. As a result, it gets the validation result.

Developers with no PKI knowledge may ask, if we need certificates at all. This is a legitimate question. The client has to trust the validation server anyway to provide correct results, and it does not do any processing of the certificate itself any longer despite picking the public key and the identifier of the key owner. In many cases, it would be sufficient to add the signer's name and the signature verification key to a signature. The verifier could then ask the validation server if this signer was the legitimate owner of the verification key at the time of signature creation. The validation server could still employ certificates for its internal processing and select the appropriate ones. However, if the CA runs the validation server, there seems to be no point in using certificates in the traditional way. Accessing databases seems to be simpler and more efficient. The signer could add a link to the certification authority to the signature to support the client in selecting the right one for validation. It may even make sense to have more than one authority at the same time. The recipient can pick a trusted one or may validate with several or all of them to increase the trust in the result. It would also be possible that the signer already gets a signed validation result from the authority at the time of signature creation. The validation result is actually nothing else than a certificate. The recipient can still do its own validation on demand. Thus, the same format can be used. All we need is an on-line service for certificate issuing from the CA. The requirements for such a service would be pretty much the same as for an OCSP service. The CA would issue a new certificate for each signature, or at least for each time instant at which the signer creates a signature. An important difference is that the validity period of such certificates would always be in the past (with respect to the issuing time) or would at most extend to the current time.

This approach has several advantages. First, there is no need for additional revocation checking. The issued certificate would already transport all information to the recipient—the signer was the legitimate owner of this key at the signing time. For encryption, the time of interest is the current time, and the sender would go for a certificate for the intended recipient.

One apparent disadvantage is the required on-line access for the client. Having a closer look, the requirements are the same as if the user would have to perform revocation checking with up-to-date status information. This is also only possible with on-line access. Users and applications can still use old certificates if they do not have such high security demands. Here, the decision is up to the users if they accept out-dated information about the binding between alleged key owner and key or if they go for reliable and definitive information.

An additional improvement is the reduced complexity. Only one format is required for certificates, no additional format for CRLs or OCSP or any other validation service. The existing certificate format could even serve as the request format for this on-line certification service. The request would not need a signature though. The response would be just a certificate. The validity period of this certificate would be just a single time instant or maybe a period in the past. For the past, the certification authority would be able to provide definitive answers. Thus, revocation may not be required at all, in the sense it is used today.

Business cases would also be simpler and more flexible with this approach. A commercial CA would be able to charge on a per-use basis, for example, per issued signing certificate. Since the CA does not need to run an additional revocation checking service, the resource planning is also easier and more predictable. Just imagine that Microsoft enables certificate revocation checking using CRLs per default in their Internet Explorer. Commercial CAs would face tough times providing their large CRLs for millions of clients in the world.

The workflow for setting up a relation or contract between the user and the CA can remain the same. The CA can issue a first certificate during this initial phase. Even though the client does not really need this certificate, it may help ensuring compatibility with existing applications. The client could even use just a self-signed certificate instead because it also includes the required public key and name. Moreover, a self-signed certificate can serve as a proof of possession during the registration. If the client registers its public key at several CAs, it would not need to have several certificates. There is nothing wrong with the registration of the same user-to-key binding at several CAs. It would only cause different CAs to vouch for the same statement. In fact, this is an increase of security because it may be possible that a single CA makes mistakes, but it is less likely that several independent CAs make the same mistake.

The client actually needs just its identifier (something like the subject name) and its key-pair. For further communication with the CA, it can use this signature key and the identifier to authenticate the requests to the CA. A request can be for lengthening the contract with the CA or for revocation of a key-to-identifier binding. Note that there is no revocation of a single certificate any longer because there is no certificate to revoke. After revocation of the binding, the CA would simply no longer issue certificates via its on-line certification service for this combination of key and user. There is no need to revoke any old certificates it issued before because the certificate statements refer to a time in the past before the revocation took place.

What is the performance of such a simplified on-line certification PKI? The bandwidth consumption is actually the same as for OCSP, assuming that the requests and responses are roughly of comparable size. This seems to be a reasonable assumption.

	Small CA	Big CA
Signature Creations [day^{-1}]	10 000	10 000 000
Request Size [Byte]	500	500
Response Size [Byte]	1 000	1 000
Bandwidth [MB/day]	14,3	14 305

Table 5: Expected performance for on-line certification

Table 5 shows the expected performance values at the CA side. The number of required signature creations is even slightly lower than for OCSP because for OCSP the CA has to issue certificates and OCSP responses. Here, we only have one signature for each certificate, the total number of which is the same as for OCSP responses. In practice, the situation may be even better because clients typically do not cache OCSP responses. Thus, they get an OCSP response each time they validate a certificate, even if they validate the same certificate with respect to the same time. In the case of on-line certification, when the signer already attaches the certificate to the signature, the verifier may not ever need to go for a new certificate.

For the big CA, the on-line certification service needs to perform 10 million signature creations per day, which are about 116 signature creations per second. This is an amount, which modern server systems can easily process with pure software implementations of cryptographic operations[5]. An on-line certification system would use secure crypto hardware anyway for sheer security requirements. Modern hardware security modules (HSMs) are typically able to perform several hundred operations per second. There are some HSMs available which can create several thousand signatures per second,[6] and there are crypto processors which can perform several ten-thousands per second[7]. Thus, even for large CAs, an on-line certification service seems to be feasible with off-the-shelf hardware.

Compared with CRLs and delta CRLs, on-line certification is a clear tradeoff between network bandwidth and processing power. On small-scale PKIs, on-line certification can require even more bandwidth than CRLs, but in large-scale PKIs, it performs equally well as OCSP. At the same time, on-line certification can provide the latest key status information, which an OCSP responder can also do, but CRLs and delta CRLs cannot.

On the client side, on-line certification can bring a significant reduction of complexity because additional revocation checking is no longer required. Moreover, clients need to handle less different data formats. They only need to handle the certificate format, which can remain the same as already used in current systems. As a result, existing protocols like S/MIME [5] and SSL/TLS [4] can run without modification. Some other protocols can even be simplified because there is no need to support CRLs, OCSP and other revocation checking formats and protocols any longer.

The service's signature key is an attractive target for attacks. This is a drawback, even though it is manageable with dedicated security devices. If an adversary can get the signature key of the service, it can issue certificates at will. Current CAs often issue certificates off-line and just issue revocation information on-line. Getting the signature key of the revocation service is not sufficient for issuing certificates, though the ability to issue wrong revocation information may allow attacks with similar severe effects.

Conclusion and Further Work

During this work, we analyzed a few typical PKI setups. Not surprisingly ([11]), it turned out that CRL-based revocation checking consumes a lot of bandwidth, especially for large CAs with many certificates and many users. For the proposed type of on-line certification, the requirements for the signature creation performance are higher than for CRLs, but they are at most as high as for an OCSP responder. Unlike CRLs and certain OCSP responders, on-line certification provides always the latest status information. This approach also reduces the complexity of PKI systems because it eliminates the need for additional revocation checking, and in consequence, there is no need to support additional formats and protocols like CRLs and OCSP. In summary, the system requirements regarding bandwidth and processing performance are comparable to OCSP. Thus, existing hardware and software

[5] For example, OpenSSL 0.9.7d compiled with assembler optimizations on an AMD Opteron 146 with 2 GHz performs about 900 signature creations per second using RSA 1024 bit keys (using CRT).

[6] The Sun Crypto Accelerator 4000 PCI Card can create 4300 RSA signatures per second with 1024 bit keys using CRT.

[7] A NITROX Security Processor from Cavium Networks can perform up to 40 000 RSA signatures per second with 1024 bit keys using CRT.

components are sufficient to implement an on-line certification system.

On-line certification has some neat properties, which makes it appealing to certain business cases. A pay-per-use scheme seems to be easy to implement. Its behavior is also better predictable than that of current systems, which include separate revocation checking. Overall, the introduction of an on-line certification service is more of a technical nature than an organizational one. Thus, the existing organizational environment can often remain unchanged. Even many parts of the technical equipment can remain unaffected or may require only small adaptations. On the client side, the reduction in complexity is even more evident. This can make PKI attractive even for such environments where current PKI systems would entail an unacceptable increase of complexity.

The next step towards an on-line certification service would be the more detailed specification of a protocol. Such a service would have to be as simple as possible. A prototype implementation as a proof-of-concept would also help to get a clearer impression of the advantages and disadvantages of this approach. It would also allow benchmarking in different environments and use-cases. Measured values from real implementations are more convincing than theoretical calculations.

Two important issues have not been considered yet. The first is the setup of trusted root keys. Even though, the same techniques as for the setup of trusted root certificates are applicable, simpler and more elegant mechanisms are desirable. The second is the process of revocation itself, i.e. the actions the key owner can take to notify the CA about revocation. Current systems solve this issue unsatisfactorily.

While recent work often increases the overall complexity of PKI systems and thus diminishes its attractiveness, on-line certification can offer a real reduction of complexity while improving utility.

References

[1] Housley, R., Ford, W., Polk, W., Solo, D.: Internet X.509 Public Key Infrastructure, Certificate and CRL Profile. The IETF, RFC 3280, April 2002, available online at http://www.ietf.org/rfc/rfc3280.txt

[2] Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP. The IETF, RFC 2560, June 1999, available online at http://www.ietf.org/rfc/rfc2560.txt

[3] Adams, Carlisle, Lloyd, Steve, "Understanding the Public-Key Infrastructure", New Riders Publishing, ISBN: 157870166X

[4] Dierks, T., Allen, C.: The TLS Protocol, Version 1.0. The IETF, RFC 2246, January 1999, available online at http://www.ietf.org/rfc/rfc2246.txt

[5] Ramsdell, B. (Editor): S/MIME Version 3 Message Specification. The IETF, RFC 2633, June 1999, available online at http://www.ietf.org/rfc/rfc2633.txt

[6] Doyle, P., Hanna, S.: Analysis of June 2003 Survey on Obstacles to PKI Deployment and Usage. The OASIS Public Key Infrastructure (PKI) Technical Committee (TC), Version 1.0, 8 August 2003, available online at http://www.oasis-open.org/committees/pki/pkiobstaclesjune2003surveyreport.pdf

[7] Myers, M.: Revocation: Options and Challenges. Financial Cryptography 1998, Springer-Verlag Berlin Heidelberg 1998, LNCS 1465, pp. 165-171.

[8] Cooper, D.: A Model of Certificate Revocation. Proceedings of the Fifteenth Annual Computer Security Applications

Conference (ACSAC), pages 256-264, December 1999.

[9] Rivest, R.: Can We Eliminate Certificate Revocation Lists? Financial Cryptography 1998, Springer-Verlag Berlin Heidelberg 1998, LNCS 1465, pp. 178-183.

[10] Fox, B., LaMacchia, B.: Online Certificate Status Checking in Financial Transactions: The Case for Re-issuance. Financial Cryptography 1999, Springer-Verlag Berlin Heidelberg 1999, LNCS 1648, pp. 104-117.

[11] Berkovits, S., Chokhani, S., Furlong, J., Geiter, J., Guild, J.: Public Key Infrastructure Study, Final Report. Produced by the MITRE Corporation for NIST, McLean, Virginia, April 1994.

[12] Common ISIS-MTT Specifications for Interoperable PKI Applications from T7 & TeleTrusT. Version 1.1, 16 March 2004, available online at http://www.isis-mtt.org

[13] Kohnfelder, L.: Towards a Practical Public-Key Cryptosystem. Bachelor Thesis, MIT, May 1978.

[14] Ellison, C.: Naming and Certificates. Proceedings of the tenth conference on Computers, freedom and privacy, pp. 213-217. Toronto, Ontario, Canada, April 2000.

Delegation Issuing Service for X.509

D.W.Chadwick, University of Kent, Canterbury, CT2 7NZ, England

Abstract

This paper describes the concept of a delegation issuing service (DIS), which is a service that issues X.509 attribute certificates on behalf of an attribute authority (typically a manager). The paper defines the X.509 certificate extensions that are being proposed for the 2005 edition of X.509 in order to implement the DIS concept, as well as the additional steps that a relying party will need to undertake when validating certificates issued in this way. The paper also presents our initial experiences of designing a DIS to add to the PERMIS authorization infrastructure. The paper concludes by reviewing some of the previous standards work in delegation of authority and anticipating some of the further standardization work that is still required in the field of privilege management.

1. Introduction

The 2000 edition of X.509 [1] defines a Privilege Management Infrastructure (PMI) based on attribute certificates (ACs). Attribute certificates are very similar to public key certificates (PKCs) but hold privileges (as attributes) instead of public keys. In the X.509 PMI model, the root of the PMI is termed the Source of Authority (SoA), and subordinates are termed Attribute Authorities (AAs). Delegation of Authority passes down the chain from SoA to AA to subordinate AAs in much the same way as the authority to issue public key certificates passes down a PKI certification authority hierarchy from the root CA to subordinate CAs (see Figure 1A). A subordinate AA is given a set of privileges by its superior AA, and may delegate these further to subordinates (AAs or end entities). A subordinate who is not allowed to delegate further is termed an end entity. In the normal situation all privilege holders (AAs and end entities) are allowed to assert the privileges that are delegated to them.

However, in some situations a privilege holder may be allowed to delegate the held privileges to a subordinate, but may not be allowed to assert the privileges itself. An example might be an airline manager who assigns privileges to pilots to fly particular aircraft, but is not allowed to fly the aircraft himself, or a hospital manager who assigns privileges to doctors on duty but is not allowed to assert these privileges himself. Whilst the X.509 standard recognizes this scenario, it offers no support for this in the ACs that can be issued to these AAs i.e. there is no way of signaling to a relying party (RP) that an AC holder may not assert the privileges contained in the AC that it holds. This deficiency needs rectifying.

Work is now progressing towards issuing the 2005 edition of X.509, and another delegation scenario has been identified in the draft amendment [2] to X.509(2000). This concerns the use of a delegation service to issue ACs on behalf of other AAs. The delegation issuing service (DIS) concept recognizes that in some organizational contexts, it might be preferable for a manager (an AA) who wishes to delegate authority to a subordinate, be not empowered to issue the X.509 AC herself, but rather should request a DIS to issue the AC on her behalf.

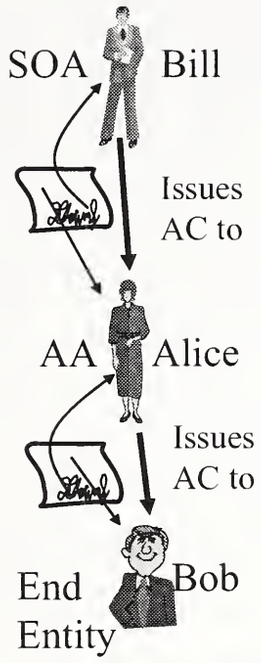


Figure 1A. Normal Delegation of Authority

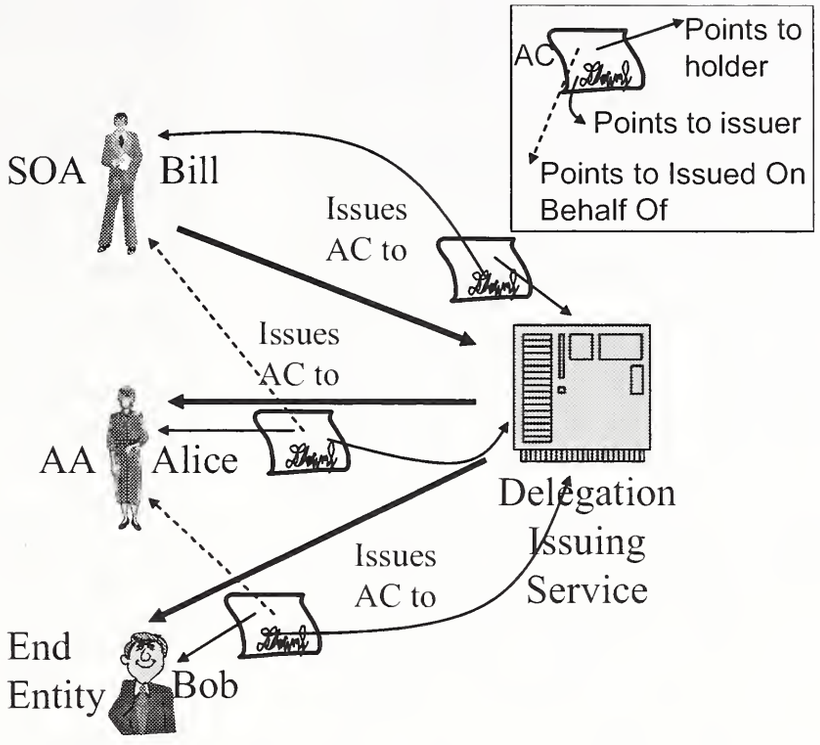


Figure 1B. Delegation of Authority using a Delegation Issuing Service

1.1 Advantages of a DIS

The benefits of using a delegation issuing service instead of AAs issuing X.509 ACs themselves are several. Firstly, the DIS can support a fully secure audit trail and database, so that there is an easily accessible record of every AC that has been issued and revoked throughout the organization. If each manager were allowed to independently issue their own ACs, then this information would be distributed throughout the organization, making it difficult or impossible to collect, being possibly badly or never recorded or even lost. Secondly, the DIS can be provided with the organization's delegation policy, and apply control procedures to ensure that a manager does not overstep her authority by issuing greater privileges to subordinates, or even to herself, than the organization's policy allows. Thirdly, the manager does not need to hold and maintain her own private signing key, which would be needed if the manager were to issue and sign her own ACs. Only the DIS needs to have an AC signing key. This could be a very important feature in organizations that use mechanisms other than PKIs for authentication such as biometrics, user names and passwords, or Kerberos etc. Finally, if the DIS is given its own AC by the SOA, it can replace the (set of) manager's AC(s) in the AC validation chain and therefore decrease the complexity of AC chain validation. The AC chain length would always be two when the DIS issues the ACs to end entities, whereas it would be of arbitrary length when the managers issue the ACs themselves. Also less CRLs will need to be issued – only the DIS will need to issue a CRL rather than each manager. This will further simplify AC chain validation.

1.2 DIS Deployment Models

Two deployment models for the DIS are envisaged in the 2005 draft amendment [2]. In both models the DIS is empowered to issue ACs on behalf of other AAs, by being issued with its own AC by the SoA. This AC confers on the DIS the authority to issue ACs on behalf of other AAs. This empowerment model is similar to the PKI concept of an indirect CRL issuer, whereby an entity is given the authority to issue CRLs on behalf of a CA. In the first DIS deployment model (which we have called the *AC PKI* mode), a privilege holder requests the DIS to issue privileges on its behalf, but the DIS does not actually hold the privileges itself. The AA tells the DIS which privileges to delegate. In the second deployment model, the DIS is actually given the privileges to be delegated by the SoA (we have called this the *PMI* mode). However, the 2005 draft amendment had no mechanisms for implementing either of these deployment models.

In our research and design of a DIS service for PERMIS [5], we have also identified a third deployment model in which the DIS is not given an AC, but has its own PKI key pair for signing the ACs its issues, with empowerment flagged in the public key certificate (we call this the *PKI* mode). The DIS now only needs to authenticate the AAs and issue ACs on their behalf, without validating the contents of the ACs. Furthermore, the users do not need to have their own PKI key pairs. This simplifies the design and deployment of the DIS service, but the downside is that it complicates the process of AC chain validation by the relying parties due to the delegation indirection introduced by the DIS, as described later.

1.3 Disadvantages of a DIS

As mentioned above, in PKI (and AC PKI) modes, AC chain validation is more complex when a DIS issues the ACs. Another potential disadvantage of a DIS is that the single CRL issued by the DIS could get very large, unless distribution points are used. A large CRL can adversely affect the performance of AC chain validation. Further, when cross certification between PMIs takes place, in PMI mode there is a loss of granularity since it has to be the DIS that is cross certified rather than any of the AAs that are trusted. But perhaps the biggest disadvantage of using a DIS for some applications is that the AC signing key should be online and ready to be used to sign ACs when requested. In some highly secure systems this would be unacceptable.

1.4 Paper Contents

This paper describes proposed extensions to the 2000 edition of X.509 that can be used to implement the DIS model for delegation of authority, as well as rectify the 2000 deficiency that there is no way to signal that an AA holds a privilege for delegation but is not allowed to assert the privilege. These extensions have recently been included as part of the revised draft amendment to X.509(2000).

The rest of this paper is structured as follows. Section 2 describes the X.509 extension that can be used to signal that a privilege holder is not allowed to assert the privileges that it holds. This corrects the deficiency in the 2000 edition of X.509. Section 3 describes the X.509 extensions that can be used to implement the DIS model. Section 4 describes how relying parties will need to use these new extensions in order to validate ACs issued by a DIS. Section 5 describes how we are implementing the DIS in PERMIS. Section 6 describes related standards work and

research in this area, whilst Section 7 describes further standardization work that is still needed to be done in the X.509 PMI framework.

2. No Assertion of Privileges

There are two scenarios where privilege holders may be given privileges in an AC[1], in order to delegate them to other entities, but where they are not allowed to assert the privileges themselves. The first is where a manager is tasked with allocating roles or duties to subordinates, but is not allowed to assert the roles or duties himself. The previous section gave a couple of examples of this scenario, in the shape of an airline manager and a hospital manager. This scenario is represented by Alice in Figure 1A. The second scenario is where a delegation issuing service (DIS) is given a set of privileges to delegate, as directed by the SoA. This is represented by the Delegation Issuing Service in Figure 1B.

We can prevent the holder of these privileges (Alice in Figure 1A and the DIS in Figure 1B) from asserting them by placing a "no assertion" extension into the AC issued to it. This extension will inform all relying parties that understand the extension that the AC holder is not allowed to assert the privileges contained within the AC. This extension obviously needs to be a critical extension, since any relying party that does not understand it, must refuse to accept the AC, rather than simply ignore the extension and allow the privileges to be asserted.

The "no assertion" extension is formally defined in ASN.1 [6] as:

```
noAssertion EXTENSION ::= {
        SYNTAX NULL
        IDENTIFIED BY { id-ce-
noAssertion } }
```

where id-ce-noAssertion is an object identifier (OID) assigned in X.509.

If present, this extension indicates that the AC holder cannot assert the privileges indicated in the attributes of the AC. This field can only be inserted into AA ACs, and not into end entity ACs. If present, this extension shall always be marked critical.

3. X.509 Extensions to Support the Delegation Issuing Service

As described in the Introduction, three deployment models have been identified for the DIS, two in [2], in which the DIS is issued with its own AC and we have termed the AC PKI and PMI modes, and one from our own research, termed the PKI mode, in which the DIS does not have its own AC.

3.1 DIS Empowerment

The Delegation Issuing Service (DIS) needs to be empowered by the SoA to issue ACs on behalf of other AAs. This is done by including an "indirect issuer" extension in either the PKC or the AC issued to the DIS by the CA or SoA respectively. The indirect issuer extension serves a similar purpose as the indirect CRL boolean of the issuing distribution point extension in PKI CRLs i.e. it gives authority to the DIS. The indirect issuer extension is formally defined in ASN.1 as:

```
indirectIssuer EXTENSION ::= {
        SYNTAX              NULL
        IDENTIFIED BY       id-ce-
indirectIssuer }
```

where id-ce-indirectIssuer is an OID assigned in X.509.

[1] We do not consider it sensible to issue privileges to AAs via the subjectDirectoryAttributes extension of public key certificates, since the AAs would not be allowed to delegate these privileges further by issuing additional PKCs, since they are not a CA.

The indirect issuer extension may be used by the relying party when validating an AC chain to check that the AC issuer was empowered to issue ACs on behalf of other AAs (otherwise anyone with a signing key could issue an AC and say it was authorized by an AA). Alternatively, it may be used by the DIS software at initialization time to check that it is empowered to act as a DIS.

The draft extension to X.509 states that the indirect issuer extension may be placed in either an AC or PKC containing the subjectDirectoryAttributes extension issued to a DIS by an SoA. In our research we have identified that this extension may also be placed in a PKC that does not contain the subjectDirectoryAttributes extension.

The presence of this extension means that the subject AA (the DIS) is authorized by the SoA to act as a proxy and issue ACs that delegate privileges, on behalf of other delegators. This extension is always non-critical, since it does not matter to a relying party if it understands this extension or not when the DIS is acting as a privilege asserter by presenting this to the RP to assert the privileges contained within this certificate. This extension can be used by a RP when validating an AC chain which has the DIS acting on behalf of another AA somewhere in the AC chain (see section 4).

3.2 Requesting an AC

When an AA wishes to delegate some of its privileges to a subordinate, and wishes to use the services of a DIS to issue the AC on its behalf, it needs to contact the DIS to request the certificate to be issued. How this communication is achieved is outside the scope of X.509. Some discussion of this is provided later. Assuming this communication is successful, i.e. that the AA is authenticated to the DIS, and is allowed by the DIS's attribute allocation policy to request the AC to be issued, the DIS will issue an AC on behalf of the requesting AA. Thus we need an extension to be inserted into the AC, informing all relying parties that this certificate was issued on behalf of a particular AA. This leads to the requirement for the "issued on behalf of" extension, which is formally defined in ASN.1 below.

```
issuedOnBehalfOf EXTENSION ::= {
        SYNTAX GeneralName
        IDENTIFIED BY id-ce-
issuedOnBehalfOf }
```

where id-ce-issuedOnBehalfOf is an OID assigned in X.509.

This extension is inserted into an AC by an indirect issuer. It indicates the AA that has requested the indirect issuer to issue the AC, and allows the delegation of authority chain to be constructed and validated by the relying party if necessary (see section 4).

The GeneralName is the name of the AA who has asked the issuer to issue this AC

The issuer of this AC must have been granted the privilege to issue ACs on behalf of other AAs by an SOA, through the indirectIssuer extension in its AC or PKC.

This extension may be critical or non-critical as necessary to ensure delegation path validation (see next section).

4. Validating Indirect AC chains

The X.509 standard already provides a procedure for validating privilege paths and delegation chains in the standard delegation of authority scenario. This chain is represented by the curved arrows that point to issuers in Figure 1A. This procedure needs to be enhanced when indirectly issued ACs are encountered in the delegation chain,

such as those in Figure 1B. As can be seen from the addition of the issuedOnBehalfOf arrows in Figure 1B, the procedure is more complex and more delegation links need to be validated when this extension is marked critical.

Three deployment models have been identified, which we have termed the AC PKI, PMI and PKI modes. In PMI mode, the DIS has been issued with an AC by the SOA, which contains a superset of the attributes that it will issue on behalf of other AAs. This model presents the simplest path validation processing, since the AC chains will always comprise of just two ACs: the end entity's AC signed by the DIS, and the DIS's AC signed by the SOA. The existing standard path validation procedure will work for this AC chain. The RP may safely ignore the issuedOnBehalfOf and indirectIssuer extensions which will be marked as non-critical, since the DIS had full authority to issue the ACs to the end entities even though in reality it was a peer AA that asked for the delegation to be performed. Note that the DIS might not have permission to assert these privileges itself, but that will be signaled separately by the noAssertion extension.

In AC PKI mode, the DIS has an AC containing the indirectIssuer extension, but does not have any of the attributes that it will issue to others. These are held by the AAs that request the DIS to issue the ACs. In this case the issuedOnBehalfOf extension must be set to critical, since the RP will need to validate that the requesting AA had sufficient privileges to delegate to the end entity. If the extension was not set to critical, the RP is likely to compute that the AC chain is invalid since the DIS issuer did not have a superset of the privileges that were allocated to the end entity.

In PKI mode, the DIS does not have an AC, but only has a PKC containing the indirectIssuer extension. In this case the ACs issued by the DIS have to have the issuedOnBehalfOf extension set to critical, since the DIS is incapable of performing any validation of the requesting AA other than authenticating that it is who it says it is. All PMI validation has to be done by the RP. But this is in fact little different to the validation performed in the AC PKI mode, and is if anything slightly simpler since the DIS only has a PKC to be validated and not a PKC and an AC.

In addition to the standard procedural tasks of validating signatures and revocation lists, the relying party will also have to perform the following additional steps.

i) Starting with the end entity's AC, the RP will need to extract the issuer name and look at the critical flag of the issuedOnBehalfOf name.

ii) If the issuedOnBehalfOf extension is marked critical, the RP retrieves the ACs of the issuedOnBehalfOf AA and validates that the AA has a superset of the privilege attributes issued to the end entity and that the ACs have not been revoked. If it does not have sufficient privileges, or they have been revoked, the end entity's AC is rejected. The RP retrieves the certificates (ACs and PKCs) of the issuer and validates that the issuer is an indirect issuer of the SoA (i.e. has the indirectIssuer extension in one of its certificates). If not the end entity's AC is rejected.

iii) If the issuedOnBehalfOf extension is missing or non-critical, the RP retrieves the ACs of the AA (the DIS) and validates that the AA has a superset of the privileges issued to the end entity. If not, the end entity's

AC is rejected.

iv) For each AC of the issuer that contains one or more of the delegated privileges, the RP recurses to step i) for each AC, thereby moving up the delegation chain. This recursion continues until the RP arrives at the AC of an AA that is issued by the trusted SoA(s) who is(are) the root(s) of the PMI. This validates that the privileges were properly delegated.

4.1 Validating the noAssertion extension

If an AA's certificate has the noAssertion extension in it, what is to stop the AA issuing another AC to itself and omitting the noAssertion extension? Clearly there is nothing to stop this from happening. For this reason, SPKI decided (in section 4.2 of [11]) that they were powerless to stop this in their simple certificates that tied authorizations to public keys. However, X.509 has the advantage that AAs are given globally unique names by CAs. Providing an AA is not able to obtain an alias name for itself from the same or another trusted CA then the relying party can check if any AA's AC in a certificate path has the noAssertion extension set, and if it does, apply it also to any subordinate ACs that contain the same holder name. Clearly if an AA is able to obtain totally unrelated aliases from one or more trusted CAs, then the RP is unlikely to know that the AA is asserting privileges that it was not intended to, by using an alias name.

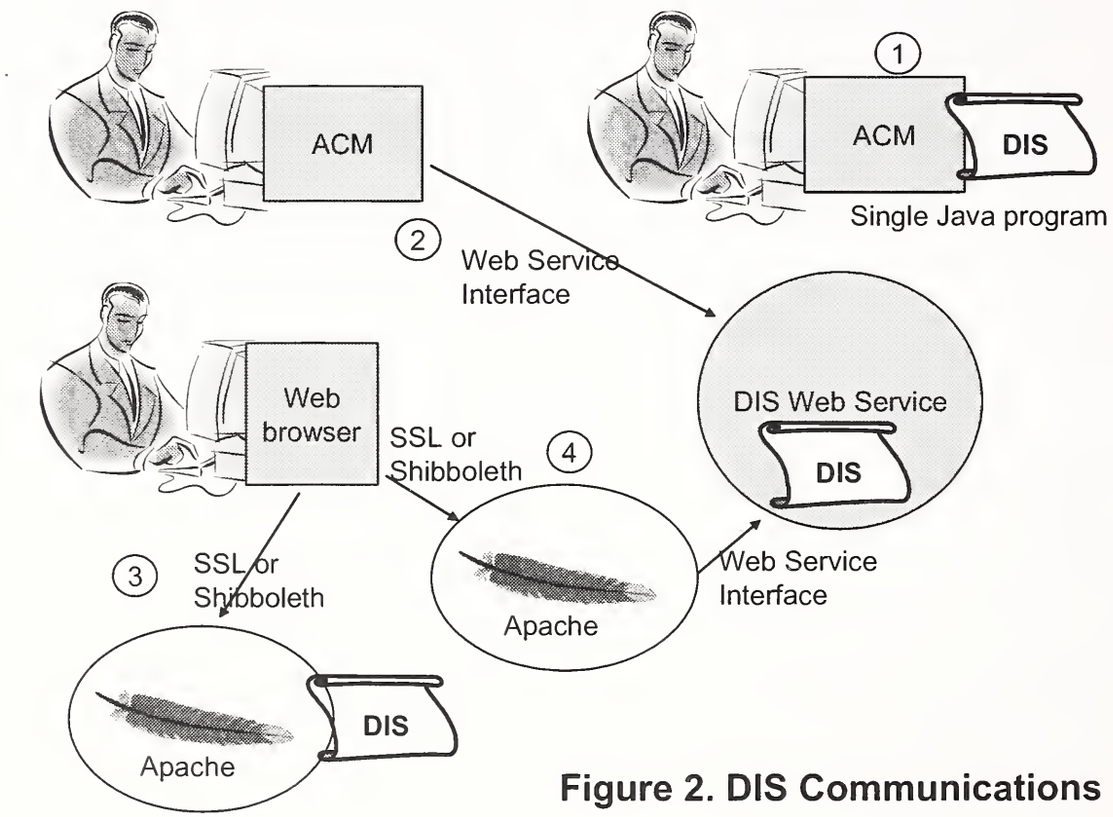


Figure 2. DIS Communications

5. Implementing the DIS

We decided to implement the DIS as part of the PERMIS X.509 PMI, as an aid to implementing dynamic delegation of authority. However, a number of issues needed to be resolved that are not addressed in the proposed extensions to X.509.

Firstly there is no mention of how the communication between the DIS and the AA should be achieved. Clearly the use of a standardized protocol is preferable to a proprietary one. One can envisage that an IETF working group such as PKIX might define a protocol similar to CMP [3], using a request similar to a PKCS#10 message [4]. In the absence of such a standard, in our own research we are proposing to use a Web Services protocol (see ② in Figure 2), and the Java GSSAPI [19] for authenticating the user. The GSS tokens will then be base64 encoded and inserted into SOAP messages. We are also defining a Java API for the DIS (see ① in Figure 2), so that the DIS can be built into other Java programs such as the PERMIS Attribute Certificate Manager (ACM) and called directly by it. In this case user authentication is not necessary. We are also proposing to adopt a 3 tiered model where an Apache server acts as the DIS client, authenticates the AAs via either Apache authentication (e.g. SSL) or Shibboleth (see ④ in Figure 2), and then acts as a proxy for them to the DIS. It would also be possible for Apache to directly call the DIS via our Java API (see ③ in Figure 2).

Secondly there are a number of issues concerned with AC path validation. As pointed out by Knight and Grandy in [18] this can be extremely complex when dynamic delegation of authority is implemented. We want to simplify this process as much as possible. We have already taken the step of not issuing role specification ACs, and instead we store their contents in each target's PERMIS policy read in by its PDP at initialization time. We thus only issue role allocation ACs. Our preferred DIS deployment model is the PMI mode, since the DIS is issued with a role allocation AC containing a superset of the attributes that it can delegate. In this way we limit AC path lengths to two, and if the target policy is willing to trust the DIS as a root (as well as the SoA) then path validation lengths are reduced to just one AC, that of the end user.

In our implementation, the DIS is given an AC containing a full set of privileges, and is configured with its own PERMIS PDP. The PDP is configured with an attribute (or role) assignment policy (RAP) [7], so that it can validate the AA requests. At initialization time the DIS will check that its AC has the indirectIssuer extension in it, otherwise it will fail to start. When an AA asks for an AC to be issued, the DIS will check that the AA is allowed to do this under the RAP policy, and also that the set of attributes requested are a subset of those held by the DIS. Validation against the RAP is done by the existing PERMIS PDP code. It is passed the (unsigned) AC requested by the AA, and it validates the credentials in the AC against the RAP. The only modification needed to PERMIS is to provide it with a null signature validation object that will return *signature valid* to every request to validate the unsigned ACs. If the AC passes the policy, the DIS will check that the requested attributes are a subset of those it holds in its own AC. The task of the RP is now made much simpler, since it only needs to validate 1 or 2 ACs, that of the user issued by the DIS, and optionally that of the DIS issued by the SOA.

Finally we wanted to simplify the use of PMIs in organizations that do not have fully functional PKIs implemented. These organizations, which are in the majority, already have a fully functional user authentication mechanism, and only have a handful of PKCs, e.g. web server certificates. It is for this reason that we have chosen to implement communications between the user and DIS as a three tiered

model via an Apache web server as in path ④ in Figure 2. This will allow organizations to use their existing authentication method. One problem that has to be solved is that of proxying, since the DIS will authenticate Apache, Apache will authenticate the user and Apache will then ask for an AC to be issued on behalf of the user. The DIS has to know if Apache is authorized to proxy in this way. We could solve this in a couple of ways. We could configure the details (name address of Apache) into the DIS. Or we could issue Apache with its own AC containing a proxy privilege. When Apache authenticates to the DIS, the DIS will call the PERMIS PDP to validate Apache's AC, and if it has the proxy credential the DIS will allow it to request ACs be issued on behalf of other AAs.

6. Related Research

Some of the earliest standardization work for attribute certificates and attribute certificate issuing servers was undertaken by ECMA in the late 80's and early 90's. This lead to the publication of ECMA Standard 219 [9] which specifies a model for distributed authentication and access control. The Privilege Attribute Certificates (PACs) described therein are a forerunner of the attribute certificates later standardized in X.509. A Privilege Attribute Server (PA-Server) is responsible for issuing PACs to users, and is similar in concept to the DIS described in this paper. However, to support delegation of authority between principals, new PACs are not issued to the delegatees (as in this paper) but rather the PA-Server provides the initial user with a PAC that contains one of more embedded Protection Values (PVs) that can be used for subsequent delegation. A PV is a hash of a secret Control Value (CV). The user is separately issued with the corresponding CVs. When a user wishes to delegate authority to another user or server, the latter is provided with the PAC and the appropriate CV (sent confidentially, of course). The delegate then presents the PAC to the target along with the CV. The target validates that the hash of the CV corresponds to a PV in the PAC, and if so allows the delegate to have the appropriate delegated access on behalf of the user. Different delegates can be given different CVs which authorize different subsets of the privileges contained in the PAC to different sets of target resources. The EC SESAME project [8] implemented a subset of ECMA Standard 219 and this was eventually rolled out into several commercial products from the project's partners. The disadvantage of the ECMA scheme is that the user has to know in advance of requesting the PAC what delegation is required, since this is built into the PAC at the time of its issuance.

ECMA Standard 219 supports both symmetric and asymmetric encryption for protection of the PACs, since it supports both Kerberos V5 [10] and digital signatures for user authentication to the authentication server prior to contacting the PA-Server. Interestingly, X.509 decided to standardize on only asymmetric encryption for its certificates, whereas Microsoft Windows decided to adopt Kerberos and symmetric encryption for its tokens when allowing users to gain access to multiple Windows services.

The Simple Public Key Infrastructure (SPKI) [11] IETF working group, whose work eventually merged with the Simple Distributed Security Infrastructure (SDSI) [12] of Ron Rivest, defined three types of certificate which mapped names, authorizations and group names respectively to public keys. Authorization certificates can be further delegated, and this is indicated by a Boolean flag set by the issuing delegator. The delegator can set the Boolean as

desired, except that if the Boolean is already false in the authorization certificate delegated to him/her then it cannot be switched back to true and be trusted. It therefore would be easy to apply the DIS concept and service to SPKI using the PMI mode deployment model, i.e. where the DIS is delegated an authorization certificate with the Boolean set to true. However it would break the theory of SPKI to implement either of the two PKI mode deployment models since these require the issuedOnBehalfOf extension to be present in the delegatee's certificate, and this would mean that the certificates are no longer *simple* according to SPKI's definition.

One feature included in SPKI that is not formally in the X.509 standard, is a rights language for expressing authorization policies. Consequently PERMIS defined its own policy language, written in XML [7]. SPKI uses S-expressions. X.509 has left it to other standards, e.g. the ISO Rights Expression Language [20] to specify the policies. This means that the policy rules by which a DIS operates are not specified in X.509.

Proxy certificates, defined initially by the Globus grid software developers, and later published as an IETF proposed standard [13], use a different model for delegation of authority. In this model a user, who is the subject of a public key certificate (and defined as an end entity by the X.509 standard), issues his own PKC (called a proxy certificate) to the public key of his grid job which has previously generated its own key pair. Validating the proxy certificate of course breaks the standard X.509 certificate path validation rules, since an end entity is not allowed to act as a CA. To rectify this, a critical extension (proxyCertInfo) is added to the proxy certificate to indicate the fact. The extension can also carry information about which privileges are being delegated, i.e. none, all or a subset, the latter being defined in an application specific way. Grid applications and users could use the DIS framework described here as an alternative to the latter, and ask the DIS to issue ACs to their grid jobs that contain a subset of the privileges contained in the user's AC. We plan to demonstrate this feature in due course, since PERMIS is already integrated with Globus toolkit [14].

More recently the work on Shibboleth [15] has implemented a limited mechanism for delegation of authority. In this case a target site delegates to the user's home site the task of authenticating and assigning attributes to the user. The user's privileges are returned to the target site in the form of a SAML Attribute Statement [16] signed by the home site. In research described in another paper at this conference [17], we have extended the Shibboleth delegation model by integrating it with PERMIS and X.509 ACs. The DIS will then be able to be used by home sites to delegate privileges even further.

7. Further Work

As indicated above, a protocol for interactions between an AA and a DIS will need to be standardized so that requests to issue ACs can be made in a standard manner. This request-response protocol may be similar to the PKIX CMP protocol, but it need not be, since proof of possession of the private key is not essential (indeed one of the motivations for having a DIS is that the users may not have their own key pairs). In many scenarios AAs may not be PKI users, but rather may use Kerberos, biometrics or symmetric tokens for authentication. In this case the AAs are computationally unable to issue X.509 ACs so will need to use the services of a DIS to issue the ACs on their

behalf. But they will be unable to sign those requests to the DIS. In this case a web services interface like the one we propose to use may be more appropriate, with the AA using a web browser to interact with the DIS via a web server, and perhaps authenticating with a username and password over an SSL link. Whatever protocol is standardized, it will need to be flexible enough to cater for the different environments in which it may be used.

Practical experience of working with X.509 PMIs is only just beginning. Most organizations who are experimenting with PMIs use them internally initially. They define their own privilege attributes and therefore the relying parties and SoAs have a common understanding of both the semantics and syntax of these privilege attributes. However, as organizations move towards role based or attribute based access controls, and federations between organizations, including the formation of virtual organizations, they will find that the attributes and roles they have defined are different from those in use by their federation partners. When this starts to occur, organizations will not want to re-issue ACs to users from the federated organizations, but rather will wish to understand and use the ACs that have already been issued. This will require cross certification between PMIs, the mapping of role allocation policies between organizations and constraints on what foreign users may asserts in home domains. It is anticipated that this work will form the bulk of the standardization activity for the sixth edition of X.509.

Acknowledgements

The authors would like to thank the UK JISC for funding this work under the DyVOSE project, and the NIST PC members who robustly reviewed this paper and made some constructive comments that helped to improve it.

References

[1] ISO 9594-8/ITU-T Rec. X.509 (2000) The Directory: Public-key and attribute certificate frameworks

[2] ISO SC 6 N12596 "Proposed Draft Amendment on Enhancements to Public-Key and Attribute Certificates", Geneva output, March 2004

[3] C. Adams, S. Farrell. "Internet X.509 Public Key Infrastructure Certificate Management Protocols". RFC 2510, March 1999

[4] Kaliski, B., "PKCS #10: Certificate Request Syntax, Version 1.5." RFC 2314, March 1998.

[5] 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.

[6] ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation

[7] D.W.Chadwick, A. Otenko. "RBAC Policies in XML for X.509 Based Privilege Management" in Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002), May 7-9, 2002, Cairo, Egypt. Ed. by M. A. Ghonaimy, M. T. El-Hadidi, H.K.Aslan, Kluwer Academic Publishers, pp 39-53.

[8] T.Parker, D.Pinkas. "Sesame V4 Overview", Issue 1, Dec 1995. Available from https://www.cosic.esat.kuleuven.ac.be/sesame/html/sesame_documents.html

[9] Standard ECMA-219 "Authentication and Privilege Attribute Security Application with Related Key Distribution Functions" Parts 1, 2 and 3, December 1994.

[10] J. Kohl, C. Neuman. "The Kerberos Network Authentication Service (V5)." RFC

1510, Sept 1993.
[11] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. "SPKI Certificate Theory". RFC 2693, Sept 1999.
[12] Ron Rivest and Butler Lampson, "SDSI - A Simple Distributed Security Infrastructure [SDSI]", See <http://theory.lcs.mit.edu/~cis/sdsi.html>.
[13] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson. "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile". RFC3820, June 2004.
[14] David W Chadwick, Sassa Otenko, Von Welch. "Using SAML to link the GLOBUS toolkit to the PERMIS authorisation infrastructure". Proceedings of Eighth Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Windermere, UK, 15-18 September 2004
[15] Scot Cantor. "Shibboleth Architecture, Protocols and Profiles, Working Draft 02, 22 September 2004, see http://shibboleth.internet2.edu/
[16] OASIS. "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) v1.1". 2 September 2003. See http://www.oasis-open.org/committees/security/
[17] David Chadwick, Sassa Otenko, Wensheng Xu. "Adding Distributed Trust Management to Shibboleth" presented at NIST 4th Annual PKI Workshop, Gaithersberg, USA, April 19-21 2005
[18] Knight, S., Grandy, C. "Scalability Issues in PMI Delegation". Pre-Proceedings of the First Annual PKI Workshop, Gaithersburg, USA, April 2002, pp67-77
[19] Charlie . Lai, Seema Malkani. "Implementing Security using JAAS and Java GSS API" Presentation from 2003 Sun JavaOne conference. See http://java.sun.com/security/javaone/2003/2236-JAASJGSS.pdf
[20] ISO/IEC 21000-5:2004, "Information technology — Multimedia framework (MPEG 21) — Part 5: Rights Expression Language [REL]". 2004

Meeting the Challenges of Canada's Secure Delivery of E-Government Services

Mike Just Danielle Rosmarin
Public Works and Government Services Canada
{mike.just, danielle.rosmarin}@pwgsc.gc.ca

Abstract

We describe a number of technical, legal, policy and business issues related to the development and deployment of Canada's PKI-based solution for authenticating individuals and businesses. We argue that tackling these issues is critical in order for Canada's PKI-based solution to truly transform Canada's e-government services delivery, and we offer insights into options that could resolve outstanding and remaining issues with this solution.

1 Introduction

Beginning in 1999, the Canadian Government initiated the Government On-Line (GOL) project, which included the development of an online presence for approximately 130 of the most frequently used government services. The project also included the provision of authentication services as part of the "Secure Channel" infrastructure, using public key credentials, giving departmental programs the means for properly identifying and controlling access to individuals' personal information. Since the initial goals for the GOL project have been accomplished, it is time to look to the future as more services move online, and more complicated transactions are supported.

While technology issues and decisions were paramount in the early years for GOL, many of today's issues reflect legal, policy and business concerns. The technology solutions are typically available or at least achievable; what remains is the deployment so as to best meet these other concerns. In this paper, we identify several current issues related to the secure delivery of services to Canadians, including interjurisdictional considerations, business registration, citizen enrolment, and evidentiary support for electronic data. In effect, we argue that tackling these issues is critical in order for Canada's PKI-based secure e-government service delivery to be truly transformative. Even though other jurisdictions' legislative and policy frameworks may differ from Canada's, the spirit will likely be similar so that other jurisdictions and solution builders should gain some valuable information from our experience.

2 Background

Before we discuss the current issues related to secure service delivery, we review the legislative and policy regime within Canada, and the technical solution supporting a secure e-government service delivery, the Secure Channel (SC) infrastructure.

2.1 Legislative and Policy Instruments

Politically, Canada is a federation in which the federal government is composed of approximately 130 departments and agencies. There are 10 provinces and 3 territories, each with their own departments and agencies. Numerous legislative and policy instruments govern their behaviour. In the federal government, each department mandate is captured in legislation. We will briefly describe the legislation and policy instruments that are most relevant to this paper.

Public Works and Government Services Canada (PWGSC) delivers the SC infrastructure supporting the secure delivery of services, and is governed by the

Department of Public Works and Government Services Act [DPW86]. This act stipulates that PWGSC is a common service agency for the Government of Canada (GoC), providing the GoC's departments, boards and agencies with services in support of their programs. The *Privacy Act* [PA85] extends Canada's laws that protect the privacy of individuals with respect to the collection, use, and disclosure of personal information about themselves held by a federal government institution. The *Personal Information Protection and Electronic Documents Act* (PIPEDA) [PIPE00] establishes similar rules and while the *Privacy Act*, applies only to GoC departments and agencies, PIPEDA also applies to provincial governments and the private sector.

Notable policies (applying only to the federal government) that affect the delivery of e-government services include: the *Government Security Policy* [GSP02], which safeguards employees and assets and assures the continued delivery of services; and the *Privacy Impact Assessment Policy* [PIA02], which prescribes Privacy Impact Assessments when there are proposals for, and during the design and implementation of programs and services that raise privacy issues. Additionally, the *Common Services Policy* [CSP] is designed to ensure that departments and agencies can acquire responsive, cost-effective support for their program delivery, while the *Common Look and Feel Standard* [CLF00] is designed to ensure that all Canadians, regardless of ability, official language, geographic location or demographic category, are given equal access to information on GoC web sites.

2.2 Canada's Secure Channel

Canada's Secure Channel (SC) is a collection of network infrastructure, operations, and security and authentication services supporting the Government On-Line (GOL) project. While the Secure Channel Network (SCNet) supports a secure enterprise communication environment for the federal government, the SC's authentication services support the identification of citizens and businesses (hereafter referred to generically as individuals). We expand upon these authentication services below (see also Just [Just03] for further information).

In order to authenticate themselves prior to accessing certain online government services, individuals can obtain an *epass*, an online credential issued by the Government of Canada (GoC). An *epass* consists of a private/public key pair and associated public key certificate. The certificate is indexed by a Meaningless But Unique Number (MBUN) that has no association with the individual's identity. On its own, this certificate is anonymous; until which time an individual enrols with a government program and the MBUN is associated with existing government program identifiers (such as an individual's Social Insurance Number – SIN, which is roughly equivalent to the US Social Security Number) for the individual (at which point the certificate is pseudonymous).

The process of establishing this secure relationship between the individual's MBUN and a government program for online service delivery is composed of the following steps:

- The individual *registers* in order to obtain their epass. No identification is required by the individual at this stage.
- The individual *enrols* with a government program, involving (where required)
    - The identification of the individual to the program (based on shared, secret information between the individual and the program),
    - The mapping of the MBUN from the epass certificate to the existing Program Identifier (PID) that identifies the individual within the government program. This mapping is thereafter retained at the program.

Upon subsequent authentication to the government program, the MBUN from the individual's certificate is mapped to the PID, thereby identifying the individual, and the resources they can access. In addition to the epass registration, management operations such as self-revocation and recovery are also supported.

Since 2002, over 285,000 *epasses* have been issued to individuals. Key *epass*-enabled applications include Change of Address with the Canada Revenue Agency and filing Records of Employment with Human Resources and Skills Development Canada.

The SC's authentication services advantage is that beyond offering a solution that supports the protection of individuals' privacy, it provides a common solution for use by all departments, avoiding the arguably less efficient "siloed solutions" that might be built by each department.

In further support of privacy, individuals are able to obtain one or more *epasses* (e.g. mapping to a number of different government programs). It is likely though that individuals will opt for a manageable number of *epasses*, supporting a more consistent experience in their authentication to government. Note also that no personal information about an individual is maintained centrally to support the *epass* service; personal information about a user is retained within the context of their relationship within a program.

3 Meeting the Challenges Related to Secure Service Delivery

As Canada continues to offer secure online services to citizens and businesses, a number of issues have been overcome, while some remain. This section reviews some of our successes, and considers how we might address a number of lingering challenges. It is anticipated that this overview will aid organizations that have encountered, or expect to encounter, similar situations.

3.1 Guiding Principles/Goals

In delivering government services, there are three entities to consider, and hence three different perspectives from which to judge the quality of service delivery: the individual user, the department and the enterprise (as representative of the whole-of-government). The user perspective is to have their access to services unencumbered by technology or politics. Therefore, the "user experience" is important. In addition, users are concerned with ensuring both the security of their information (and their person!) and well as their privacy. Departments are similarly concerned with the needs of their users, but also want to offer the most effective and efficient solution possible. Cost is always an important concern so that re-usable solutions offer an important advantage to departments. It is the enterprise perspective in which the needs for the whole-of-government are considered. Effectiveness and efficiency across all of government is recognized, without the sometimes-narrower constraints of departmental budgets or other requirements. Within the Canadian government, the enterprise perspective is upheld by designated central agencies (that develop government policy) and common service organizations, such as PWGSC, that implement common solutions (including the Secure Channel) within the government's policy framework.

Throughout the evolution of Secure Channel, there are a number of challenges have arisen from attempting to satisfy these perspectives within our legal, policy and business constraints. In the following subsections we overview a number of these challenges.

3.2 Inter-Jurisdictional Issues

As mentioned earlier, a number of jurisdictions exist within Canada, each with their own legislative and policy framework, and with their own business requirements.

Despite these differences, each jurisdiction or level of government (federal, provincial, and municipal), interacts with the same set of individuals: Canadian citizens and businesses. For example, the federal government is responsible for tax collection, the provincial for access to health services and for issuing drivers licenses, while the municipal is responsible for water and sewer services. In addition to public services, individuals also interact with, and obtain services from the private sector (e.g. banks).

The Secure Channel solutions currently support the delivery of federal government services. To support an improved user experience, it would be advantageous for services to be similarly offered across other jurisdictions. Of course, such integration of service offerings must be performed within the constraints of legislation and policy, especially with regard to privacy.

As an example, consider the use of a single authentication infrastructure for citizens to access government services. As with the current *epass* solution for federal services, users can choose to use one *epass* across all services, or use separate *epasses* for any number of services. Although there might exist some obstacles to such an endeavour (e.g. technical, perception), one clear obstacle is found in current legislation. The common service organization that provides the Secure Channel services is governed legislatively by the *Department of Public Works and Government Services Act* (see Section 2.1) In both this legislation and other policy (for instance, the *Common Services Policy* (see Section 2.1)), PWGSC is recognized as a "common service organization" for the Government of Canada, but the legislation limits this authority to offering services only to "departments, boards and agencies." Thus, the Act currently prevents the offering of services to our provincial counterparts, for example.

There are a number of options for dealing with this issue, including offering solutions through another common service organization, or even choosing to reject the idea of an improved government experience across multiple levels of government. However, one can also recognize that we are now in an environment that may not have been anticipated by legislation and for which it is reasonable to investigate legislative alternatives. In Canada, this option can be achieved through an Order-in-Council (OIC), involving approval by the Governor General, Canada's head of state, and is currently being investigated.

Beyond current legislative hurdles, potential policy obstacles also exist. For example, in the federal government (in support of consistent user experience), there exist standards supporting a "Common Look and Feel" for government online systems. These standards go so far as to specify the required presentation format for web pages. However, in a joint federal-provincial presentation, provinces have similar, sometimes conflicting, policies suited to their own requirements. As the federal policies apply only to federal departments, there is a gap with regards to policies across all levels of government; the common look and feel standards are likely but one example.

Therefore, as we endeavour to offer systems across multiple levels of government, we'll be sure to uncover other areas where a pan-jurisdictional policy regime is lacking.

3.3 Business Registration Issues

A business transacting with the Government of Canada (GoC) typically has a number of government-issued identifiers for its dealings with the various departments and programs of the GoC. While the Business Number (BN)[1], issued by the Canada Revenue Agency (CRA) to Canadian businesses for tax purposes seems to have the widest coverage, the use of the BN beyond tax purposes is currently limited by legislation. As a result, although many businesses have a government issued BN,

not all government departments can collect and use the business number for identification purposes in their programming. Thus, businesses obtain other identifiers for interacting with the GoC for non tax-purposes. This is important to note when considering that a common identifier could be used in a common registration process for access to online government services. Having a single registration number for a business' transactions with the GoC would simplify and enhance a business' interactions with the GoC as a whole and further enable business-to-government online transactions.

An ideal scenario for business registration is that there would be an efficient way of identifying businesses once for their dealings with the GoC. Subsequent authentication would be simple for both the businesses and the GoC and would be used for all business-to-government online transactions. One may also wonder if an anonymous credential such as an *epass* could be used for business registration purposes. And although *epass* meets privacy requirements that are critical for individuals, the purpose of an identifier for a business is precisely to identify from an authoritative source that a business is legitimate. In addition, with businesses, individual enrolments may not be necessary as they are currently with citizens in order to maintain the information silos.

Presently, CRA uses the BN to identify its business clients and departments including Industry Canada, PWGSC and Statistics Canada, and the provinces of British Columbia, Manitoba, New Brunswick, Nova Scotia and Ontario are using the BN for business identification in varying ways. These departments and provinces were able to adopt the BN as a common identifier because they *changed their legislation* to allow the use of the BN for this purpose and signed memoranda of understanding with CRA to use it. This business registration solution only partially fulfils the need for a single registration number, because even though the BN is used as a common identifier, the solutions are for specific uses, rather than having a definitive identifier that be consistently used for online transactions. Making the BN the sole common identifier for business and making its usage more widespread would entail legislative change either on the part of CRA or other departments and provincial jurisdictions.

Alternatively, an altogether distinct number, publicly available and created specifically to be a common identifier for businesses in their dealings with the GoC, may resolve this issue. It is likely that legislation would have to be enacted to allow the creation, collection, use and disclosure of this new ubiquitous identifier. That said, it may be simpler to create a new number than persist in using an identifier that requires legislative change to enable its expanded use within the GoC and in the provinces. Other benefits include that the public could view the business number and other businesses could rely on it to verify the legitimacy of other entities.

This very scenario occurred in Australia in 1999 when the *A New Tax System (Australian Business Number) Act 1999* came into force. The act created the Australian Business Number (ABN)[2], which is related to, but distinct from a businesses' tax file number and is used when dealing with the Australian Tax Office, other government agencies, and when supplying goods and services to other businesses.

The absence of a persistent identifier for businesses dealing with the Government of Canada hinders the seamless development of online business-to-government transactions because departments must individually collect similar identifying information from the same business when this information could be collected once and then shared and re-used by other GoC departments. Having a common identifier, supported by a central, searchable business registry would streamline the registration process for online services, allow government departments to leverage the work of others while presenting

a single-window to business interacting with the GoC. While the issue of determining a common identifier for online business authentication with the GoC has not been resolved, "catalytic projects" that are part of Canada's GOL initiative are addressing elements of the problem. For instance, the purpose of the Common Business Authentication catalytic project is to implement a common authentication service for businesses and individuals in dealing with government [SCF04]. Providing tombstone information once, the business would then be authenticated once for multiple services and while having a view of all government transactions.

While the Canada Revenue Agency BN need not become the common identifier for businesses, it would be advantageous to make a publicly available business identifier mandated by legislation for use by the entire Government of Canada. As in Australia with its ABN, this business identifier would then be the foundation for authenticating the business entity transacting with the GoC. Furthermore, it could also become an identifier used among businesses, and the public in verifying the legitimacy of a business entity.

3.4 Citizen Enrolment Issues

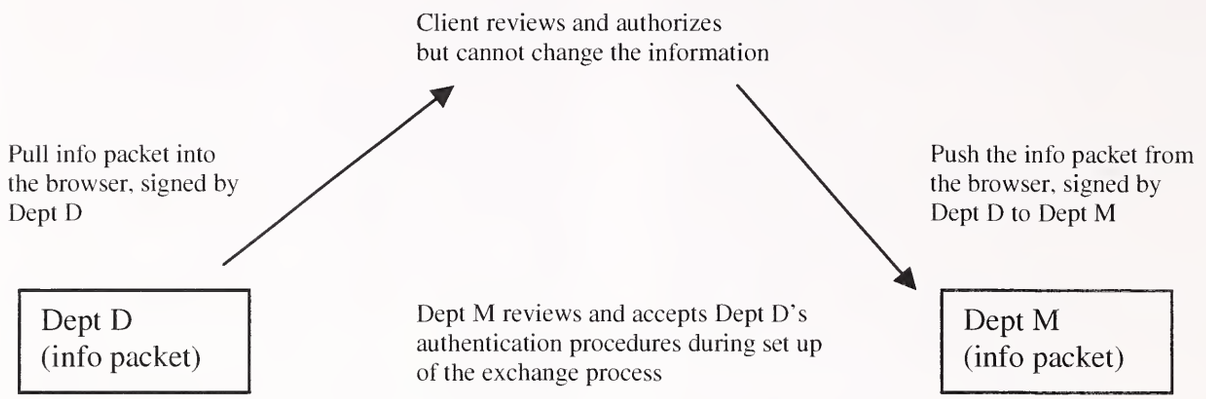
When a citizen chooses to use the GoC *epass* service to transact online with the GoC, the individual registers for one or any number of *epasses* through a central service.[3] But in order to access specific PKI-enabled GoC services, individuals must enrol and identify themselves separately with each government program offering a service. This is because privacy policy and legislation [PA85] restrict either the central maintenance of such information, and the sharing of information between government programs. The GoC entities offering the individual services therefore control their "program space" and determine what information is required in order for the citizen to authenticate him/herself to that program.

The scenario described above allows the programs offering the service to identify individuals to the level of assurance it wishes to have and it also keeps the information provided for enrolment in a program-specific silo. Registration for, and maintenance of, the credential can be centrally managed (as no personal information is collected) offering significant benefit. However, to the individual who may assume that GoC departments already share this information, separate program enrolments may lessen the quality of the individual's experience. In other words, a pleasant, non-repetitive experience presupposes either a shared, central repository of information, or an information exchange facility in which the individual could decide to share enrolment information across government programs. As it stands now, an individual has to authenticate her/himself for each *epass* enabled services despite being able to register for only one *epass*. Due in part to legislative and political hurdles, there is no central authentication facility for users and departments cannot use another service's authentication for its own. In the following paragraphs, we examine the feasibility of two other options.

Joint Information Exchange Facility

The Joint Information Exchange Facility is a concept that demonstrates how a user can choose to share authentication information provided for one service with another. This client controlled sharing of information means that the user does not have to re-enter identity-proofing information so long as it can be re-used from a previous enrolment, resulting in a faster enrolment process and an improved user experience. The following scenario illustrates how this concept would be put into action. The client visits a department (Department M) and wishes to re-use her authentication that was previously completed at another department (Department D). The client pulls the information from the originating department (Department D). Department D prepares the

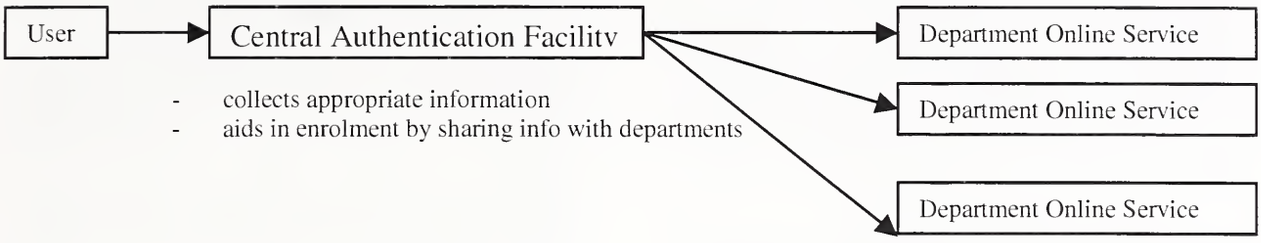
information packet for transmission, digitally signing the information packet and sending it to the client's browser. The client can review the information but cannot change the information and then the client elects to pass the information packet onto the receiving department (Department M). Thus, by establishing individual consent, while retaining other privacy features, the user experience can be greatly improved.



Central Authentication Facility

Although the Joint Information Exchange Facility improves the user experience, individual participation is still required to convey (though not physically enter) their enrolment information. The purpose of a Central Authentication Facility (CAF) is to serve as the authentication authority for user access to online government services. Authenticating individuals on behalf of departments, it could allow users to register once for access to the suite of online government services. The following scenario illustrates how this concept could be put into action. An individual wishes to use two departments' online services. The user is directed to the CAF. The CAF collects the individual's tombstone (e.g. name, date of birth) and contact information and requests any additional information as appropriate. This then becomes the individual's profile. The tombstone and contact information collected by the CAF is made available to the departments' programs, while for additional assurance, the programs may choose to ask for additional e.g. "shared secret" information specific to the business of the program, such as previously submitted tax information. Over several program-specific authentications, an assurance level might be assigned to the individual's profile, and raised subsequent to successful presentation of these shared secrets. As a result, the individual registers once and authenticates through the CAF, which shares this information with departments when the individual requests access to specific services.

While the high-level description given here suggests that technical solutions are readily available and straightforward, current legislation would keep it from being operationalised. For instance, the *Privacy Act* prevents the collection use and disclosure of personal information for purposes other than those for which it was collected. As a result, the simplest way of making a CAF happen is that the department offering the service would have to legislatively make it one of its "operating programs." An operating program can be defined as a series of functions designed to carry out all or part of an entity's operations. The CAF would then be part of a department's line of business, collecting personal information and authenticating individuals on behalf other departments offering online services.



3.5 Evidentiary Support for Electronic Data

The admissibility or acceptability of electronic data is critical to the success of our online systems, and therefore critical to our ability to obtain the purported savings and efficiencies. Within the world of PKI, such issues have often been mired in narrow discussions of "non-repudiation." While ensuring the correctness of technical solutions is important, we will not obtain our goals of supporting electronic data unless the technical solutions fit within the legal, policy and business framework for the overall online solution in question. And while a digital signature is an important, if not necessary requirement for many applications, a digital signature alone does not necessarily provide sufficient evidence. If repudiated in a court of law, there is much information that can contribute to demonstrating that some action was or was not performed. The ability to support such evidentiary needs is as much a question of information management and the processes that support this management, and needs to be driven (at least in part) by the relevant legislation (should it exist). Fortunately, in Canada we have some legislation to help guide the determination of required evidentiary material.

Changes to the *Canada Evidence Act* [CAE04] in 2000 included clauses describing the evidence required for "the proof of the integrity of [...] electronic documents" for any "person seeking to admit an electronic document as evidence." The rules of evidence in this matter weigh heavily upon being able to demonstrate that "the electronic documents system was operating properly." Evidence of proper operation will rely heavily on demonstration of the environment at the time in question, and more generally, that this environment was operated upon using standard protocols and procedures. Therefore, in support of evidentiary requirements, sufficient standards need to be defined (and of course followed).

While the *Canada Evidence Act* refers to the integrity of "electronic documents", part 2 of the 2000 *Personal Information Protection and Electronic Documents Act* (PIPEDA) [PIPE00], defines more specific tools supporting this integrity, including defining a "secure electronic signature." As specified later in the draft 2004 *Secure Electronic Signature Regulations* (SESR) [SESR04] (which serve as a companion to PIPEDA), a "secure electronic signature [...] is a digital signature" as constructed using public key cryptography. The SESR also identify a recognition process for a "recognized Certification Authority" in support of issuing digital certificates for digital signature production. Though not fully specified at this time, the recognition process will rely heavily upon the operating procedures for CAs as defined in the Government of Canada's Certificate Policies.

In essence, legislation has provided a first step in bridging the legal and technical realms from statutes and regulations to public key technology. The SESR recognize the need for a recognition process, and additional areas may need to be similarly addressed. As recognized in the *Canada Evidence Act* (above), further standards will help to better support the rules of evidence. For instance, the draft Canadian standard CGSB (Canadian General Standards Board)

72.34 on "Electronic Records as Documentary Evidence," describes conditions and practices for good data and records management which will ensure their evidentiary value and admissibility over time.

Within the model and solutions for Secure Channel, there are three entities affected by this legal framework: the Secure Channel operations itself, participating departments, and the user (citizens and businesses). Each has a role and responsibility in ensuring the confidentiality and integrity of a Government On-Line transaction and ensuring that sufficient evidence is collected with regard to such transactions.

With regard to the operations of Secure Channel, as with any other large infrastructure, there are practical concerns regarding what information (records and logs) needs to be retained, and for how long, in support of evidentiary requirements. The retention duration will often relate to departmental requirements for evidence. Work is proceeding in this area, though current evidential areas where standard processes are described and evidence of assessments exist include the PKI Certificate Policies (CPs) and Certificate Practice Statements (CPS), the results of Secure Channel Certification and Accreditation, design documentation, routine assessments (e.g. ensuring proper CA operation), and numerous audit logs relating to key events and for day-to-day operations.

We are well on our way past the question of non-repudiation, and considering solutions to the broader questions of evidentiary support, with the legal and business framework provided with our Government On-Line solution. It is likely that many of the evidentiary requirements are refined through case law, though as in most jurisdictions around the world, we have not yet had sufficient experience with repudiated digital evidence.

4 Concluding Remarks

Despite the strides we've made in the development of our e-government solutions, we often find ourselves having to justify our selection of PKI. The questions are commonly asked by individual departments, and are asked in comparison to the option of a PIN-based SSL solution. Such questions often take one of the following two forms:

1. Why should we use a common PKI solution instead of developing and using our own solution, either with PKI or with PIN-SSL?
2. Why is a PKI solution better than a PIN-SSL solution?

The argument for a common solution, versus individual department solutions, is often tied to the principle of whether cost savings can be achieved by building and re-using a single solution across multiple organizations. When done properly, this option has the potential for great savings, and also delegates many issues related to managing a solution (e.g. software updates) to the common service provider. However, the benefit of improved user experience should not be overlooked. To many citizens, departmental distinctions are often irrelevant. At least from the point of view of their experience, they often just want to obtain service from "the government," as opposed to a particular department.

With regards to comparisons to PIN-SSL, the bottom line is that comparisons are often apples-to-oranges, with our established PKI-based solution, adapted to the policy and legal regimes of government, compared to *some* PIN-SSL solution. While a PIN-SSL solution could be well implemented, with additional functionality and security features, there are also a number of poor, ill-defined implementations. However, our fundamental needs have always been consistent, including requirements for persistent integrity with digital signatures, persistent confidentiality from end-to-end, support for single sign-on with a single

epass, and robust credential management. And our novel variant [Just03] has allowed us to overcome, or at least better deal with common problems associated with PKI [Gut, Gut02].

With our PKI-based solution, we have thus far been able to design and deploy a solution that fits within the legislative and policy framework of the Canadian government, and that has attracted over 285,000 individuals since 2002. As we move forward, it is most often these issues, and not those of technology, that present us with hurdles. The issues we identified in the previous section: inter-jurisdictional considerations, business registration, enrolment, and evidentiary support for non-repudiation, attest to the complexity of assuring secure e-government service delivery in Canada. These and other challenges are not insurmountable, and indeed the possible solutions we offered may contribute to making an already world-renowned solution even more innovative.

5 References

[CAE04] Department of Justice – Canada, *Canada Evidence Act,* 1985. Available at http://laws.justice.gc.ca/en/C-5/15974.html

[CLF00] Treasury Board of Canada, Secretariat, *Common Look and Feel – Background* 2000. Available at http://www.cio-dpi.gc.ca/clf-nsi/back-cont_e.asp

[CSP] Treasury Board of Canada, Secretariat, *Common Services Policy*. Available at http://www.tbs-sct.gc.ca/Pubs_pol/dcgpubs/TB_93/CSP_e.asp

[DPW86] Department of Justice – Canada, *Department of Public Works and Government Services Act*, 1986. Available at http://laws.justice.gc.ca/en/p-38.2/text.html

[GSP02] Treasury Board of Canada, Secretariat, *Government Security Policy*, 2002. Available at http://www.tbs-sct.gc.ca/pubs_pol/gospubs/TBM_12A/gsp-psg_e.asp

[Gut] P. Gutmann, "Everything you Never Wanted to Know about PKI but were Forced to Find Out"; see http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf

[Gut02] P. Gutmann, "PKI: It's Not Dead, Just Resting", *IEEE Computer*, August 2002; see http://www.cs.auckland.ac.nz/~pgut001/pubs/notdead.pdf

[Just03] M. Just, "An Overview of Public Key Certificate Support for Canada's Government On-Line (GOL) Initiative", in *Proceedings of 2nd Annual PKI Research Workshop*, April 2003.

[PA85] Department of Justice – Canada, *Privacy Act*, 1985. Available at http://laws.justice.gc.ca/en/p-21/text.html

[PIA02] Treasury Board of Canada, Secretariat, *Privacy Impact Assessment Policy*, 2002. Available at http://www.tbs-sct.gc.ca/pubs_pol/ciopubs/pia-pefr/paip-pefr_e.asp

[PIPE00] Department of Justice – Canada, *Personal Information Protection and Electronic Documents Act*, 2000. Available at http://laws.justice.gc.ca/en/p-8.6/text.html

[SCF04] B. Watkins, "Secure Channel Future", presentation to *GTEC Special Forum – Security*, October 20, 2004.

[SESR04] *Secure Electronic Signature Regulations,* draft, Canada Gazette Part I, 2004. Available at http://canadagazette.gc.ca/partI/2004/20040508/html/regle6-e.html

Notes

[1] The Business Number (BN) is a 15 character client identification number that consists of a nine digits to identify the business and two letters and four digits to identify each account a business may have. http://www.cra-arc.gc.ca/E/pub/tg/rc2/rc2eq.html#P72_2512

2 The Australian Business Number (ABN) is a unique 11 digit identifier issued by the Australian Tax Office (ATO) to all entities registered in the Australian Business Register. It is used when dealing with the ATO, other government agencies, and when supplying goods and services to other businesses. http://help.abr.gov.au/content.asp?doc=/content/16974.htm&usertype=BC

3 Here we distinguish between registration – the process of obtaining an *epass* from *epass* Canada – and enrolment – the action of signing up or applying for a Government of Canada service.

The Use of PKCS-12 in the Federal Government

Tice F. DeYoung, PhD
National Aeronautics and Space Administration

INTRODUCTION

The purpose of this paper is to provide a brief description of the history of the Public-Key Cryptography Standard number 12, or PKCS-12, the Personal Information Exchange Syntax, for exporting public key infrastructure (PKI) private keys and certificates; to investigate the implications of using this mechanism as they apply to the Federal PKI Policy Authority; and to present a set of conclusions which are not recommendations *per se*, but are rather a list of things to consider before one permits end users to export their PKI private keys and certificates using PKCS-12.

BACKGROUND

Before we describe PKCS 12, we should first mention what the Public-Key Cryptography Standards are. These specifications are produced by RSA Laboratories in cooperation with a number of developers worldwide for the purpose of accelerating the deployment of public-key cryptography. The first PKCS was published in 1991. Since then the PKCS documents have become widely referenced and implemented.

THE PKCS-12 STANDARD

To understand how PKCS-12 came about, we have to go back to 1995. At that time all of the encryption software was proprietary and there was no mechanism for people to securely communicate unless they had the same product. This lack of interoperability was recognized by a number of people.

There were several development options that could lead to interoperability. First, you could set up a new application specific certificate authority (CA) to issue PKI certificates for every user of that application. Second, you could develop application specific plug-ins for every other application. Because each of these options leads to unnecessary complexity and/or expense, the community arrived at the best option; they all agreed that a single standard should be developed. That standard came to be known as PKCS 12, published by RSA Laboratories in 1999. [1]

The PKCS 12 standard built upon and extended the 1993 PKCS 8: Private-Key Information Syntax Standard [2] by including additional identity information along with private keys and by instituting higher security through public-key privacy and integrity modes. The PKCS 8 standard described syntax for private-key information, including a private key for some public-key algorithm and a set of attributes, as well as, syntax for encrypted private keys.

The PKCS-12 standard describes a transfer syntax for personal identity information, including private keys, certificates, miscellaneous secrets, and extensions. Applications that support this standard will allow a user to import, export, and exercise a single set of personal identity information. This standard supports direct transfer of personal information under several privacy and integrity modes, the most secure of which require the source and destination platforms to have trusted public/private key pairs usable for digital signatures and encryption, respectively. PKCS 12 permits both software and hardware implementations. Hardware implementations offer physical security in tamper-resistant tokens such as smart cards. [1]

FEDERAL PKI POLICY AUTHORITY

The Federal PKI Policy Authority (FPKI-PA), under the auspices of the Federal Identity and Credentialing Committee (FICC), is responsible for the policies of the various Federal PKI implementations; the Federal PKI Bridge CA (FBCA), the Common Policy Framework CA (CPFCA), the eGovernance CA (eGOVCA) and the Citizen and Commercial Class CA (C4A). This paper will only discuss the relevancy of the FBCA and its concomitant Certificate Policy to the use of PKCS 12.

The Federal Government is required to use cryptographic modules that have been accredited as meeting the NIST Federal Information Processing Standard 140 level 2 (FIPS 140-2) accrediting process [3]. Additionally any entities PKI that want to cross-certify with the FBCA must follow US Government PKI Cross-Certification Methodology and Criteria. [4]. Once an entity PKI has completed this process, they are required to sign a Memorandum of Agreement (MOA) with the FPKI-PA, which lays out the rights and responsibilities of both parties. One of the items in every MOA is the requirement that the entity PKI cross-certifying with the FBCA shall maintain compliance with the requirements in the MOA and shall notify the FPKI-PA if any material changes occur.

ISSUES WITH THE USE OF PKCS-12

The FBCA Certificate Policy [5] is silent on the use of PKCS 12, so that its use does not apparently violate any of the requirements in the MOA. However, as we will discuss later, there are issues associated with private key protection and activation, which are application specific that must be addressed. One of the questions that arises is whether or not an entity must notify the FPKI-PA if it intends to use PKCS 12; another is whether the entity must notify the FPKI-PA of every application that it intends to import the PKI secret keys and certificates into. There are others, which will be discussed in more detail in the following sections.

Exporting the private keys and certificates isn't the problem. PKI applications that follow the PKCS-12 standard keep the exported data in an encrypted state. To further explain this, let me use an analogy. Let's assume that the PKI application is equivalent to a safe, because the private keys and certificates are maintained securely, in one place and easy to centrally manage. When the information is exported, it is no longer in the safe, but is in a portable lock box, or secure briefcase, which is portable. The container is protected, so that doesn't cause a problem. However, the portable container can be used in an insecure fashion if it isn't carefully controlled.

Let me use as an example the FBCA CP Medium Level of Assurance requirements. The FBCA CP requires private keys to be protected with FIPS 140-2 accredited devices and applications for cross-certification at the Medium Level of Assurance. I use this example because the overwhelming majority of entities cross-certified with the FBCA and those who have applied for cross-certification have been at this level. Furthermore, the Medium Level of Assurance at the FBCA covers both Levels 3 (software PKI) and 4 (hardware PKI) as outlined in the OMB Guidance on Authentication Levels[6] and the associated NIST Electronic Authentication Guideline[7]. For this example I am assuming that the PKCS 12 exported PKI data is secured in accordance with FIPS 140-2. Therefore, exporting the PKI data from the PKI application (safe) to the PKCS-12 container (secure briefcase) has not violated the FBCA CP requirements. One of the FBCA CP requirements is that passwords used to unlock access to the private PKI keys must be at least 8 characters in length and contain at least one from each of the following categories (upper case, lower case, numbers and special characters). This requirement is easily met when the private key data is in the PKI application and when the data is exported using PKCS-12.

Now things begin to get interesting. While within the PKI application, the keys and certificates are centrally controlled and managed. However, when the PKI data is exported, the PKI applications are now unable to provide this management because they have no control of the portable PKCS-12 container, nor do they control what applications and devices the private keys and certificates are imported into. The end user is the only one who can control things once they have been exported. Therein lies the rub! The end users will have to maintain consistency between their keys and certificates that have been exported and those that are still within the PKI application. They will have to keep track of updates, key rollover, what applications and devices they have exported them to, etc. They are also the only ones responsible for determining if the applications and devices meet the FIPS 140-2 requirements. How can we ensure that the end users maintain their exported PKI data in a manner consistent with their CP and CPS and their FBCA MOA?

This brings us to the next section of this paper, namely the things one should consider before allowing end users to use the PKCS-12.

THINGS TO CONSIDER

First and foremost, you do not want to do anything that will cause your use of PKCS-12 to violate the level of assurance of your CA and, if you are cross-certified with the FBCA, you don't want to jeopardize the MOA requirements you have with the FPKI-PA; so tread carefully if you do decide to permit PKCS-12 export of private keys and certificates.

What are the benefits of permitting the use of PKCS-12 mechanism for exporting private keys and certificates? One that comes quickly to mind is that it permits you to import them into the Blackberry Personal Digital Assistant (PDA) beloved by upper level management in most Federal agencies. These devices are ubiquitous throughout the ranks of these Senior Executives who tend to be the least versed in information technology security issues. We have to provide them and their data without bothering them with details or interfering with their ability to properly lead their agencies.

Blackberries are only the tip of the iceberg when it comes to PDAs. There will soon be smart phones,

other intelligent PDAs and possible Personal Access Networks (PAN). The one thing they have in common is wireless connectivity. Wireless access points will rapidly follow and we will have to provide adequate security for these devices using some form of cryptographic mechanism. Having the same private keys and certificates for the myriad devices and applications is an efficient way to easily manage this. PKCS-12 gives you that capability.

Now, what can you do to ensure that giving your users this capability doesn't compromise your security and thus your level of assurance? First, review your CP and CPS to see if there are any changes required. Note that if you are operating at the High Level of Assurance, you cannot permit the use of PKCS-12 export or you will violate your policies. Second, notify the FPKI-PA of your intention and the steps you will take to ensure that you do not violate your level of assurance. Next, consider additional procedures and processes for the subscribers that will use PKCS-12 export. Here are a few suggestions, but his list is far from complete:

A. Put thePKCS-12 users in a separate group;
B. Use a separate OID in their X.509 certificates;
C. More closely audit their usage;
D. Require additional training in managing their private keys and in the procedures for exporting and importing them;
E. Require subscribers to obtain permission from your agency Policy Authority (PA) before importing their private keys and certificates into an application or device;
F. Develop a list of FIPS 140-2 approved applications and devices that would be the basis for your PA decisions in E.;
G. Issue their credentials at the Basic Level of Assurance, but only if absolutely necessary to maintain cross-certification
H. Develop a Supplemental Subscriber Agreement that describes their additional responsibilities and notifies them that extra training is required.

Here are some things you might want to include in the additional training for your users who want to use PKCS-12. Again, this list is for illustrative purposes only and should not be considered to be all-inclusive:

A. Describe their responsibility for and methods of managing their exported keys and certificates;
B. Explain that the users can only import their private keys and certificates into applications and devices that their PA has approved as meeting FIPS 140-2 requirements;
C. Describe the processes and procedures to followed for exporting and importing their PKI data;

CONCLUSIONS

As we stated at the beginning, we are providing no conclusions or recommendations on the use of PKCS-12; but instead, have briefly discussed some things to consider before embarking into the brave new world of permitting users to export their PKI private keys and certificates using the PKCS-12 export mechanism.

However, for your information, we at NASA have carefully weighed the attractive benefits and the potential dangers of permitting users to export their private keys and certificates and have made the decision to permit certain users to use PKCS-12. We are now implementing some of the above steps to ensure that we do not compromise our security. We are cautiously optimistic that things will proceed smoothly.

REFERENCES

[1] PKCS 12 v1.0: Personal Information Exchange Syntax, RSA Laboratories, June 24, 1999.

[2] PKCS 8: Private-Key Information Syntax Standard, RSA Laboratories, November 1, 1993.

[3] Security Requirements for Cryptographic Modules, NIST FIPS 140-2, December 3, 2002.

[4] US Government PKI Cross-Certification Methodology and Criteria, March 2003.

[5] X.509 Certificate Policy for the Federal Bridge Certification Authority, 27 September 2004.

[6] E-Authentication Guidance for Federal Agencies, OMB 04-04, December 16, 2003.

[7] Electronic Authentication Guideline, NIST Special Publication 800-63, v. 1.0, June 2004.

Secure Access Control with Government Contactless Cards

David Engberg
CoreStreet, Ltd.
One Alewife Center, Suite 200
Cambridge, MA
dengberg@corestreet.com

Abstract—**Government agencies have begun widespread usage of public key technology for information security applications such as secure email, document signing, and secure login. These deployments use PKI tokens in the form of contact smart cards with private keys protected through a match-on-card second factor such as a PIN. More recently, the government has begun to standardize card technology for contactless physical access. These contactless cards will be capable of symmetric key usage, but will not be able to perform private key operations. They will also typically be limited to 1-4kB of read/write data storage.**

This paper discusses ways to use digitally signed messages to perform strong authentication and authorization using the current generation of contactless smart cards. This will compare different strategies under consideration and discuss the security and usability considerations of each. Particular emphasis is placed on techniques to support the use of these cards in inter-agency and offline settings.

Keywords—contactless, smart cards, biometrics, access control

1 Overview

There are two questions that must be answered by an access control system before permitting access. The first question is: "*Who are you?*" The answer to this question is your **identity**, which is permanent throughout your life. The process of answering this question, **authentication**, must rely on one or more factors to uniquely determine your identity. These factors are typically divided into three categories:

Something you have: E.g. a badge, a metal key or a smart card

Something you know: E.g. a PIN or a password

Something you are: E.g. your fingerprint, your iris or your voice

An access control system authenticates these factors to identify each user. Since your identity never changes, the process of authentication should always yield the same result, even if the factors used may change.

Once your identity is determined, the system must answer a second question: *"Are you currently allowed to access this resource?"* This question is answered through a process called **authorization** or **validation**. Unlike authentication, which should always yield the same identity for each person, the result of authorization may change frequently. This change may be the result of a change in user privileges (e.g. a promotion), a change in policies, or a change in the environment (e.g. time of day, etc.).

This document describes techniques and technologies that can be used to perform secure access control using the current generation of government contactless cards. This focuses on solutions that will support cards based on ISO 14443 Parts 1-4 [ISO01], such as those using Philips' DESFire chips. These cards comply with NIST's Government Smart Card Interoperability Specification (GSC-IS) version 2.1, Appendix G [SDW+03]. The general techniques described in this document should also be applicable to other contactless memory cards, including those with other symmetric key schemes (e.g. HID's iClass).

The techniques described in this document are primarily compared in their ability to permit strong authentication and authorization within federated environments where a single central access control system is not possible. This support for federated access control also leads to the ability to perform authentication and authorization in disconnected settings where no communication is available to central management servers.

2 Secure Contactless Authentication

The process of authentication uses one or more factors to securely determine the identity of a cardholder. These factors may be fully independent (printed photo, contactless card serial number), or may be interconnected (e.g. contact chip PIN and PKI applet). An effective authentication factor will uniquely and unambiguously identify a specific individual, binding to their universal identifiers. Different authentication factors also vary in their level of protection against modification or duplication. Finally, some factors that may be appropriate in a closed environment with guaranteed network

connectivity may not be usable in federated or disconnected settings.

The following sections describe various factors that may be used with contactless DESFire cards to perform authentication.

2.1 DESFire card unique identifier (CUID)

Every DESFire card is manufactured with a unique and read-only card unique identifier (CUID), which is made up of a one byte manufacturer code (e.g. Philips: 0x04) and a six byte card number that is set by the manufacturer. Barring a manufacturing error, no two legitimate DESFire cards should have the same CUID.

The card CUID can be retrieved by any ISO 14443 reader within range of the card. This CUID is read in clear text form during the card selection and anti-collision processing, which precedes any other card actions.

Pro: The serial number is unique and unambiguous. The UID of a legitimate card cannot be modified.

Con: The serial number has no cryptographic or protocol-level protections to prevent an attacker from asserting the same serial number as any real card. By implementing ISO 14443 directly, an attacker can imitate any desired CUID.

The CUID only represents a basic assurance factor for authentication.

2.2 Stored identification string

In addition to the manufacturer's CUID, it is possible to write a more extended identification string into the card memory that represents the cardholder. Example encodings would include a SEIWG-012 string [SEIWG02] or a PAIIWG Card Holder Unique Identifier (CHUID) [PAIIWG04]. Under current proposals, this string would be written to the card in a known location where it would be generally available in a "read-only" mode.

These identification strings can contain a larger amount of unique identification such as organizational affiliation and unique personnel identification number within that organization.

A digital signature on the CHUID by a trusted authority can be used to prevent the forgery of modified CHUIDs.

Pro: For legitimate cards issued by the government, a stored identification string such as a SEIWG-012 offers a unique identifier that also includes affiliation information for cross-organizational interoperability. This string cannot be modified on a valid card without access to the issuer's master key.

Con: The stored identifiers are not strongly bound to either the cardholder or the physical card, so they may be easily duplicated or imitated onto another card. By implementing ISO 14443 directly, an attacker can imitate any desired CHUID. Digitally signed CHUIDs prevent the assembly of arbitrary false identifiers, but this does not provide any protection against the complete duplication of a valid CHUID onto another real or emulated card.

A stored identification string only represents a basic assurance factor for authentication.

2.3 Symmetric key authentication

High-end ISO 14443 cards such as the DESFire offer strong mutual authentication and over-the-air encryption using symmetric (secret) keys. For example, a DESFire application can be configured to only permit access by reader that knows a secret Triple-DES key that is stored on the card itself. Only readers that know this shared secret key are capable of accessing the application. Separate keys may be enabled for different types of operations (reading, writing, card management) on each card.

Typically, each card has its own secret key or keys which can be derived using an application "master key" (which is present on every reader) and some other card-specific identifiers (such as the card serial number). This means that each card doesn't have the same secret key, so a compromise of one card's key does not compromise any other cards. On the other hand, every reader in a domain must share the same master key(s).

Pro: Strong "something you have" factor for smaller environments. Key cannot be copied or cloned without access to domain master key.

Con: Access to master key would compromise every card in that domain, permitting duplication of any card and access to any reader. Key protection issues significantly constrain the number of places that this authentication factor can be used. Cross-domain authentication in federated environments is largely impractical, particularly in disconnected environments due to master key management issues.

Symmetric keys on cards represent a high assurance factor for authentication in closed environments, but are not secure for use in inter-agency or disconnected environments.

2.4 Raw biometric templates

Some deployments of contactless storage cards such as DESFire use biometric templates to perform authentication using a "something you are" authentication factor. They do this by storing a raw biometric template in the card storage area in a read-only form. This template can be read off the card and compared against a user to help confirm the identity of the user.

This template can be represented using an older proprietary scheme, or could use forthcoming standard

representations defined under ISO/IEC 19794 [ISO04] or INCITS 377+ [INCITS04].

Pro: The biometric is tightly bound to the user.

Con: The biometric is not bound to the card or any identifying serial number, so it may be trivially copied to another card or emulator. This does not offer a useful identification factor in inter-agency or disconnected environments. The lack of any cryptographic protection would allow any biometric to be presented from a card.

Raw biometric templates constitute a low assurance factor due to their lack of strong binding and copy protection.

2.5 Signed biometric interchange files

Rather than storing raw biometric templates on cards, some groups have promoted the storage of one or more biometric templates on the contactless card with a digital signature to protect them from modification. These files would typically be written using a standardized signed interchange format such as CBEFF [PDR+01] or X9.84-2003 [ANSI03].

Pro: The basic representations in these standards provide a digital signature around the biometric template, which prevents the creation of arbitrary templates for non-registered users.

Con: Standard CBEFF and X9.84 do not provide strong binding to the card or any identification factors. The biometric template for any registered user can be copied to another real or emulated card, which provides no protection against duplication. Once a user has been registered (so they have a signed biometric template), they can reuse the generated interchange file indefinitely. In addition, if the card contains more than one biometric template, the reader must retrieve all of the biometric values (the entire CBEFF) before signature validation can be performed, which will negatively impact transfer speeds for the user. If each biometric template were split into a separate signed file, the time to retrieve one template would be reduced, but the total storage requirement would increase significantly due to the overhead from the interchange file format (dozens of bytes) and the digital signature (approximately150 bytes for RSA-1024).

Simple signed biometrics provide only basic assurance for interoperable and disconnected environments.

2.6 Signed biometric interchange files with card ID binding

Groups such as the Interagency Advisory Board task force are considering extending biometric interchange formats such as CBEFF to include the card serial number (CUID) in the signed CBEFF body to provide a stronger binding between the biometric template(s) and the card itself. As long as the CUID is treated as the primary identifier for the user, this provides protection against the transfer of identity to other cards.

Pro: Adding the card's CUID into the signed interchange file provides a strong binding to a unique identifier which mitigates against copying templates between cards.

Con: The CUID identifier may not be a sufficient reference ID for interoperability, since the CUID may not be securely known by other entities. This identifier is also not tied into the digital identity represented on the contact half of the card. As in **Error! Reference source not found.**, above, this representation will be expensive in either IO times or memory if more than one biometric template is stored on the card.

Adding the external CUID into the signed message provides a high assurance authentication factor.

2.7 Signed biometric templates with card ID and certificate binding

To provide a stronger binding to the user's overall digital identity, it would be straightforward to extend the logical scheme proposed by the Interagency Advisory Board Data Model Task Force to bind the biometric template to both the card (via CUID) and the user's more general digital identifier [IAB04]. This could be done by including the relevant serial number and issuer information from the cardholder's Identification digital certificate. This would provide binding to a universal unique identifier which would be strongly represented on both the contact and contactless interfaces.

The stored biometrics could be bundled together with this identifying information and bound using a single digital signature, as shown in Figure 1, below.

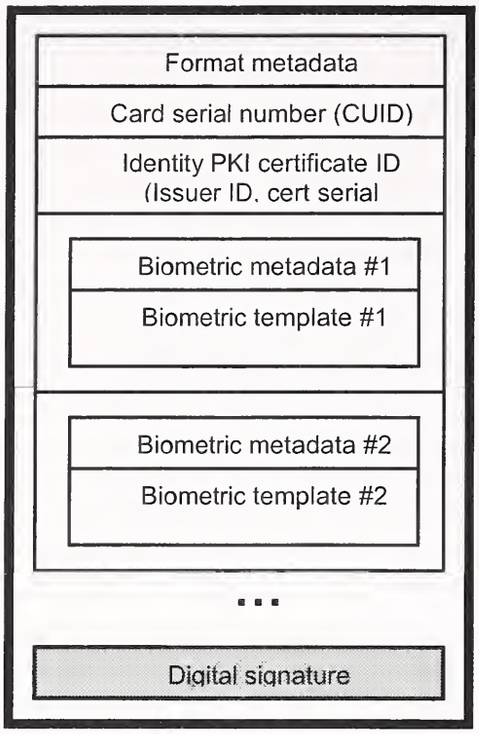


Figure 1

Alternately, each separate biometric template (fingerprints, hand geometry, iris scans) could be stored in a separate digitally signed format that is bound to the card and user, as shown in Figure 2, below.

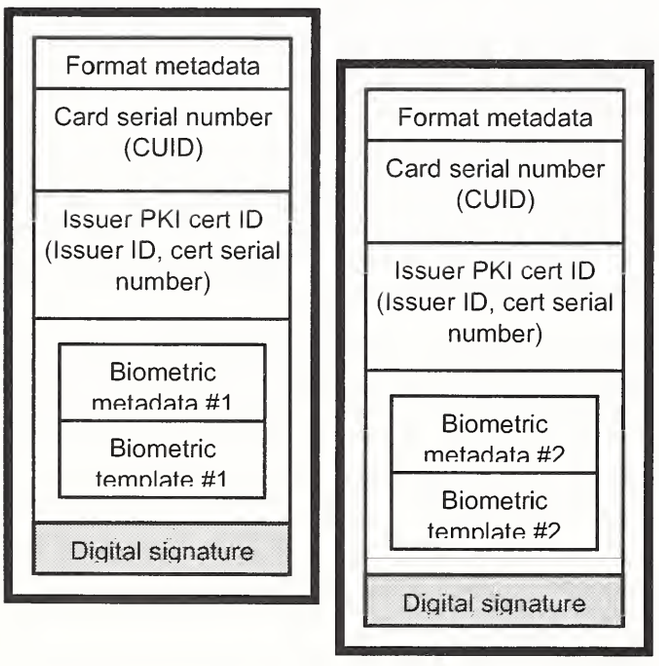


Figure 2

This logical representation could be expressed through a CBEFF Patron format in the same manner as the IAB's proposed data format. This representation could be as simple as adding a certificate identifier field into the IAB proposal.

Alternately, a different encoding could be used. For example, an X.509 Attribute Certificate [ISO01b] would provide a signed, extensible data format that uniquely binds the cardholder's identity certificate to one or more biometric templates, the CUID, and any other issuer-defined fields as needed. This would offer compatibility with existing standards and encodings with greater future flexibility.

Pro: Binding the biometric to the card's CUID and the user's cert ID provides a mapping that ties the biometric, the card, and the high-level digital identity of the user. This also permits a unified approach to identity management and validation, since the cert ID can serve as a universal identifier for all transactions. This could allow inter-agency identification through a federated identity instead of relying on pre-registration.

Con: If more than one biometric template is stored on the card, then this scheme will be either inefficient in data transfer times or storage usage. If one signature encapsulates all templates, then all templates must be transferred before any can be used. This may consume a significant amount of time due to the limited data transfer rates for contactless cards. If, on the other hand, each template is put into a separate digitally signed file, then the retrieval of one template is efficient, but a significant portion of the limited memory capacity of the card will be wasted with redundant data and extra digital signatures.

With either representation, digitally signed biometrics bound to cert and card IDs represent a high assurance authentication factor.

2.8 Signed biometric references with card and cert ID binding

As an optimization to the bound biometric templates described in 2.7, above, CoreStreet believes that the data storage and bandwidth aspects of signed, bound templates can be reconciled by signing secure *references* to biometric templates rather than the templates themselves.

Under this scheme, a digitally signed authentication file (e.g. Attribute Certificate) would be placed onto the card. Like the previous architecture, this message would bind together the card CUID, the cardholder's identity cert ID, and biometric information. However, rather than storing the entire biometric templates within the signed authentication file, this scheme would only store a one-way secure hash of each biometric template, as shown in Figure 3, below.

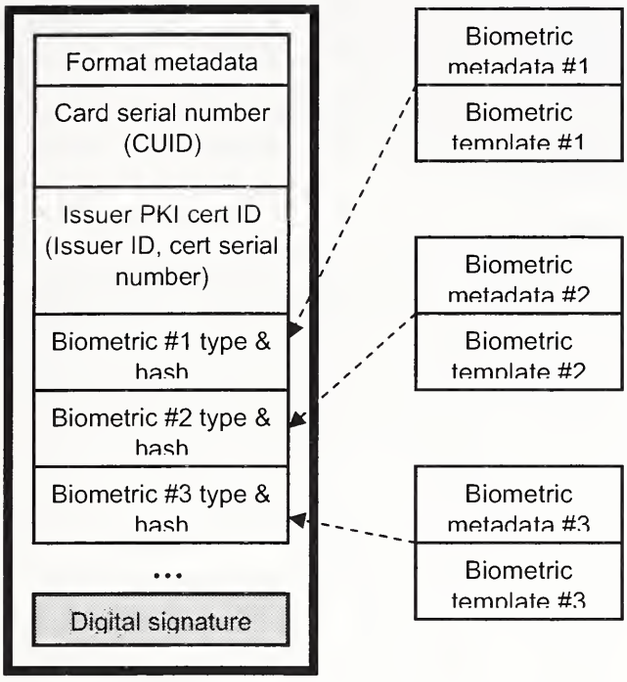


Figure 3

Using this scheme, the card stores a single authentication file, with a single copy of the binding information and the digital signature. For each biometric template on the card, this authentication file will contain a indicator of type (e.g. INCITS 377 Fingerprint Minutiae template) and a one-way secure hash of that particular biometric file. This would only add around 30 bytes per biometric to the base authentication file.

The biometrics themselves would be each stored in a separate file on the card. Each biometric would be unsigned. However, the single signature on the authentication file could be used to confirm the integrity of any of the referenced biometrics. We would recommend that the individual biometric templates be stored in separate card files in the same order that they are represented in the authentication file. For example, an application could be provisioned on the DESFire card with the following files:

File ID: 0	Signed authentication file (e.g. Attribute Certificate)
File ID: 1	Biometric template file #1: Finger #1 minutiae
File ID: 2	Biometric template file #2: Finger #2 minutiae
File ID: 3	Biometric template file #3: Iris template
...	...

Using this scheme, a reader capable of authentication using a particular biometric technology (e.g. Iris scan) could initially read File #0 to retrieve the signed master authentication file. This would contain strong binding to the card (which would be verified against the retrieved CUID) and the user's digital identity (attribute certificate). This initial file would be relatively small (200-300 bytes) since it does not contain any of the biometrics.

After reading and verifying the authentication master file, the reader could determine that the desired template type (iris template) is located in File #3. This template could be retrieved without touching any of the other biometric templates on the card. Its integrity could be confirmed by hashing its bytes and comparing against the master authentication file.

Pro: Permits strong authentication in federated and disconnected environments with minimum of wasted data and communication. Optimal scheme when multiple independent biometrics are represented on the card.

Con: Small storage overhead (~30 bytes) if only a single biometric template is stored on the card. Data model and representation not defined by existing standard (e.g. CBEFF).

Use of this type of strongly bound signed biometric represents a high assurance authentication factor.

2.9 Contactless PKI

For comparison, it must be noted that cards are currently available that can perform asymmetric operations on a contactless (ISO 14443) interface. For example, Oberthur currently distributes FIPS-certified contactless cards based on Philips chips that can perform RSA operations on both contact and contactless interfaces. While this technology has not been selected for the current generation of government smart cards, the protocol-level compatibility could permit a simple transition in the future.

These contactless capabilities could be enabled by either linking the contactless antenna to the contact chip (dual-interface) or else by integrating an independent contactless chip (combo card).

Pro: Provides strong authentication without requiring access to biometric information.

Con: Dual-interface cards may introduce security and privacy concerns if access becomes available to sensitive applications on the contact chip. Combo cards may significantly increase per-card costs over simpler symmetric chips like DESFire.

Use of contactless cards with private key capabilities would represent a high assurance factor.

3 Biometric Considerations

The previous descriptions of biometric-based authentication assume the existence of ideal biometric algorithms that are acceptable for widespread usage. For real-world applications, the use of biometric templates may introduce several issues.

3.1 Privacy

The collection of biometric information by government agencies can raise concerns that this data may be misused. For example, a database containing the fingerprint templates of all government-affiliated individuals could be a tempting target for someone wishing to establish the owner of a latent fingerprint. Similarly, a database of face images could be searched to identify lawful protestors.

This concern for biometric searches (1-to-N) may be lessened by an approach that only stores biometric information on user cards. The schemes, above, would permit an attacker up to a meter away from a user to silently pull the user's biometric templates along with the user's serial number(s).

This attack should be contrasted with the ability of a nearby attacker to gather equivalent data through more prosaic means. For example, a facial image on the card would be more difficult to capture than a snapshot from a digital camera. A fingerprint template would be no easier to retrieve than a latent fingerprint left by the cardholder.

More importantly, the biometrics on the card should not be tied to identifying biographic information. The schemes, above, recommend binding the biometrics only to arbitrary serial numbers, not biographic identifiers such as name or social security number. This means that a passive reader in the Pentagon Metro station may be able to silently retrieve a large number of government fingerprint templates, but these would be no more useful for building an identification database than random fingerprints from the subway's poles.

3.2 Forgery

Another possible concern with the use of biometric templates is the potential for an attacker to use the template as a basis for a forged biometric that could fool some sensors. For example, a facial image suitable for face recognition could also be used to create a printed image capable of fooling some face recognition systems.

This property of biometrics also prevents the effective revocation of the biometric factor if it is ever compromised. Unlike a private key, which can be revoked and replaced, a duplicated finger cannot be comfortably discarded.

The ability of an attacker to forge a biometric authentication factor depends on the countermeasures provided by the biometric vendors. For example, advanced fingerprint sensors attempt to detect the difference between live fingers and duplicates using proprietary detection of temperature, moisture, conductivity, etc.

3.3 Interoperability

In spite of ongoing efforts to standardize biometric templates and sensors, unacceptable incompatibilities may exist between templates and algorithms from multiple vendors. It is believed that these interoperability issues will improve as the relevant standards are finalized, but this may not provide an adequate solution for the current generation of cards.

This may require a fallback from efficient representations (e.g. fingerprint minutiae) to bulkier forms (e.g. full fingerprint images) that may exceed the storage capacity of contactless cards.

4 Secure Contactless Authorization

If contactless authentication is performed using only factors that are bound to the card's serial number (CUID) or domain-specific authentication string (e.g. CHUID), then any solutions to validate and authorize the cardholder will be inherently limited to the physical access domain, since there is no strong tie to the digital identity represented on the contact interface of the card.

If, however, the authentication is tightly bound to the cardholder's digital identity, as represented by their identification public key certificate, then the same unique identifier can be used for both contactless physical access and contact PKI transactions.

This property permits a unified approach to securely managing the privileges and revocation of a cardholder. For example, OCSP [MAM+99] or CRLs could be used to determine whether the cardholder has been revoked, and this same scheme would be usable for both physical and logical access. Privileges could be securely delivered for use in both physical and network environments.

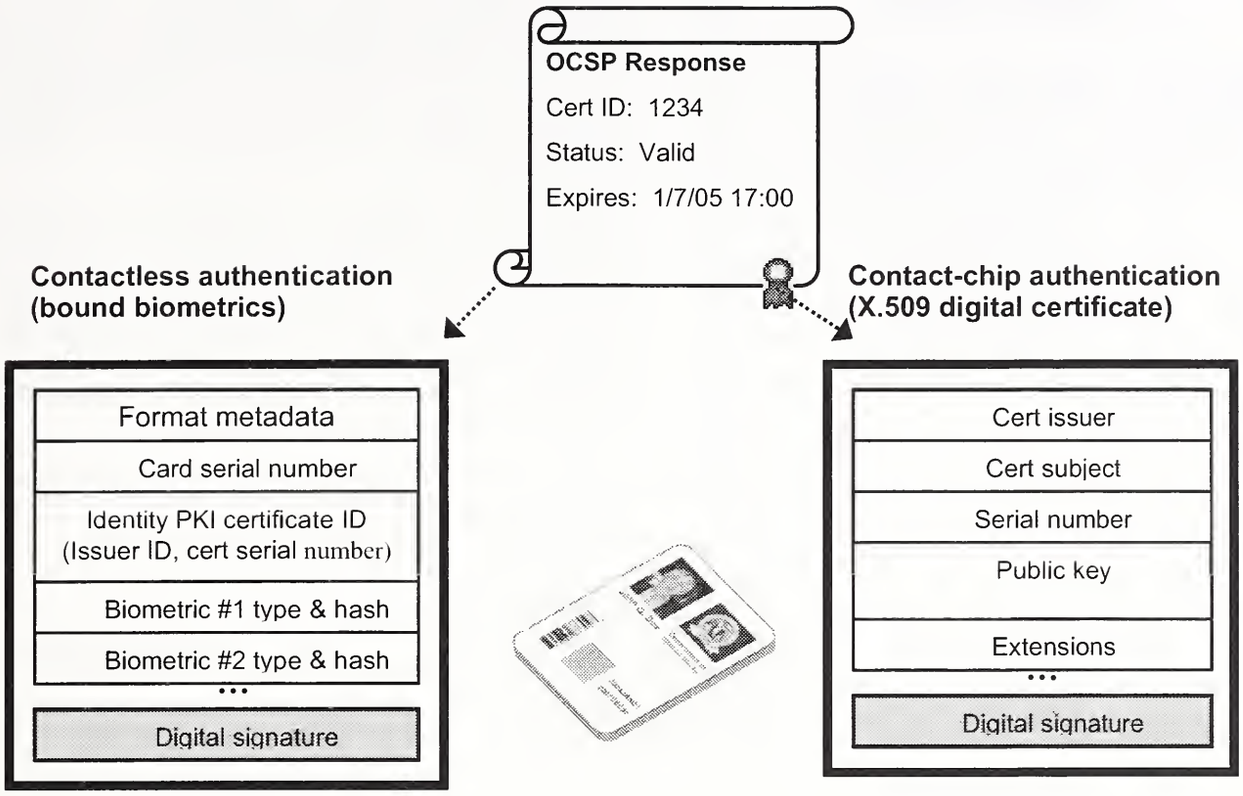


Figure 4

4.1 Unified authorization messages

Figure 4 shows the same authorization message, in the form of a digitally signed OCSP Response, being used for physical and logical access for the same card. This authorization message could be represented using any other desired standard such as a digitally signed SAML assertion [OASIS02], an X.509 attribute certificate, etc.

This scheme also provides a smooth migration to dual-interface cards where the same general applications would be available though either the contact or contactless (T=CL) interfaces. By logically identifying users using their cert ID today in a DESFire contactless environment, there is an easier migration to a future when the public key identity applet itself is available for secure asymmetric challenge-response authentication.

4.2 Offline authorization

If authentication factors such as signed, bound biometrics are available on the contactless interface, then strong authentication can be performed in offline settings without any access to an online directory. Similarly, secure authorization can also be performed in offline settings by storing signed authorization messages on the contactless interface.

Each authorization message is strongly bound to the cardholder's digital identity by including the cardholder's identity certificate ID within the signed message body. Any reader can inspect the authorization message to confirm its integrity and timeliness, and then use the validation and privilege information to grant access.

Rather than proscribe a particular authorization message format for the entire government to permit inter-agency and offline authorization, CoreStreet recommends that the government permit the allocation of a reusable "authorization container" on the contactless card that may be used to store any authorization information used within an individual organization.

Figure 5 shows the structure of a possible general authorization container on a contactless DESFire card.

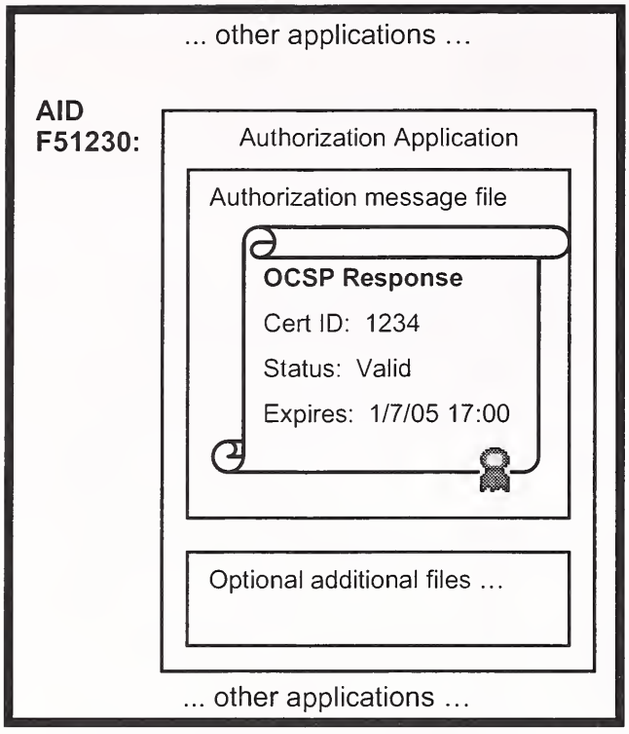


Figure 5

For example, the inter-agency standards bodies could define an application identifier (AID) and file number which has free read/write access. A cardholder with this applet could arrive at one agency, which may put a signed OCSP response onto her card to indicate validity and privileges. Offline readers in that agency would retrieve this OCSP message and use it to determine access privileges. At a second agency, the cardholder's card may be loaded with a signed SAML assertion of the user's privileges, which would be used to determine access at offline readers within that second agency.

The three important characteristics of this authorization container are:

- Standardized location on the card (3-byte AID on DESFire cards)
- Unrestricted read/write access
- Sufficient space for signed data (ideally, at least 1kB)

In Figure 5, the authorization message file is currently holding an authorization message in the form of an OCSP Response, but the authorization message could be any digitally protected format that is strongly bound to the identification credential.

This scheme would permit the greatest flexibility for supporting inter-agency and offline authorization by allowing each organization to locally specify their chosen representation, while guaranteeing that the card hardware in use by different agencies will interoperate.

5 Conclusions

The current generation of contactless identification cards have limitations which make it difficult to provide strong authentication for large, federated environments. Various approaches achieve different choices to balance scalability, security, and performance for physical access control.

If strong authentication is required for federated environments, we believe that this can only be achieved using either strongly bound biometrics or contactless public key support. Unfortunately, these approaches may run into issues of privacy and cost which could prevent them from being adopted. Lower-assurance alternatives may result in a higher risk of compromise through cloned identification credentials.

Strong federated authorization, on the other hand, may be possible under either scheme through the use of signed authorization messages for access control.

6 References

[ANSI03] American National Standards Institute. *Biometric Information Management and Security for the Financial Services Industry.* ANSI X9.84-2003. 2003.

[INCITS04] InterNational Committee for Information Technology Standards. *Information Technology – Finger Pattern Based Interchange Format.* January 2004.

[IAB04] Interagency Advisory Board (IAB) Data Model Task Force. *IAB Data Model Task Force Report v0.6 (Draft).* October 2004.

[ISO01] International Standards Organization. *Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Parts 1-4.* ISO/IEC 14443. 2000-2001.

[ISO01b] International Standards Organization. *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.* ISO/IEC 9594-8. August 2001.

[ISO04] International Standards Organization. *Information technology -- Biometric data interchange formats.* ISO 19794. Under development, 2004.

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP.* IETF RFC 2560. June 1999.

[OASIS02] Organization for the Advancement of Structured Information Standards

(OASIS). *Guidelines for using XML Signatures with the OASIS Security Assertion Markup Language (SAML).* Draft 02. September 2002.

[PAIIWG04] Government Smart Card Interagency Advisory Board's Physical Security Interagency Interoperability Working Group, *Technical Implementation Guidance: Smart Card Enabled Physical Access Control Systems, Version 2.2.* July 2004.

[PDR+01] Fernando Podio, Jeffrey Dunn, Lawrence Reinert, Catherine Tilton, Lawrence O'Gorman, M. Paul Collier, Mark Jerde, Brigitte Wirtz. *CBEFF: Common Biometric Exchange File Format.* NISTIR 6529. January 2001.

[SDW+03] Teresa Schwarzhoff, Jim Dray, John Wack, Eric Dalci, Alan Goldfine, Michaela Iorga. *Government Smart Card Interoperability Specification, Version 2.1.* NIST Interagency Report 6887. July 2003.

[SEIWG02] Security Equipment Integration Working Group, US Department of Defense. *Access Control Technologies for the Common Access Card.* April 2004.

Identity-Based Encryption with Conventional Public-Key Infrastructure

Jon Callas
PGP Corporation
Palo Alto, California, USA
jon@pgp.com

18th February 2005

Abstract

This paper proposes an identity-based encryption (IBE) scheme based on traditional public-key cryptographic systems, such as RSA, DSA, Elgamal, etc. This scheme has a number of advantages over other systems. It can rely upon these traditional systems for its security. Since it uses these traditional encryption schemes, it is interoperable with and easily embedded within an existing security system that uses these functions. Additionally, its construction as an on-line system avoids the operational security flaws of IBE systems that allow off-line key generation.

1 Introduction

Conceptually, public keys behave a lot like telephone numbers — if I want to call you, I need your telephone number. I don't need my own number to make calls (I can use a pay phone, for example), I need one to receive them. In a fashion that is analogous to a telephone number, I need *your* public key to encrypt something so that it is secret to you.

Unlike telephone numbers, public keys are far too big for people to remember. Even elliptic curve keys, which are much shorter than the traditional ones are far too large for a person to remember. George Miller's classic research [MILLER56] done on telephone numbers is that the average person can remember seven give or take two digits. A 160-bit key will be something over 40 digits long (exactly 40 if we use hexadecimal). So memorizing someone's key the way you memorize their phone number is completely out of the question.

Consequently, we need to have some blobs of data that say that a name such as `alice@example.com` belongs to some key. There is also value in digitally signing the blob so that the receiver has some assurance that the association is accurate. These blobs are *certificates*.

Like any data management problem, certificate management is harder than people would like. This is why in 1984 Adi Shamir suggested the idea of coming up with a crypto scheme in which any string can be a public key [SHAMIR84]. Thus, there is no need to associate a name with a public key, because they're effectively the same. This is *Identity-Based Encryption*.

While typically we think of IBE systems converting names into public keys, it should be possible to make any arbitrary bit-string, $\mathsf{ID} \in \{0,1\}*$, a determinant of a public key in an IBE system. Thus, a name, an email address, or even a binary object such as a picture or sound can be considered equivalent to a public key. Thus, an IBE system can be thought of as a function of the form $K_i = IBE(\mathsf{ID}_i)$ that produces keys from arbitrary bit strings that we call identities, without loss of generality.

2 Overview

IBE can also be thought of as an *Attribute-Based Enrollment* mechanism. Its goal is to reduce the overhead required to bring an entity into the system. Thus, we take some attribute of the entity and use that as a functional equivalent to a public key.

In the past, work on IBE has been mathematical. It has developed a new public-key cryptosystem that has as a part of its key creation some arbitrary bitstring that is the identity. We examine this past work and look at how they are put together, as well as the limitations on these previous systems.

Next, we construct a framework for an IBE system that satisfies the basic goals of IBE — that this attribute of an entity, its so-called identity is equivalent to a public key — and uses a parallel structure. However, this new framework can construct key pairs that are of a familiar cryptosystem such as RSA, rather than requiring its users to adopt a new public key algorithm.

This construction also differs from present IBE systems in that it does not allow off-line generation of keys, but we also note that off-line generation has security drawbacks as well as advantages. However, on-line generation also permits a hybrid PKI that has both traditional and identity-based aspects in the same infrastructure.

Lastly, we look at open and unsolved problems that surround IBE systems in general, including this one. All IBE systems created so far have a set of limitations as well as characteristics that are not yet solved. They do not remove the utility or desirability of IBE, but do limit where it can be effectively deployed.

3 Components of IBE

An IBE system contains four basic components in its construction:

1. **System Setup**: IBE systems rely upon a trusted central authority that manages the parameters with which keys are created. This authority is called the *Private Key Generator* or PKG. The PKG creates its parameters, including a master secret K_{pkg} from which private keys are created.

2. **Encryption**: When Bob wishes to encrypt a message to Alice, he encrypts the message to her by computing or obtaining the public key, P_{Alice}, and then encrypting a plaintext message M with P_{Alice} to obtain ciphertext C.

3. **Key Extraction**: When Alice wishes to decrypt the message C that was encrypted to her name, she authenticates herself to the PKG and obtains the secret key S_{Alice} that she uses to decrypt messages.

4. **Decryption**: When Alice has C and S_{Alice}, she decrypts C to obtain the plaintext message M.

No matter the specific parameters or requirements of the system, these functional aspects are always present in IBE systems as their defining components.

4 Previous Work

Shamir's original system was based upon RSA encryption and is a signature-only system. Shamir was unable to extend it to an encryption system. Between 1984 and 2001, a number of IBE systems were created, but they all had limitations, such as requiring that users of the system not collude, or requiring large amounts of computation on the part of the PKG. In 2001, two new proposals were published, improving on previous work.

Clifford Cocks created a scheme based upon quadratic residues [COCKS01]. Cocks's system encrypts bit-by-bit, and requires expansion of the message; for a 1024-bit modulus and a 128-bit bulk encryption key, 16K of data must be transfered. With modern networks, this is a completely acceptable overhead[1].

Dan Boneh and Matt Franklin created a scheme based upon Weil Pairings [BF01]. Pairing-based systems use bilinear maps between groups to establish a relationship whereby hashes of the identity create the encryption scheme. Boneh-Franklin IBE has had further work [BB04] and is an active area of research.

Horwitz and Lynn [HL02], Gentry and Silverberg [GS02] improved upon performance characteristics of a Boneh-Franklin PKG by extending IBE systems to *Hierarchical IBE* (HIBE). Their work is important particularly because of its attention to the practical details of constructing a scalable PKG. Gentry also described *Certificate-Based Encryption* (CBE) that uses an IBE system with certificates to create a hybrid approach [GENTRY03] that essentially makes the "identity" not be a name. but a well-defined certificate. In a conceptually related approach, Al-Riyami and Paterson have their *Certificateless Public Key Cryptography* [AY03].

Benoît Libert and Jean-Jacques Quisquater also created an identity-based signcryption scheme based on pairings [LQ03]. These signcryption schemes combine both aspects into one operation. There is other somewhat related work on combining signing and encryption as well such as [ZHENG97].

[1]Other discussions of IBE have characterized this expansion as an *unacceptable* overhead. Debating how much expansion is tolerable is orthogonal to this paper, but I feel it necessary to explicitly state that I find acceptable what previous authors find unacceptable. Networks continually get faster. Many messages are small enough that other overhead they already have to deal with (like conversion to HTML) also expand them. In the case where the message is large, clever software engineering could use compression and efficient bulk encryption to make this no worse than other message bloat.

5 Limitations on Previous Work

All of the existing IBE systems have their own limitations. Shamir's system signed but did not encrypt. Cocks's system needs care to avoid an adaptive chosen ciphertext attack. It is also inefficient, but still efficient enough for use on reasonable communications paths. While others have proofs of security, there is a notoriously poor relationship between proofs of security and actual system security. Security proofs can show where a system is safe, but not protect against new assumptions that an adversary can bring to bear against the system nor against uses of a system that its creators did not think of which may be outside of the scope of the original threat model. Still other subtle problems have shown up on other systems, such as the ability in early HIBE systems for colluding users to determine the PKG's master key.

With the exception of Shamir's system, IBE systems rely on new public-key cryptosystems, most often Weil pairing. Consequently, they are not compatible with existing systems that use RSA, Elgamal, or DSA. This limits their practical application, since there are many existing systems built upon these cryptosystems. Also, experience and comfort with the security of these established systems is high. A key advantage that Shamir's system has over all those that follow it is that it was based on established public key cryptography, and thus (had it been successful in being both a signing and encrypting system) interoperable with non-IBE systems. Had Shamir's system been successful at encrypting, an RSA-based IBE system would likely be the dominant IBE system today, if for no other reason than its interoperability with deployed systems.

This is an important observation — if we can construct an IBE system that uses traditional, integer-based, public key cryptography, the barriers to adoption of IBE systems might be lowered. The value that IBE has can be fully realized if it can be made to work with these established systems. Furthermore, this system has the advantage that it can rely on twenty years of mathematical and operational familiarity with these traditional public-key cryptosystems.

6 Security Parameters of the Off-Line and On-Line worlds

Previous IBE systems have as a desirable property that they support off-line generation of keys. That is to say, Bob receives key-generation parameters from the PKG once, and then can generate an arbitrary number of public keys.

While off-line key generation is desirable, it is not without its own security consequences.

6.1 Advantages of Off-Line Generation

Off-line generation is ideal in an off-line environment. If communication with the PKG is slow, expensive, or unreliable, then off-line generation is a huge advantage to its users. They need only one interaction with a given PKG to be able to do all needed work with that server.

This advantage becomes less, however, as communication with a PKG becomes cheaper, easier, and faster. One some level, off-line key generation is nothing more than a key server that is an algorithm instead of a database. This is an advantage when databases are static and expensive, but not when databases are cheap and fast. In an environment where the *contents* of the database

are dynamically changing, a database change is not only an algorithm change, but an algorithm change that must be propagated to all clients of the PKG.

6.2 Disadvantages of Off-Line Generation

Oftentimes, the strengths of a system are also its weaknesses. This is also true with off-line generation. Off-line generation makes key generation easy not only for legitimate users of of the system but for illegitimate ones.

An issue that PKIs must consider in their design is that of a *Directory Harvest Attack*, in which senders of unwanted advertisements or outright fraudulent confidence games use the directory as a way to discover information paths into the system. Off-line generation of keys allows spammers and other attackers to to pre-generate email attacks in their own system or create a distributed system for encrypted attacks. These attacks are not an issue in off-line systems.

Off-line generation has the disadvantage that there is complete transparency in the directory, since the directory is an algorithm. Anyone with that algorithm has all possible entries in the directory and their public keys, and this can be exploited in side-channel attacks that are not attacks on the cryptographic system *per se*, but the way the system is used.

Off-line generation has as an additional disadvantage increased revocation problems. A conventional PKI must be able to re-issue certificates and handle for revisions in the PKI. An off-line IBE system must not only handle revocation of the certificates themselves but a revocation of the *algorithmic parameters* that comprise its own PKI. No IBE system before this one has even considered this real-world problem.

In fact, the key advantages of this on-line system are that it considers and solves these real-world problems.

6.3 On-Line IBE for the On-Line World

Sadly, trends in the real world make the advantages of off-line IBE moot, and turns its disadvantages into outright security problems. There is little need for off-line generation in an on-line world, and the advantages of off-line generation benefit attackers more than defenders.

Nonetheless, IBE has desirable characteristics. The core IBE concept, that there is an equivalence relationship between bit-strings and keys has appeal. Designing an IBE system that has the advantages of name-to-key mapping without the security flaws of off-line key generation can make IBE acceptable to the complex security requirements of the Internet.

Furthermore, if we shift the IBE system to an on-line system, we can construct it so that it uses traditional keys. This permits an IBE system to be embedded within an existing cryptosystem and interoperable with existing systems that use these keys. Not only does this remove adoption issues, but it also simplifies proofs of security; it is trivial to prove that an encryption portion of an IBE system is as secure as RSA if the underlying encryption is RSA.

Another advantage is that an on-line system can normalize the identity. It is common for users of an email system to have equivalent identities on the system. For example `alice@example.com`

and `asmith@example.com` may be the same user, and it is desirable to have only one key. An on-line system can canonicalize identities at runtime.

Finally, and perhaps counterintuituvely, this permits IBE keys to be used in certificates. We usually think of IBE as a way to eliminate certificates. However, all keys require standard data structures for transport. Whatever flaws they have, certificates are existing, standard ways to format key material in a way that systems can reliably use them. Objections to certificate-based systems are not objections to the *certificates* per se, but to the *certification process*. Without a standard set of transport data structures, IBE proponents must standardize on key transport data structures and convince developers to use those structures as well as the new crypto algorithms and protocols. Using existing certificate systems reduces the Key Extraction problem to an existing problem that has a simple solution, e.g. a lookup in a directory.

Combining certificates with IBE is not new to this proposal. Gentry's CBE combines a form of certificates with Weil pairings.

On-line systems are ubiquitous and becoming more available every day. Consequently, the advantage of off-line key generation in an IBE system not only has less value today than it did when Shamir first suggested IBE in 1984, but new attacks turn it into a boon for the attacker of a system. Relaxing the parameters of an IBE system so that Bob is required to ask the PKG for each key is certainly practical, and permits us to exploit these other desirable system features.

7 Constructing IBE to Use Conventional Cryptography

It is a goal of this system to describe how to construct an IBE from well-known components that have easily-understood security constraints, including proofs of security. Thus, what follows is actually a *adaptive framework* for constructing an IBE system that is not bound to a single algorithm and is functional even in the face of security advances such as new attacks on hash functions [BIHAMCHEN04] [JOUX04] [WANG04].

7.1 System Setup

Setting up the PKG consists of the following steps:

1. The PKG selects a master key, K_{pkg}. This key must be selected with care, as the security of the underlying system can be no more than the security inherent in this key. This key may be a symmetric key, or an asymmetric key.

2. The PKG selects an *Identity Digest Function*, IDF. This is a pseudo-random bit function of the identity, ID, and K_{pkg} that gives an *Identity Digest Token*. IDT such that $\mathsf{IDT} = \mathsf{IDF}(K_{pkg}, \mathsf{ID})$.

   The IDF can be a symmetric-cryptographic function using the K_{pkg} as some simple secret. For example, it could be a an HMAC, a CBC-MAC, or some other suitable pseudo-random bit function. The IDF may also be an asymmetric-cryptographic function such as RSA, in which case K_{pkg} might be an appropriately strong RSA key and IDT is thus the result of an

RSA encryption of either ID directly or a hash of ID. Note that in this and similar cases, padding must be considered carefully to preserve the needed determinism of the IDF as it establishes a one-to-one correspondence between ID and IDT. Without a one-to-one correspondence, then this is not an IBE system at all.

It may be desirable for this selection to be part of the setup; the PKG could be built with a number of options of IDF, one selected at setup time.

Regardless of IDF selection, the resultant IDT is a limitation on the security of the IBE keys. If, for example, it were the CBC-MAC of a block cipher with a 64-bit block, then the underlying system has a birthday attack on the IDT that is probably less than the other parameters of the system. Selecting the IDF requires analysis of the overall system lest this be the security bottleneck of the system.

3. The PKG selects a deterministic pseudo-random number generator, RNG that will be seeded with IDT. This RNG is not the same function as IDF as it will in turn be used by a key generation function, Kgen, that generates an IBE key pair. This would be an RSA, DSA, Elgamal, or other key generation function[2]. Of course, it itself must be deterministic, as the same key must be generated any time a given identity is put into the system.

This construction has advantages beyond the simplicity of being able to use any key type within an IBE system. The security of the system relies on previously-studied components, which provides for easier security analysis. It also implicitly guards against some forms of attacks, such as collusion attacks. Breaking the $K_{pkg}$is as hard as breaking known forms of cryptography. So long as a suitable IDF function is selected, the whole Kgen process is as secure as its underlying cryptographic subsystems.

7.2 Key Extraction

When the PKG is requested for a key for a given ID, it follows the following process:

1. The PKG produces an IDT, such that $\mathsf{IDT} = \mathsf{IDF}(K_{pkg}, \mathsf{ID})$.
2. The PKG seeds RNG with IDT.
3. The PKG generates a key with $\mathsf{Kgen}(RNG)$ to produce the appropriate IBE key pair, IKP_{ID}.
4. If the PKG has an unauthenticated request for the given ID, then it responds with $IKP_{\mathsf{ID}_{public}}$. This happens when Bob asks for Alice's key.
5. If the PKG has an authenticated request for ID, such as when Alice asks for her own key, then the PKG responds with both $IKP_{\mathsf{ID}_{public}}$ and $IKP_{\mathsf{ID}_{private}}$.

At this point, Alice and Bob each have the appropriate piece(s) of a conventional key pair and they use it normally.

[2]Without loss of generality, the Kgen function can also be a function such as an elliptic-curve key generator. However, since one of the advantages of this design is that it produces keys that are usable within widely-used systems. When elliptic-curve systems are more widely used, it will be trivial to extend this to an IBE system based on them.

7.3 Encryption and Decryption

Encryption and decryption are trivial: they are simply the encryption and decryption functions of the base cryptosystem of the IBE keys. Note that if the cryptosystem is a signature-based cryptosystem such as DSA, it is signing and verification rather than encryption and decryption.

8 Security Limitations

As with all IBE systems, there are a number of security limitations of this system. However, in all cases the limitations of this system are no different than for other IBE systems.

8.1 Key Escrow Problem

IBE systems are effectively key escrow systems. It is a limitation, if not an outright flaw of IBE that the PKG holds all the parameters needed to generate any key pair, if not the key pair itself.

Consequently, Bob can never be completely assured that Alice and only Alice can decrypt a message or created a signature. In the real world this is less of a problem than it is in theory, as the security Alice's secret key is always bounded by the operational parameters of her key storage. It is undeniable, however, that an RSA key generated on a secure token is going to be more secure than one generated in a PKG.

IBE systems, including this one, may be unacceptable for some uses. If there is a legal requirement that Alice's private half of her signing key be in her possession alone, then no IBE signing system will be acceptable.

Boneh and Franklin suggest a partial solution to this problem. In their partial solution, their master key can be split using a secret-sharing system [SHAMIR79]. This has the advantage that no single entity has any of the core secret parameters. An adversary would have to compromise enough members of a set of PKGs to reconstitute the secret. Nonetheless, this is only a partial solution. At some point, the set of PKGs must reconstitute the parameters, and an adversary that sufficiently compromises the appropriate member can still get the parameters. Furthermore, since the members of the PKG set are likely to be close to identical, they are not independent in their security. If an adversary can compromise one member of the set, it is more possible if not likely that the adversary can compromise the whole set.

Another solution would be to keep the master parameters in secure hardware, or even secret-shared across a set of pieces of secure hardware. But this adds complexity on top of complexity to the system.

In this system, we accept that the IBE parts of this system are by necessity a key escrow system, but note that it can fully interoperate with another other PKI that is not a key escrow system. Furthermore, this system can be integrated with a more secure public key system to provide it with IBE features. For example, the IBE in this system gives a way that keys can be created for roles such as *Security Officer* or *Ombudsman* without pre-defining these roles or their owners prior to use. This is another advantage to merging IBE aspects into conventional PKI. Within a given

PKI, you can have parts of it that are IBE-derived, and parts that are fully-secure, edge-generated public key pairs. Moreover, they all interoperate seamlessly.

8.2 Security of Key Generation

The security of the keys generated by the PKG are bounded by the selection of the underlying functions as well as the scale of the PKG. If the PKG is to generate many, many keys, then factors such as the possibility of identity collision have to be taken into account as well.

This is not an intractable problem — there are many underlying functions that can be used for components of the PKG that have adequate security parameters for security. It must simply be noted that these are security design factors that are unique to an IBE system.

8.3 Security of Key Extraction

When Alice extracts her private key from the PKG, the PKG must deliver it to her securely. There are many ways to do this, including secure network connections such as TLS [TLS]. It also must be packaged securely (and this is another place where existing data structure systems such as certificate standards gain help). This is again, not precisely a security problem but more of where the PKG builders must take care in their delivery system.

9 Open Problems

IBE is not yet a mature discipline. There are a number of open problems beyond improving the security of the underlying system that are yet to be solved. Here is a short discussion of some of them.

9.1 Removing Key Escrow

All IBE systems, including this one, accept the fact that they are key escrow systems. However, nearly any discussion of IBE includes a class of people who consider the escrow aspect to be a severe flaw. It certainly makes the system brittle, as security of the system relies on non-mathematical security. A real-world PKG *must* be an un-hackable computer, even if that computer has advanced mathematical techniques such as secret-sharing as an additional bit of armor. It makes the simplicity that IBE gives on one axis be balanced by complexity on another.

As cryptographers and systems designers, we have accepted key escrow as a part of the playing field because if we don't, there's no IBE. In an academic paper, this is a reasonable assumption, but in the larger world, this assuming key escrow as an implicit part of the system cannot be simply brushed away.

This is a large open problem with no good solution. IBE exists for operational simplicity, but has operational complexity as a cost. Removing that cost should be the primary goal of future work, in this author's opinion.

9.2 Is it Possible to Eliminate Certificates?

The whole *raison d'être* for IBE is to "solve" the certificate problem. However, this means that IBE assumes that it is *possible* for a certificate to consist solely of a name and a key. In the real workd, certificates have always been more than a mere binding between a name and a key: they also carry metadata about the name, the key, parameters of use, and even metadata about the metadata.

One of the most important bits of metadata about a key is revocation data. If a name *is* a key, then it is not possible to revoke the key without revoking the name as well. The utility of Alice not having to have a certificate is small if she must revoke her email address if she loses a smart card. Furthermore, telling everyone who has Alice's email address that they must use her new one (and thus new key) is *precisely* a key revocation problem with added disadvantages for Alice.

Boneh and Franklin [BF01] suggest a clever solution to this situation. In their paper, they suggest that ID_{Alice} not be `"alice@hotmail.com"` but `"alice@hotmail.com || 2004"`. They even suggest that the PKG can create daily-use keys such as `"alice@hotmail.com || February 29, 2004"`.

As elegant as this solution is, it prompts other questions. Once an identity is not simply a name, but is now a name and a date, is it still Identity-Based Encryption? Phrased another way, isn't `"alice@hotmail.com || 2004"` merely a different way to code a certificate?

Implementing this solution also requires other surrounding standardization that detracts from the essential simplicity of the IBE concept. At this simplest form, you have to standardize on what a date is. This isn't difficult, but you have to do it. You must also translate malformed dates (perhaps `"2 Février 2004"` into `"02/02/2004:12:00:00.00UTC+0100"` which again detracts from the simplicity of IBE, as this is no longer something that a human being can reliably type the way that they can reliably type `"alice@hotmail.com"`. However, this is a problem that can be solve through software, an one where an on-line system has an advantage as it can canonicalize time in a central place, or even round to an internal epoch.

Previously in this paper, we discussed the algorithmic revocation problem as well. No IBE system before this one has even considered how the IBE parameters, the IBE algorithm itself, can be revoked. The fact that IBE is brittle in its reliance on central secrets makes this lack a larger open problem.

Lastly, there is no means in an identity to express variables within a cryptosystem. There can be no negotiation about acceptable block ciphers, data compression, data MACing, both start and stop date of a given key, and so on. An IBE system cannot help but be a one-size-fits-all system for these parameters. This may not be bad, it may actually be a simplifying assumption. However, expressing these concepts are part of why we have certificates despite the problems in managing them.

There are two possible approaches to dealing with this paradox — one being to make an IBE system that codes a more formal certificate and then uses that as an IBE key, such as Gentry's CBE, or this approach which adapts IBE so that it can be used within a traditional certificate system.

9.3 Is it Possible to Prove Ownership of a String?

When we describe how IBE works, we get to the Key Extraction phase and glibly say Alice authenticates herself to the PKG to get her private key. How?

If Alice is already an authenticated user of the PKG, this isn't very difficult. If it is to `hotmail.com` that Alice must prove ownership of `alice@hotmail.com`, this is an easy problem. If worst comes to worst, she has a password she can type in.

If it is to `brand-new-service.com` that Alice must prove ownership of `alice@hotmail.com`, it is a bit more difficult, but hardly impossible. A simple, if low-security mechanism is for `brand-new-service.com` to send her an email with some authentication token that she delivers back to `brand-new-service.com`. For those who believe this insufficiently secure, there are other protocols that are easy to devise that are more secure. For example, Alice could generate an ephemeral RSA key pair, give `brand-new-service.com` the public key, and then deliver to `brand-new-service.com` the decrypted authentication token as before. While not perfect, it's an improvement. Devising a protocol that is immune to man-in-the-middle attacks is left as an exercise to the reader.

However, if Alice must prove the ownership of `"Alice Jones"`, then we have a very difficult problem. Names are hard to express in a certificate system, and among the many criticisms of certificate systems the most basic objections concern the way they handle naming [ELLISON00]. If a name is a key, then a certificate is a key, and all the naming problems we have in certificates we have in names. Making names be keys exacerbates this problem.

If the IBE system uses other bit strings such as photographs, music, etc. as keys, proof of ownership could be arbitrarily hard, both technically and legally.

9.4 Performance Bottlenecks

IBE systems in general suffer from centralization on many fronts. Not only does centralization create security issues, but it also creates performance issues. The PKG may have to do large amounts of work, especially when the system uses many short-lived keys. In this system, the need for the PKG to generate keys makes more work for it. Furthermore, generating a key for some cryptosystems such as RSA require more computation than for others, such as DSA.

One possible solution to this problem is HIBE. HIBE expresses the user's identity as a composite of identities. For example, Alice's identity would be the tuple $(\mathsf{ID}_{Alice}, \mathsf{ID}_{hotmail.com})$ [GS02] [HL02]. While attractive from a performance viewpoint, it also blurs the conceptual simplicity of a name being a key. It also requires that the identities themselves have some structure in them that can form a hierarchy. HIBE also provides a partial solution to the escrow problem as no single server has the key for any hierarchical identity; an adversary must compromise more than one part of the hierarchy.

Additionally, systems that secret-split in a set of authorities could potentially also use this as a way to distribute the computational workload of IBE over the set. Nonetheless, performance is another consideration that IBE systems must take into account, and this one more than most, since there is no off-line generation of keys.

10 Conclusion

This presents hybrid system that combines Identity-Based features with a conventional public-key cryptosystem. Its advantages over the previous systems are that it provides interoperability with existing systems and by becoming an on-line system avoids the security problems associated with other IBE systems that permit off-line key generation. Consequently, this brings the advantages of an IBE system — that any bit string be equivalent to a public key, without the disadvantages of permitting an attacker complete knowledge of the PKG. It thus brings at this modest cost the advantages of IBE to conventional public key cryptosystems.

References

[AY03] S. Al-Riyami and K. G. Paterson, *Certificateless Public Key Cryptography*, extended abstract in Proceedings of ASIACRYPT '03, LNCS 2894, Springer-Verlag, 2003. Full paper available in the IARC eprint archives, <http://eprint.iacr.org/2003/126/>.

[BB04] D. Boneh and X. Boyen, *Secure Identity Based Encryption Without Random Oracles*, extended abstract in Proceedings of CRYPTO '04, LNCS 3152, Springer-Verlag, 2004. Full paper available in the IACR eprint archives, <http://eprint.iacr.org/2004/173/>.

[BF01] D. Boneh and M. Franklin, *Identity-Based Encryption from the Weil Pairing*, Proceedings of CRYPTO '01, LNCS 2139, pages 213-229, Springer-Verlag, 2001.

[BIHAMCHEN04] E. Biham and R. Chen, *Near-Collisions of SHA-0*, Proceedings of CRYPTO '04, LNCS 3152, Springer-Verlag, 2004. Also available in the IACR eprint archives, <http://eprint.iacr.org/2004/146/>.

[COCKS01] C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*, Proceedings of the 8th IMA International Conference on Cryptography and Coding, LNCS 2260, pages 360-363, Springer-Verlag, 2001.

[ELLISON00] C. Ellison, *Naming and Certificates*, Proceedings of the tenth conference on Computers, freedom and privacy: challenging the assumptions, ACM, <http://www.cfp2000.org/papers/ellison.pdf>.

[GENTRY03] C. Gentry, *Certificate-Based Encryption and the Certificate Revocation Problem*, Proceedings of EUROCRYPT '03, LNCS 2656, pages 272-293, Springer-Verlag 2003. Corrected version available as <http://eprint.iacr.org/2003/183/>.

[GS02] C. Gentry and A. Silverberg, *Hierarchical ID-Based Cryptography*, Proceedings of ASIACRYPT '02, LNCS 2501, pages 548-566, Springer-Verlag 2002. Also available as <http://eprint.iacr.org/2002/056/>.

[HL02] J. Horwitz and B. Lynn, *Toward Hierarchical Identity-Based Encryption*, Proceedings of EUROCRYPT '02, LNCS 2332, pages 466-481, Springer-Verlag 2002.

[JOUX04] A. Joux, *Multicollisions in Iterated Hash Functions*, Proceedings of CRYPTO '04, LNCS 3152, Springer-Verlag, 2004.

[LQ03] B. Libert and J. Quisquater, *New Identity Based Signcryption Schemes from Pairings*, IEEE Information Theory Workshop, 2003. Also available as <http://eprint.iacr.org/2003/023/>.

[LQ04] B. Libert and J. Quisquater. *What Is Possible with Identity Based Cryptography for PKIs and What Still Must Be Improved*, Proceedings of EUROPKI 2004, pages 57-70, Springer-Verlag 2004.

[MILLER56] G. A. Miller, *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*. The Psychological Review, 1956, vol. 63, pp. 81-97. Also available as <http://www.well.com/user/smalin/miller.html>.

[SHAMIR79] A. Shamir. *How to share a secret*. Communications of the ACM. Volume 22, Issue 11 (November 1979), pages 612-613.

[SHAMIR84] A. Shamir, *Identity-based Cryptosystems and Signature Schemes*. Proceedings of CRYPTO '84, LNCS 196, pages 47-53, Springer-Verlag, 1984.

[TLS] T. Dierks and C. Allen, *The TLS Protocol Version 1.0*, RFC2246 <http://www.ietf.org/rfc/rfc2246.txt>

[WANG04] Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu, *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*, IACR eprint archive. <http://eprint.iacr.org/2004/199/>.

[ZHENG97] Y. Zheng, *Digital Signcryption or to achieve cost(signature & encryption) « cost(signature)+cost(encryption)*. Proceedings of CRYPTO '97, LNCS 1294, pages 165-179, Springer-Verlag, 1997.

A Proxy-based Approach to Take Cryptography Out of The Browsers for Better Security and Usability

Marco Antônio Carnut (kiko@tempest.com.br)
Tempest Security Technologies

Evandro Curvelo Hora (evandro@tempest.com.br)
Universidade Federal de Sergipe - DCCE/CCET

ABSTRACT

This paper describes the implementation of a multiplatform selective filtering/rewriting HTTP proxy that allows the PKI-related operations – such as digital certificate issuance, web form field signing and HTTPS client authentication – to be performed entirely outside the browser, even though the browser continues to be used for what it's good at: rendering web pages. Implications such as better usability through improved user interfaces are discussed in light of a prototype implementation.

1 INTRODUCTION

The SSL/TLS protocols were originally designed to provide encrypted and authenticated channels for web servers and clients. Even today, they are almost exclusively used to authenticate servers, despite its support for client authentication. There are many reasons for that: in [4], it is shown that getting a client certificate – even a free an instantaneous one – is too much of a hassle for the average user. Internet Explorer (IE), the most popular web browser, makes it all too easy to store the certificate without a passphrase; besides, its client certificate-based logon window is confusing, showing expired and revoked certificates along with valid ones and it is outfitted with a "remember password" checkbox that causes the passphrase to be stored unencrypted, invalidating much of the security the process might provide.

The way failures are handled is also confusing: when the server can't validate the client certificate (either because it couldn't build a trusted certificate chain or the client certificate was found to be revoked), it simply breaks the connection; there are no provisions to redirect the user to a nice page explaining what went wrong and how to fix it.

All these usability problems cause enough user rejection that webmasters find it simpler to use weaker authentication schemes such as name+password+cookies. Although vulnerabilities have been discovered (and in some cases fixed) in most browser's crypto implementations, bad human-computer interface (HCI) is often appointed as a serious hinderance to PKI adoption in general [14] and client-based authentication in particular [18].

There have been a few attempts to improve the user-friendliness of client authentication, such as VeriSign's Personal Trust Agent [17] and RSADSI's Keon WebPassport [16]. However, as they are both ActiveX controls, they are Windows-only solutions and since they are activated after the SSL handshake, they have to resort to proprietary authentication schemes.

Another great promise brought by public key cryptography is the use of digital signatures as a way to detect tampering on digital documents. Some web browsers can natively sign the contents of web form fields, but many – most notably IE – do not support this feature. In IE, it can be implemented using ActiveX or even Java (although that requires installing CAPICOM, making the process less transparent), but they tend to be too cumbersome for large-scale deployment.

This paper investigates an alternative way to provide client certificate-based authentication and web form signature, along necessary subsidiary services such as digital certificate issuance, by performing all the cryptographic and user interface chores in a separate program: we use a selective cryptographic filtering/rewriting HTTP proxy to implement all the PKI-related features, leaving to the browser only what it's good at: rendering web pages. This approach has the advantage that it works with any browser that supports proxies.

Specifically, we wanted to make a general purpose utility for handling digital certificates that provided easy-to-use digital signature generation and verification functions; and that could be integrated with the web browser to allow web form signature and client certificate authentication in HTTPS with a much better user interface and security features under our control. We also wanted this utility to be a testbed for new HCI ideas applied to client-side (primarily, but not limited to, web-based) PKI applications.

This paper focuses on the cryptographic, PKI and protocol issues needed to "take crypto on our own hands" (as opposed to letting the browsers do it), while simultaneously striving to maintain backwards compatibility. Although we do make extensive use of screenshots to illustrate some features and preliminary user interface (UI) ideas we implemented – and sometimes we even indulge in describing some of its details and user feedback we received –, an analysis of the merits of our tool's UI is beyond the scope of this paper, for it requires entirely different approaches and techniques. What we want here is to show one possible way it can be done.

Besides general familiarity with the X.509/PKIX/PKCS standards and PKIs in general, this text assumes the reader has considerable familiarity with the HTTP [1] and HTTPS [2, 3] protocols.

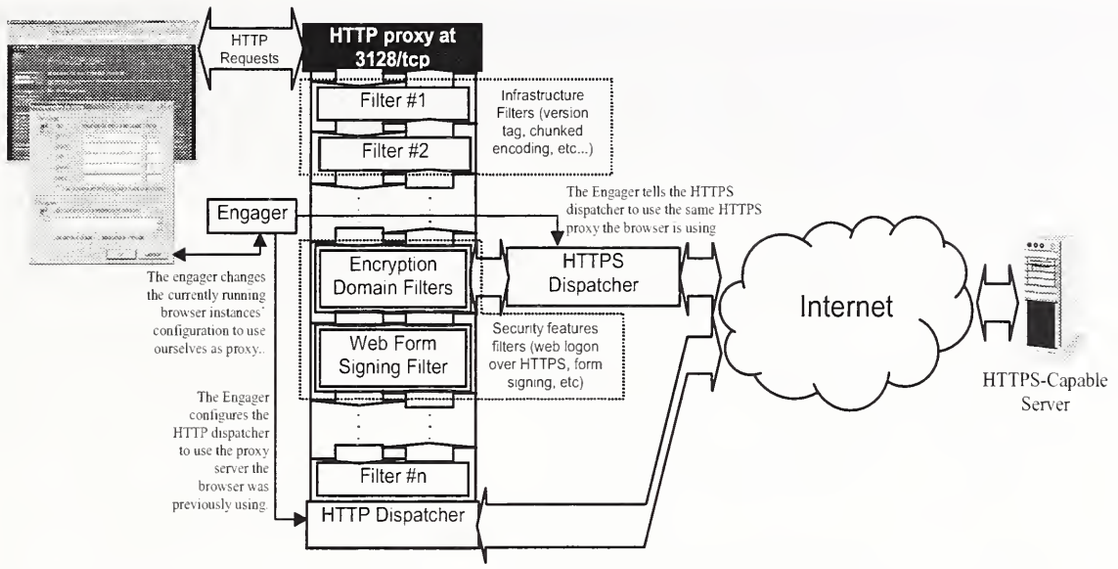


Figure 1: Overall architecture of the client proxy, which runs in the same computer as the browser. The engager changes the browser's proxy settings so that it uses our own local proxy. Before doing that, though, it detects which HTTP and HTTPS proxies the browser was using and configures our dispatchers to use them. This effectively puts us in the middle of the proxy chain. New HTTP requests originated by the browser will pass through our proxy, where our filters may act upon them. In fact, we have two filter chains, one for the outgoing requests and other for the incoming responses; some of them act upon the headers, other upon the bodies. Some features actually require several filters in cooperation to implement. If none of the filters actually consume the request (i.e., take it out of the chain), it reaches the default dispatcher in the end of the chain and it is sent as an HTTP request. The Encryption Domain filterset is a special set of filters that reroutes the requests that match certain criteria to be sent over as HTTPS. The HTTPS dispatcher makes use of the certificate store services (not shown in this picture) to validate the certificates and perform client authentication (with a friendly UI) if the site so requests.

2 OVERALL ARCHITECTURE

Our tool, code-named Kapanga, is an executable program that typically (although not necessarily) runs in the same computer as the user's web browser. A schematic depiction of its overall architecture can be seen in Figure 1. A brief description of its major components follows:

- **Certificate Store Manager (CSM):** provides all the underlying cryptographic services needed by all the other components. It manages and provides access to all the cryptographic objects (certificates, certificate revocation lists, private keys, signatures, etc) stored in various kinds of storage media (the local disk, removable storage devices, crypto-capable devices such as smart cards, etc); provides access to cryptographic algorithms and protocols. The CSM is detailed in section 2.1 .
- **Filtering HTTP Proxy Server**: receives the requests from the browser and feeds them through the filter chain. If no filters consume the request, it is passed to the HTTP dispatcher nearly unchanged. Filters may alter either the request before they're sent to the dispatcher or the replies berfore they're sent back to the browser. These changes implement the program's main features, as it will be detailed further along.
- **Engagers:** they are in charge of changing the HTTP proxy settings of all supported browsers to point to our own proxy described above, so that we get to intercept all HTTP traffic initiated by the browsers. Engagers are described in detail in section 2.3 .
- **Default Dispatcher:** an embedded HTTP user agent that sits at the end of the filter chain. It acts like a "default route" in a routing table: any requests that reach it are sent their destinations, either directly or via another next-hop proxy. It also proxies any authorization requests (e.g., Basic, Digest or NTLM authentication) that the next-hop proxy may require, so the authentication protocol is handled by the browser itself and any username+password dialog boxes that may be required is also shown by the browser itself. Upon receiving the results, it pipes them back to the response filters, which also play crucial security roles.
- **HTTPS dispatcher and the Encryption Domain:** similar to the default dispatcher, but tunneling the requests over TLS/SSL [2]. For performance reasons, it features support for rehandshakes and session caching [3]. It relies heavily on CSM services for validating the servers' certificates and providing client authentication if the server so requires. A request is sent through this dispatcher if the `host:port` of the request is listed in a set called *Encryption Domain* (this detour is actually accomplished by a special filterset collectively known as the

"Encryption Domain filters"). As with the default dispatcher, it may either send the request directly or use the `CONNECT` method to tunnel it over a next-hop proxy [5] if the engager has previously told it so. It is also responsible for sending back any authentication requests that the next-hop proxy may require.

2.1 Certificate Store Manager

Underlying the whole program is the Certificate Store Manager, providing crypto and PKI services to the other subsystems:

- **Certificate, CRL and private key enumeration and caching**: all those objects can live in one or more physical media. The local hard disk is called the primary store location, bringing a minimal set of certificates right from the installation procedure.

  The user may configure one or more secondary locations. Those are usually removable media, such as CD-ROMs, diskettes or USB flash memory devices ("pen drives"). Every three seconds or so the certificate store manager checks to see if these devices are readable and, if so, rescans them. This way, a user may have and use his/her certificates/keys in a removable storage medium during their entire lifetime[1].

  Crypto devices such as smartcards are also supported, although they are handled as special cases because some objects (private keys, primarily) may not be exported and we may only operate on them via the device's built-in cryptographic capabilities.

  The resulting in-memory cache can be seen as a concatenation of all the contents of all of the devices. CRLs are handled as a special case – since some of them tend to get very big, they are deallocated from memory as soon as the CSM is done using them in the trust calculations.

  Private keys are handled as special cases as well. When stored in crypto devices, the CSM directs all its crypto primitives to the device's drivers to make use of its embedded functionality; otherwise, they are loaded only when needed and the crypto primitives (signing/decryption) are directed to the software-based implementation.

  Our certificate store has another type of object called *attestation signature* or simply *attestation*. It is a signature block on someone else's certificate made by the private key of a user to indicate that it trusts that certificate (typically a root CA). This signature is detached – that is, it is stored in a separate file in a file format of our own devising; we will have more to say about attestations in section 2.1.1. .

- **Chaining:** after the certificates are loaded from the physical stores, the CSM tries to chain them. First, duplicates are discarded and certificates issued by the same CA are sorted by their `notBefore` fields and assembled as a doubly-linked list. The *best current* certificate is selected by applying two criteria: a) it is the one with the most recent `notBefore` and b) it must be still within validity (that is, with the current date/time before its `notAfter` field). If no certificate satisfies both requirements, we settle for the one that satisfies only (a).

  After that we build several indices for fast lookup: one keyed by the certificate's SHA1 hash, other by its Subject Key Identifier extension [7] and another by subject DN. This last one has a peculiarity: only the best current certificates make to this index; the future and previous editions don't get there.

  We then chain the certificates in the usual way, using the recently computed indexes to speed up finding the issuer of each certificate in the store (matching SKI/AKI pairs, when available, and by subject/issuer DNs as a last resort). We set the parent pointer of each certificate to the issuer and record its the depth in the tree (the whole chaining algorithm uses a breadth-first search precisely to make that trivial).

  CRLs are considered as an appendage to their issuer certificates and are chained to them. Private keys are also appendages and are linked to the certificates with the corresponding public key (the private key format stores the public key as well, so this comparison is straightforward).

- **Trust Status Calculations:** With all the certificates and associated objects properly chained, we start to verify their validitity periods, signatures of its issuers, attestations, etc. It is interesting to notice that all trust calculations are relative to the currently selected default ID, since attestations depend on it. Thus, whenever the user changes the default ID via the GUI, the whole trust statuses are recomputed. Section 2.1.2. describes each trust status in detail.

The CSM has a few other utilities and services:

- **Public CSM Server:** We coded a version of the CSM in a web server that is offered as an associated on-line service to the user and acts a public certificate/CRL repository. Over the years,

[1] Some of our users like to call this "the poor man's smartcard". We try to tell them this is a particularly nasty misnomer – not only because certain media such as USB "pen drives" are actually *more expensive* than smartcards (even including the cost of the reader), but also because they lack the tamper-proofness and crypto capabilities of the latter.

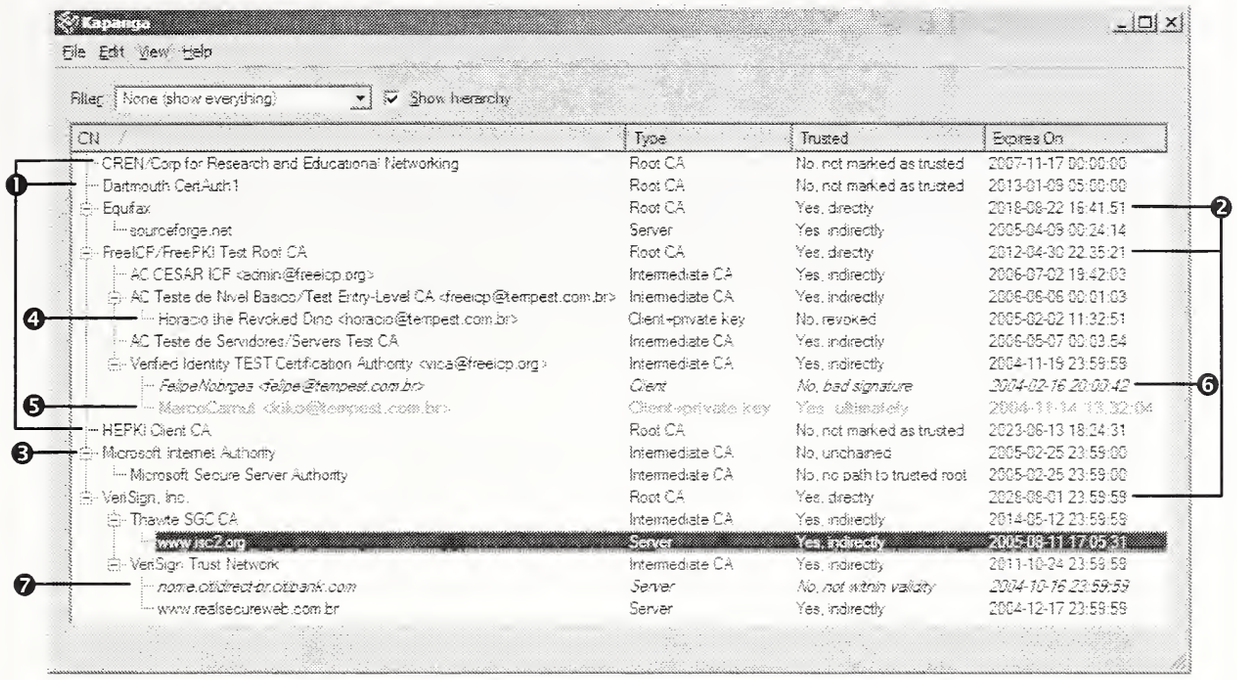


Figure 2: The CSM trust calculation results as displayed in the GUI. Here we have a certificate store with (1) three unattested roots and (2) three attested, trusted roots. There is also an Intermediate CA (3) that cannot be trusted because it's unchained, meaning that we lack its issuer root. All children of trustred roots are marked as indirectly trusted unless they've been revoked (4), their signatures don't match (6), or it's not within its validity period (7). The item marked in (5) is the current ID (aking to PGP's "default key"). All trust calculations are relative to the attestations (detached signatures on root CAs) this ID has previously performed.

we've been dumping on this server every CA certificate we lay our hands on. Whenever a certificate is unchained, the Kapanga may query the CSM server (either automatically due a configuration option or manually through a pop-up menu in the GUI) to see if it knows the missing issuer. The external CSM may also return CRLs – it has a built in robot that tries to fetch CRLs of all CAs it knows about from its distribution points.

- **Automatic CRL Download:** Just like the public online CSM server, the program's internal CSM has a similar feature – it automatically tries to download the latest CRLs from the addresses advertised in each CA certificate's `cRLDistributionPoints` extension. It can do so upon user request or automatically in the background. In the automatic mode, the list of candidates URLs is rebuilt each four hours (configurable) and we try to downloads CRLs from them. In case of success, the whole trust settings are recomputed and redisplayed. If some download fails, the next attempt time is subject to an exponential backoff algorithm with a maximum period of one week.

The overall effect we tried to achive is that the user doesn't have to worry about the intricacies of certificate management at all: he/she would only use the program features, collecting certificates along the way, and the CSM will do its best to ascertain its trust statuses and keep everything updated – without removing from the user the possibility of doing things manually if he/she so wishes.

2.1.1. Attestations

Attestations are signatures of a private key in someone else's certificates as a means of informing Kapanga that the owner of that private key trusts the signed certificate. They are akin to PGP's key signatures or introductions, except that they are stored in a file separate from the certificate itself.

We originally implemented attestations only for Root CAs as a more secure means to tell the CSM that particular CA is to be considered trusted. We were trying to avoid a simple vulnerability most browsers have: it's quite easy write a malicious executable that inserts a new fake root CA in their trusted certstore – in IE's case, it can be done in a few lines of code, since the root CAs are stored in the registry; for Mozilla-derived browser's, it requires only slightly more effort, since the root CAs are in a Berkeley-DB file.

As we will see in the next section, Kapanga trusts root CAs only if they're signed by the user's key. We say that the only truly trusted certificate is the user's own, because he/she has the corresponding private key. All the trust placed in the other certificates, even root certificates, stems from the user. Hopefully it also makes the root insertion attack slightly harder, for it will require the attacker to induce the user to sign the corresponding certificates.

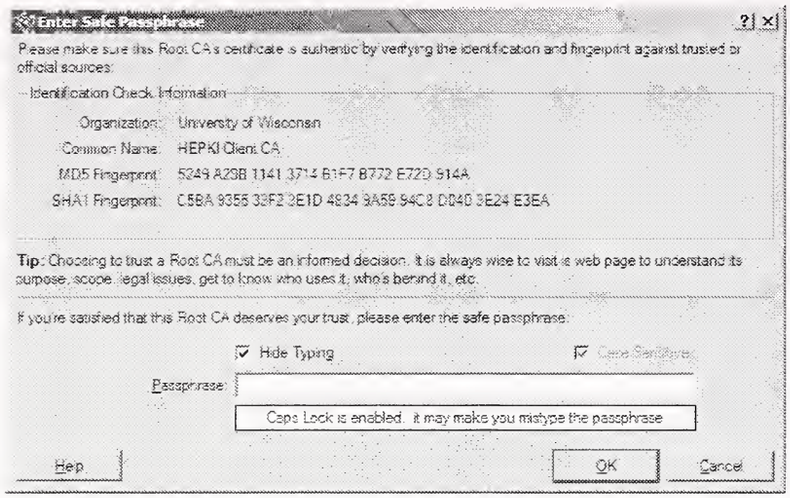


Figure 3: Manual Attestation Process. The user must sign the Root CA's certificate with his/her private key. Only then this CA will be considered directly trusted. The UI was designed as a single-step dialog box presenting the most important certificate identifiers for manual checking against other trusted sources, instead of the unecessarily complex and sometimes scary multi-step wizards most browsers have. The text succintly explains that this must be an informed decision. In this screenshot, we see the program requesting the private key's passphrase, which reinforces the sense of importance of this action. An always-enabled but impossible to disable check box reminds the user that passphrases are case-sensitive. Root CA attestations are also integrated with the certificate issuance/import dialogs, so users rarely come to this dialog to perform root attestations; it is more often used to attest certificates other than roots.

While this does improve security, this may seem as an extra complication: we *require* the user to have a certificate and private key; Kapanga is nearly useless if the user doesn't, since it will trust no one. After getting a certificate, the user would need to perform a few attestations. We tried to make this simple by integrating the attestation process with the certificate issuance/import processes: as shown in Figure 4, when the user gets a new certificate, a few checkboxes cause its root to be automatically attested, as well as all other roots this root trusts: root attestations on other roots are our bridge-CA mechanism.

Later on, we generalized the attestation system: the user now can sign any certificate he/she chooses.

This effectively makes Kapanga's trust system a cross-breed between the X.509's strictly hierarchical and PGP's web-of-trust models. While we were inspired by and tried to follow RFC 3280's certificate validation rules, we can't really say we follow them to the letter because it ended up evolving in a different direction.

Other interesting analogies can be drawn with other public-key based systems: for instance, signing an unchained server certificate in Kapanga is akin to adding a SSH server key to the `~/.ssh/known_hosts` file, except it's harder to spoof because of the signature.

2.1.2. Trust Statuses

The trust calculations assign one of the following statues for each certificate:

- **Ultimately Trusted:** this means that we have the private key corresponding to this certificate. Thus, it is an identity we can assume. Those certificates are considered to be "the root above the Roots", the true starting point all trust stems from. As a result, such certificates are considered trusted even if they're not properly chained or if its chain doesn't go all the way up to a trusted root; we say this status *overrides* the "Unchained" and "No path to trusted root" statuses described below.
- **Directly Trusted:** this means that this certificate has been *attested* by the current user. In other words, there is a signature block on this certificate correctly verified against the current user's public key as proof that the user gave his/her direct consent that this certificate must be considered trusted. If this a CA certificate, this causes all child certificates to be considered indirectly trusted.
- **Indirectly trusted:** this means that the CSM has verified that the signature of the issuer is valid and that the issuer is trusted (either directly or indirectly).
- **Not Within Validity:** the certificate is not trusted because the current date and time is after the value specified in the certificate's `notAfter` field or before the value specified in the `notBefore` field. This status overrides all others (even the Ultimately Trusted stauts): the CSM doesn't even bother checking anything else.
- **Unchained:** the certificate cannot be considered as trusted because it we don't have its issuer. This status applies only to intermediate CAs and end-entities; it obviously can't happen in Root CAs. This status can override all the previous ones except the "Ultimately Trusted".
- **Not Attested:** this only happens to Root CAs. The certificate cannot be considered as trusted because we either have no valid attestation signature on this root from the current user's.
- **No path to trusted root:** the certificate cannot be considered as trusted either because the root of the chain has not been attested (it is not directly trusted) or some of its issuers are unchained (the chain doesn't go all the way up to a root CA).

- **Revoked:** the certificate is not trusted because its serial number is listed in its CA's Certificate Revocation List (CRL) and the CRL itself is valid.

The trust statuses for CRLs work a bit differently. A CRL is considered valid if the signature of its CA matches just fine, regardless of whether it is outdated or not. If a given certificate is listed in some CRL, it is flagged as revoked even if the CRL is not the freshest possible; the CRL checking engine tries to do the best with what it has. It is the responsibility of the automatic CRL dowload feature to try to keep CRLs as up-to-date as possible.

When the CRL checking engine is asked about whether a certificate is revoked or not, it returns an answer consisting of three items:

- **Is_revoked**: a tristate flag saying whether the certificate is revoked (true), or not (false) or if we can't ascertain because we have no CRL for this CA (unknown). If this flag is unknown, the remaining two items describe below are undefined.
- **Is_outdated:** it says whether the CRL used to compute the is_revoked status is outdated or not.
- **Reference date:** if is_revoked is true, it returns the revocation time and date as taken from the CRL. Otherwise, it returns the CRL's `lastUpdate` field, meaning that we can be certain that this certificate isn't revoked only up to the moment the CRL was issued.

2.1.3. Certificate Issuance, Import and Export

Another important service provided by the CSM is providing support for having new certificates issued through a Certificate Authority. From the point of view of the CSM itself, its just a matter of having an RSA keypair generated and converting it to an Netscape SPKAC (Signed Public Key and Challenge, see [12]) format (a Certificate Signing Request would seem a better choice, but the reason for that will become clear further along).

From the point of view of the user interface, there are two very different implementations:

- The classic web-based style, in which the user directs his/her browser to the CA web page, fills some web forms and the browser activates the key generation procedure. Since this issuance system is intrisically intertwined with the filter system, it will be described along with our discussion of the HTTP filters in section 2.2 .
- We also wanted to have a PGP-like wizard-based instantaneous key generation. To that end, we implemented a specialized wizard that uses FreeICP.ORG's Entry-Level CA [4] to allow the user to get a free, instantaneous short-lived certificate. The rest of this subsection describes some particularities of this process.

In the first step, the user enters his name and email address, being also warned that the process requires being online or else the process will fail – this is unlike PGP. The user is also asked to reconfigure his/her spam filters to prevent the CA notification messages from being blocked.

After that, the wizard asks the CA whether the email address the user requested is already taken – that is, whether the CA has in its database a valid certificate issued for that email address. This is implemented by sending the CA a request for a "Revocation Reminder". If the server responds with a "No valid certificate associated with this email address" message, we let the user proceed. Otherwise we inform that the user is going to receive an email with revocation instructions and ask him/her to follow it before coming back to try to issue the certificate again. In this situation, the "Next" button of the wizard is grayed out, making impossible to proceed.

The next step is setting up the passphrase – historically the step users hate the most. This is constitutes a good opportunity to describe what kind of usability ideas we've been experimenting with, so we will detour from the "protocol nuts and bolts" approach we've been adopting so far and make an aside about our UI designs.

The philosophy is to try to steer the user to do the right thing, both through education and trying to prevent unwittingly dangerous actions. However, it can't be frustrating either, so the restrictions must not be absolute; they have to be bypassable, although the user must feel frowned upon when choosing the insecure path.

As usual, we have two passphrase text entry boxes. By defaut, they are set not to show the text being typed, replacing the characters by asterisks. Just like in PGP, however this is bypassable by unchecking a "Hide Typing" checkbox. This is needed because some poor typist users take too many attempts to make the content of the text boxes match that they become frustrated and quit. But unlike in PGP, if they opt to do this, they get a insistent blinking message warning them to make sure they aren't being watched or filmed.

We also implemented a warning about Caps Lock being enabled, now common in many programs.

Also common is the passphrase quality meter. The metering algorithm tries to estimate the entropy in the password roughly by making a weighted average of two criteria: the word entropy and the character entropy. The former is exceedingly simple-minded: we assume that each word adds about 11 bits of entropy. The latter is more complicated: we determine the bounding set of the characters of the passphrase in the

ASCII code space and use it as an entropy per character estimator. Then we multiply it by the number of characters and divide it by the efficiency of a customized run-length encoder. This has the effect of yielding very low scores to regular sequences such as "aaaa" and "12345". The quality meter displays its score in a progress bar and with a scale categorizing them as "simple", "good" and "complex".

The reason we didn't bother to be much more scientific than that with the quality meter is that in our early attempts it became clear it would result in it being overly frustrating to the end users. Our priority is to keep the users happy (or at least not too unhappy), so we calibrated (or rather downgraded) the algorithm many times to quell their complaints. We did perform some research about it, but in the limited time we had we could find no real good papers with general design guidelines for passphrase quality meters. We opted for trial and error based on the users' feedback.

In the end, we struck a middle ground with the following strategy: we made the meter slightly challenging and by default it doesn't allow you to go on if the score doesn't lie in the "good" range. However, we added a checkbox that allows you to disable the meter restriction altogether – in which case the user gets a polite message telling something like "now you're on your own risk, hope you know what you're doing – don't say I didn't warn you".

A frequent question our users pose is "what's a good passphrase anyway?" To try to answer that we implemented the passphrase suggestion dialog shown in Figure 4d. It generates passphrases suggestions using the Diceware method [18], which consists of generating a random number and mapping them onto a dictionary of 7776 words and common abbreviations, yielding 12.92 bits of entropy per word. (The method was originally designed to be performed by hand, pencil and paper using five dice tosses to select each word.) With 5 words we get more than 64 bits of entropy, which provides good enough protection against brute force attacks under quite general conditions while remaning reasonably easy to memorize.

Our user's feedback to the passphrase suggestion box has been a mixed bag. Some love it and many hate it – the primary complaint is that passphrases are way long. Many system administrators have been asking us to add a passphrase suggestion algorithm that matches their password policies like "8 characters with at least one punctuation character and a number not in the end nor the beginning". No amount of arguing that the diceware passphrases are more secure than those approaches seems to convince them. On the good side, however, our rejection rate has been zero precisely because we give the users the choice: they can simply disable the quality meter and ignore the suggestion box altogether if they really want to. Over time, we see that users gradually start to explore the passphrase suggestion box and the number of good passphrases slowly increases. A quantitative characterization of those intuitive perceptions may make fertile ground for a future paper.

The last page of the wizard is the one where the key pair is generated. As many other implementations do, a progress bar tracks the possibly lengthy key generation process; we were working on an educational animation to add to this window, but a discussion of its features is beyond of the scope of this paper.

After the key pair is generated, the private key is encrypted with the passphrase and saved in the Certificate Store. The public key is converted to the SPKAC format. When the wizard is invoked from the Keygen Interceptor filter (see section 2.2.2.), we return this SPKAC to the filter. We also store along with the private key the state of the attestation checkboxes in the final page of the wizard (the CSM has facilities to add `property=value` tags along with any object) – they will be needed later when it's time to pick up the issued certificate and insert it in the CSM.

If, on the other hand, wizard has been invoked from the main menu, the SKPAC is sent in a HTTPS message to the FreeICP.ORG Entry-Level CA (the destination URL is configurable but with a hardcoded default). The Entry-Level CA responds immediately with the certificate in a PKCS#7 bag right in the HTTP response body.

2.2 Filters

Filters are routines that change the request header, the request body, the response header or the response body of the HTTP requests received by our internal HTTP server. In our implementation, each filter can change only one of these items; the cooperation of several filters is often needed to implement a single particular feature. The filters are organized in two filter chains: the request chain and the response chain. Within a chain, the filters are executed sequentially in the order they've been set up. Some filters depend on others, so the chain setup tries to ensure that they are topologically sorted.

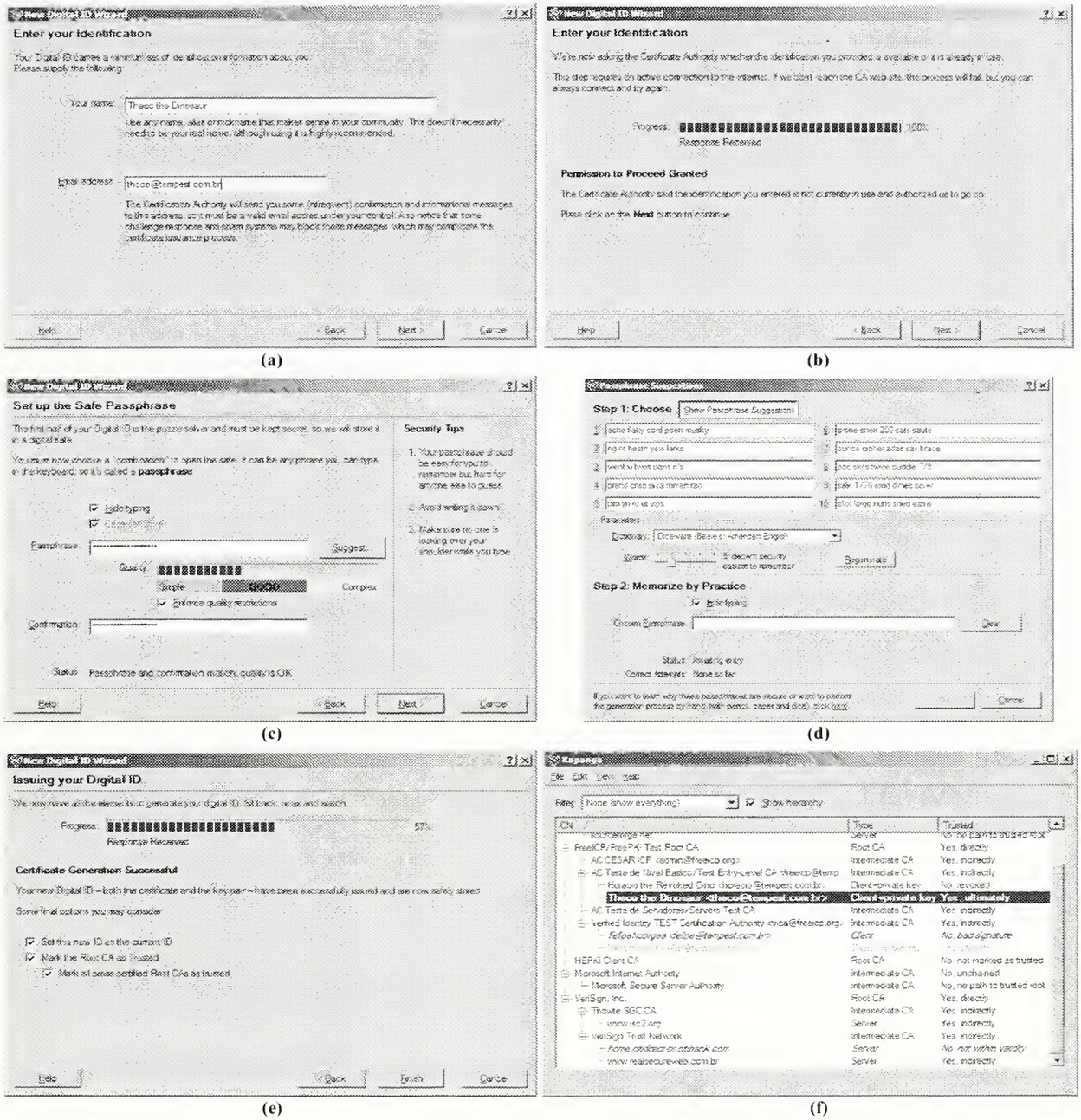


Figure 4: Wizard-style UI for using the FreeICP.ORG instantaneous Entry-Level certificate issuance process. In (a) the user enters his/her name and email address, while being advised the need to be online and that notifications will be sent over email. In (b) the CA is queried to see whether that email address is already in use. If so, the CA will send an email with revocation instructions and the process is halted. In (c) the user sets up a passphrase for the private key that is about to be generated. A quality meter gives prevents the user from choosing too weak a passphrase – unless the "Enforce quality restrictions" checkbox is disabled. The status texts indicate in real time when the confirmation matches and some educational security tips are also offered. In (d) we see the passphrase suggestions dialog: ten suggestions are put forth so that the user can choose visually without revealing the passphrase to shoulder surfers. As the first character is typed, all fields turn to asterisks. Each time the user correctly retypes the passphrase causes the chosen box to blink. Typing a different one resets the counter. Cheating by using copy-and-paste works but the user is politely warned that this doesn't help memorization. In (e), the key pair has been generated, converted to SPKAC, sent to the CA and the signed certificate has been received back. In (f) we see the new certificate and its associated private key in the certstore main window, already set as the default ID. The "Mark the Root CA as Trusted" checkbox caused the attestation of root certificate, so it's shown as directly trusted; the "Mark all cross-certified Root CAs as Trusted" checkbox caused an attestation on VeriSign's Root CA as well. The whole process takes 20 seconds or so for experienced users and less than two minutes for novices – most of it spent figuring out how to either please or bypass the quality meter. The user gets out of the process with all attestations already performed, so he/she will rarely have to perform manual attestations.

Notice that request filters can *consume* the HTTP request entirely, removing it from the chain so that it won't reach neither the subsequent filters nor the default dispatcher at the end of the chain. It then becomes this filter's responsibility to either issue the

request and insert the response back in the chain or to abort the request entirely.

Filters can be divided in two main groups described in the following subsections.

2.2.1. Infrastructure Filters

Infrastructure filters aren't directly involved in implementing the security features; they primarily provide services for the other filters. A description of the most important filters in this category follows, roughly in order from the simplest to the most complex:

- **Command Parser:** this is a simple request header filter that detects and extracts a special query string on the form "`x-kapanga-cmd=[command]`" from the URL. Below we have a short summary of the commands; each will be discussed in detail further along:

  `http://example.com/?x-kapanga-cmd=addsite(port,title,errpath)`
  Adds the current site (example.com:80) to the encryption domain. TLS/SSL connections will be sent to the TCP port specified in "port". If the certificate validation fails, the request is redirected to "errpath". The parameter "title" is a user-friendly added to the bookmark/favorites lists.

  `http://test.com:8080/?x-kapanga-cmd=delsite`
  Removes the current site (test.com:8080) from the encryption domain.

  `http://yasite.com/?x-kapanga-cmd=sign(data,sig)`
  Prepares to sign the field named "data" in an web form that will be downloaded as a result of this request. The signature will be performed when the user hits the submit button in his/her web browser and the result will be placed in a (possible new) form field named "sig" as a S/MIME signature.

  `http://somewhere.net/?x-kapanga-cmd=send-usable-ids`
  Forces the request to become a POST and sends a list of valid ultimately trusted certificates (without their respective private keys, of course).

  `http://whatever.org.ar/?x-kapanga-cmd=activate(sha1)`
  Sets the ultimately trusted certificate with fingerprint SHA1 as the default for client authentication with the server specified in the URL (in the example, "whatever.org.ar:80")

  `http://dummy.net/?x-kapanga-cmd=ua(string)`
  This command interacts with two filters. First, it tells the Version Tag filter to change the User-Agent header to *string*, effectively lying about the browser's identity and version. This will be needed to redirect us to the Netscape-style certificate issuance system in commercial web-based CAs. Second, it arms the Keygen interceptor filter.

- **Version Tag:** A simple request header filter that appends an identifier and our version numer to the `User-Agent` header, without removing the browser's identification. This allows the web server to detect whether our tool is enabled and perhaps offer customized functionality. For instance, a client authentication-capable website could detect that Kapanga is engaged to the browser and offer its login URL already including the `x-kapanga-cmd=addsite` command.

  This filter is also responsible for "lying" about the browser's identity when the command parser has previously received the `x-kapanga-cmd=ua(string)` command. It changes all User-Agent request headers to the specified string (typlically something like "Mozilla/5.0"). It also replaces all occurences of `navigator.appVersion` in JavaScripts by the specified string, since most web-based commercial CA software uses embedded scripts to determine the browser's version.

- **Encoding Dampers:** quells any encoding negotiation we can't understand, such as gzip or deflate encodings. In our current implementation, we don't support any encodings, so this is a simple filter that sets the the `Accept-Encoding` field of the HTTP request headers for the identity transformation. This is needed because several filters down the chain will need to parse the HTML when it comes back. This, of course, hinders any performance gains that those encodings might bring. Future implementations will replace the damper by a proper set of decoders.

- **Chunked Transfer Encoder:** converts the HTTP response bodies to the chunked transfer encoded form (see [1], section 3.6). This is needed because the response body filters will very likely change the length of the body, so the browser must not employ the ordinary strategy of relying on the `Content-Length` header. All that, in turn, is a consequence of the fact that the body filters perform on-the-fly rewriting, that is, they act upon each data block read from the network. The alternative would be to buffer the whole body, compute its new length after all filters had been applied and then send it along to the browser – a bad idea because response bodies can grow arbitrarily large, often several megabytes long, which would make latency too high and memory consumption prohibitive. The Chunked Transfer

Encoding scheme was invented precisely for this kind of situation when we don't know beforehand the size of the HTTP object we're transmitting.

An example may clarify what those filters accomplish. Suppose our browser issues the following HTTP request (indented for better readability):

```
GET http://testserver.example.com/t1.html?x-
kapanga-cmd=delsite HTTP/1.1
Accept: image/gif, image/x-xbitmap,
        image/jpeg, image/pjpeg,
        application/vnd.ms-excel,
        application/vnd.ms-powerpoint,
        application/msword,
        application/x-shockwave-flash, */*
Accept-Language: pt-br
User-Agent: Mozilla/4.0 (compatible; MSIE
            6.0; Windows NT 5.1)
Host: testserver.example.com
Connection: Keep-Alive
```

The full URL on the GET request gives away the fact that our browser was configured to use a proxy. This request also includes a Kapanga-specific command. After passing through the infrastructure filters, it would be sent over the network like this:

```
GET /t1.html HTTP/1.1
accept: image/gif, image/x-xbitmap,
        image/jpeg, image/pjpeg,
        application/vnd.ms-excel,
        application/vnd.ms-powerpoint,
        application/msword,
        application/x-shockwave-flash, */*
accept-language: pt-br
accept-encoding: identity;q=1, *;q=0
connection: keep-alive
host: testserver.example.com
proxy-connection: Keep-Alive
user-agent: Mozilla/4.0 (compatible; MSIE
            6.0; Windows NT 5.1) + Kapanga
            0.22
```

Since in this example Kapanga was not configured to relay the request to another proxy (that is, IE was not using a proxy before the engager did its job), the URL in the GET line is relative. Also notice that the command parser removed the "x-kapanga-cmd".

The encoding damper has also left its mark in the `Accept-Encoding` line telling that only the identity encoding is acceptable and all others are not. We can also see that the version tag filter added our name and version number to the `User-Agent` line.

After the request is issued over the network, the server responds with something like this:

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 132

<HMTL>
 <HEAD>
  <TITLE>
   Infrastructure filters demo
  </TITLE>
 </HEAD>
 <BODY>
  <H1>Test!</H1>
  All's well.
 </BODY>
</HTML>
```

The chunked encoder converts this to:

```
HTTP/1.1 200 OK
content-type: text/html
transfer-encoding: chunked

40
<HMTL>
 <HEAD>
  <TITLE>
   Infrastructure filters demo
  </TIT
40
LE>
 </HEAD>
 <BODY>
  <H1>Test!</H1>
  All's well.
 </BODY>
</
5
HTML>

0
```

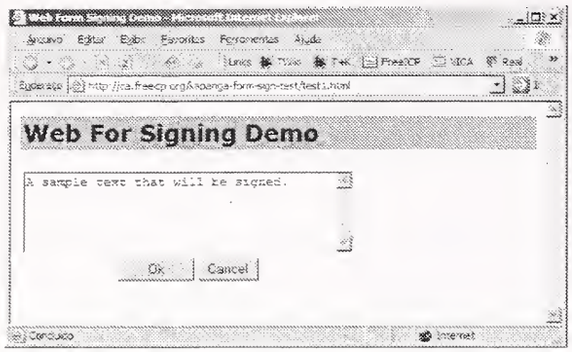
In this example, we lowered the maximum chunk size to 64 bytes to accentuate the encoding result; in our actual implementation, the maximum chunk size is 32Kb and it almost never gets that big because the networking layer sends it to us in even smaller chunks due to the underlying TCP buffers.

The chunk encoder filter has some heuristics to detect old browsers (such as IE3) that don't support the chunked transfer encoding. In those cases, it refrains from altering the body but it also quells the HTTP keepalive feature, so that the browser will rely on the connection termination to know when the body data finishes.

2.2.2. Feature Filters

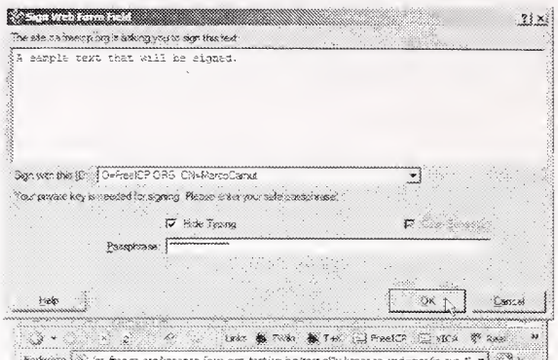
These are the filters that actually implement the security-relevant features, relying in the infrastructure provided by the previous filters and the CSM:

- **Web Form Signer:** this is a request body filter that acts only on POST requests with the "application/x-www-form-urlencoded" MIME type. It is activated when the command parser previously received a command of the form `sign(in,out,flags)`. The argument "in" is the name of a form field in the current page form which the filter will get data for signing. The filter displays a dialog box confirming the data being signed and requesting the passphrase for the private key that will be used for signing. When it receives these data from the user, it creates a S/MIME signed message and encodes as a (possibly new) form field named "out" (if "out" is ommited, it is assumed to be the same as "in"). The flags control things like whether we want our own certificate included in the signature, whether

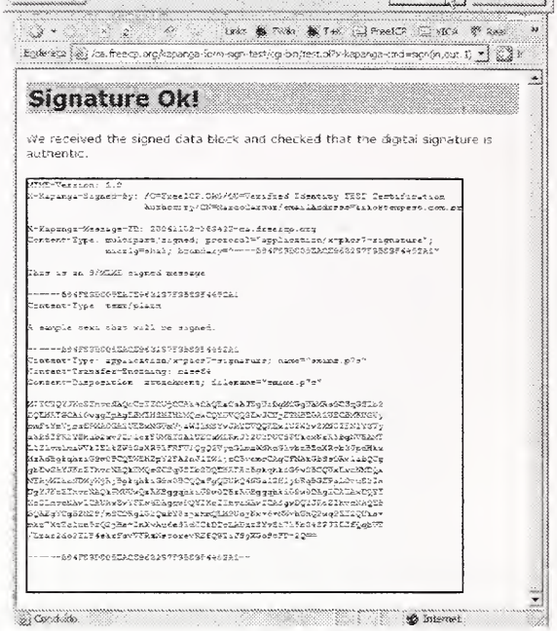


(a)

```
<HTML>
 <HEAD>
  <TITLE>Web Form Signing Demo</TITLE>
 </HEAD>
 <BODY>
  <table width=100% bgcolor=#00DDFF>
   <tr>
    <td>
      <font face="Verdana" size=+2><b>
         Web For Signing Demo
      </b></font>
    </td>
   </tr>
  </table>
  <form method=post
  action="test.pl?x-kapanga-cmd=sign(in,out,1)">
  <table border=0><tr><td>
<textarea rows=5 cols=40 name="in">
A sample text that will be signed.
</textarea>
</td></tr><tr><td align=center>
  <input type=submit name="submit" value="  Ok  ">
  <input type=reset  name="cancel" value="Cancel">
</td></tr></table>
  </form>
 </BODY>
</HTML>
```



(b)



(c)

Figure 5: The web form signing filter in action. In (a) we see a minimalistic web in the browser and its source HTML. Notice the action URL with a Kapanga special command. The command parser field intercepts this command and set things up to intercept the POST request body and sign the "in" field, putting the result in a new form field named "out". The final one in the sign command is a flag to hasve the S/MIME signer not include the signer's certificate, just to keep the signature block small enough to fit this screenshot. In (b), the exact intercepted data that will be sign is shown in a dialog box, where the program allows the user to specify which key he/she wants to sign with and asks for the key's passphrase. In (c), the signature has been performed and sent to the server. A script in there displays the signed block for us. For sake of brevity, we have shown only the successfull case. Lots of failure conditions are handled as well – for instance, when the signature doesn't match, or the signed data has been changed by the client, when the user cancels without signing or when the proxy isn't activated.

to add the whole certificate chain up to the root, etc.

The advantage of this approach is that we can add form signing functionality to some web application just by activating Kapanga and making just minor changes in the web application – if it doesn't bother to verify the signature, it's just a matter of chaning the HTML to include the `sign` command and storing the "out" field somewhere. A signature verification engine, however, would be recommended to deal with exceptions such as invalid signatures or to ensure that the signed contents is the same as previously sent (since it's within the client's control, a malicious user may change it).

- **Usable ID enumeration:** This filter is triggered by the "send-usable-ids" command. First, it forces the request to become a POST (even if the browser has sent it as a GET or HEAD). Kapanga then builds a body with a list of PEM-encoded ultimately trusted certificates it has. This is extremely useful because the site can know in advance which identities we can assume, inform the user which ones are acceptable or not and help the user select an appropriate one for login or registration, reducing the likelihood of frustrating failures.

  The webmasters we have been working with point this particular feature as the one that mostly contributes for the overall user acceptance – it makes it viable to make helpful web-based certificate enrollment/registration system almost as simple as traditional name+password+cookie methods, as shown in Figure 7.

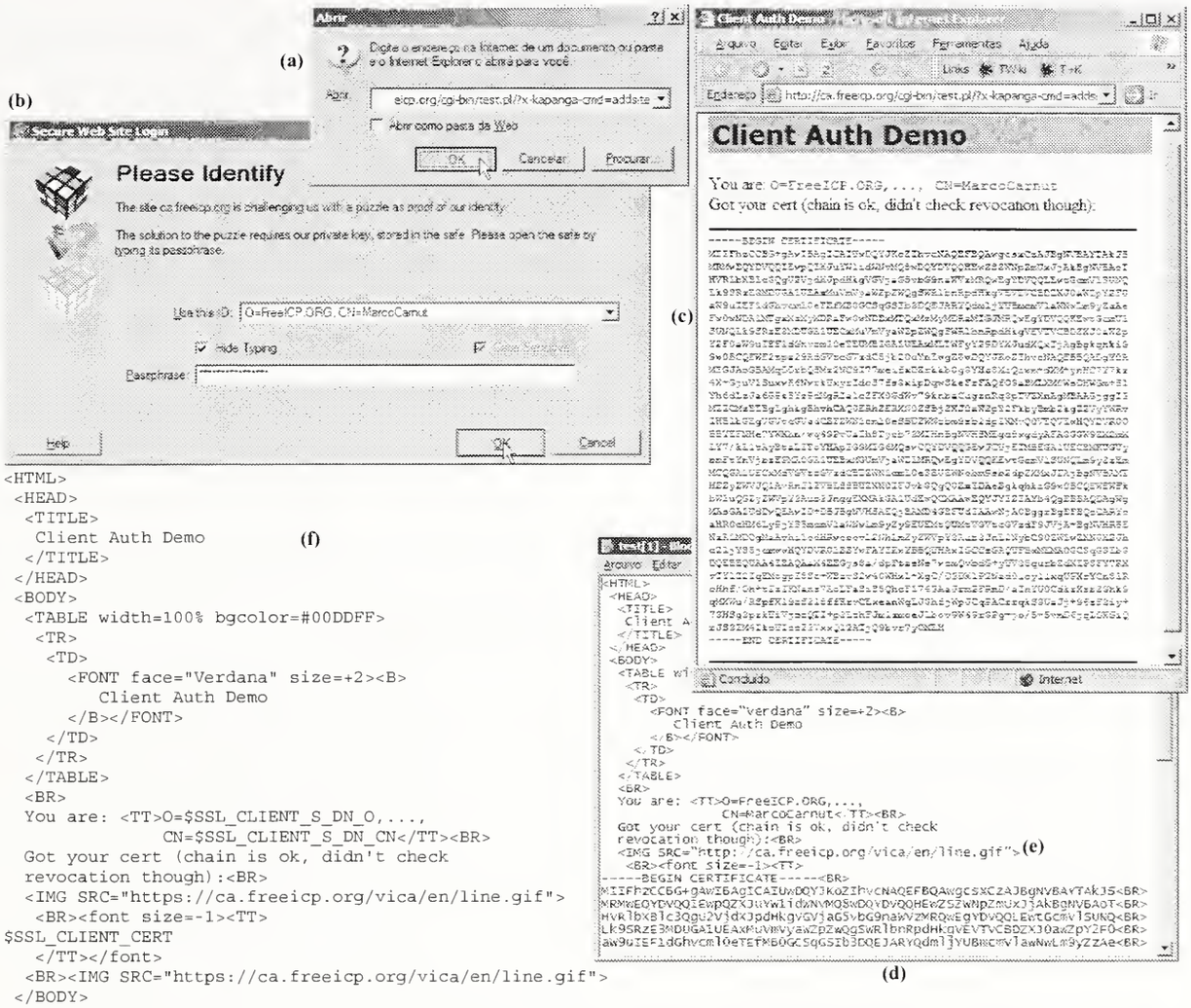


Figure 6: The HTTPS Logon filterset in a client authentication scenario. In (a) the user directs the web browser to an HTTP URL containing the command for adding the site to the Encryption Domain. As Kapanga was engaged to the browser, the request is actually sent over HTTPS because the command parser filter is executed early in the filter chain. Thus, when the request reaches the HTTPS Logon filter, the site address and port is already in the Encryption Domain. In (b), the site has requested client authentication and Kapanga asks the user which certificate he/she wants to use and the passphrase of its associated private key. Unlike Internet Explorer, Kapanga doesn't show expired, revoked or altogether untrusted certificate, nor has a "remember password" checbox to ruin the whole security of the process. In (c), and the server have sucessfully completed the TLS handshake, sent the request and got the response back, where we see that the server sucessfully received and validated the user's certificate. In (d) we see the returned page source HTML; comparing with the source HTML template in (f), we can see that the absolute URL in (e) was rewritten (notice the change from "https" to "http") so that the image download would pass through our proxy as well.

Granted, this kind of enumeration may be abused by rogue sites to collect email addresses or tracking the user's habits. We argue this is a necessary evil to provide a seamless HTTP ⇔ HTTPS transition. Just in case, we left a configuration option that allows the user to either disable this feature entirely or get a popup then the site sends the enumeration command.

- **Remote ID activation:** this filter is trigged by the "activate(sha1)" command. It sets the preferred default ID for this site (as identified by the host portion of the URL) as the certificate with the specified SHA1 fingerprint. If we have no such certificate or if it's not ultimately trusted, no action is performed.

  This command is typically used in pre-logon page just before the "addsite" command to have the correct ID selected by default in the Web Site Login Dialog (where the user is prompted for the passphrase).

- **HTTPS Logon:** this filter is activated by the "addsite" command previously seen by the Command Parser filter. Recall that this command has three parameters: the SSL port (443 by default), a user-friendly site title/name and the error redirect URL.

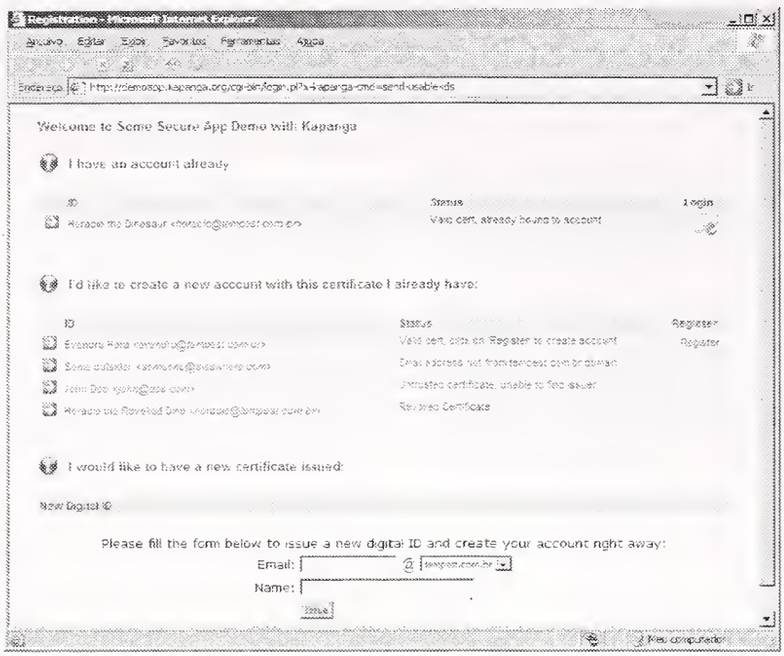


Figure 7: The `send-usable-ids` command allows the web application to provide friendly account creation assistance, explaining beforehand which certificates are acceptable and which are not and thus minimizing frustrating failures. The "login" and "register" links, use the `activate` command to force that particular certificate to be selected, minimizing user errors and then redirects to another URL with the `addsite` command, inserting the site into the Encryption Domain and starting the transition to the HTTPS site. Even so, the SSL handshake might still fail if the site's certificate doesn't pass Kapanga's validation. In this case (not shown in the picture) the "errpath" parameter in the `addsite` command would redirect the user back to a page explaining what went wrong and offering further help. At the bottom of the page, a form allows the user to start the wizard-based certificate issuance process directly from the web page: then clicking on Issue, the wizard pops up with the name and email fields already filled in.

The first thing this filter does is a purely user-friendliness action: it inserts the site URL and title in the Bookmarks/Favorites list (accessible via a menu), unless there is already a bookmark for this site there. That way, the user can easily come back to this site without having to remember the URL.

Then the filter inserts the site's address in the Encryption Domain, which is just a simple set mapping host:port pairs to SSL ports and error URLs. Since the Encryption Domain filter is right next in the chain, the request will be immediately rerouted to the HTTPS dispatcher.

- **Encryption Domain Filter:** this filter checks whether the host:port in the URL of the current request is in the Encryption Domain. If it isn't, the filter simply lets it follow its way on the filter chain, so it will ultimately reach the standard dispatcher and sent to the network over HTTP.

  Otherwise, the request is taken out of the chain (so it won't reach the standard dispatcher anymore) and fed to the HTTPS dispatcher, which, in its turn, starts the SSL handshake to the port specified in the site's entry in the Encryption Domain.

  If the server requests client authentication, the HTTPS dispatcher asks for the user's private key for the default ID set for this site; if this key is not cached, the CSM will display a dialog box stating the exact site name and prompting for the user's private key. If, on the other hand, the server doesn't require client authentication, this step is skipped.

  At this point in the SSL handshake, the HTTPS dispatcher receives the server's certificate. If the CSM deems the it untrusted or if any network or handshake error happened, the dispatcher displays a dialog box explaining the failure and returns down the response chain a redirect to the error URL specified in the site's entry in the Encryption Domain (if none was specified, the connection is simply broken). This way, the site has an opportunity to display a nice message telling the user that the HTTP ⇔ HTTPS transition failed and maybe provide options to retry or choose other authentication options (such as plain old name+password). This in direct contrast with popular web browsers, which simply break the connection on SSL failures, leaving the non-technical user wondering what went wrong.

  Finally, if no errors occurred and the certificate is held as trusted, the original HTTP request is sent over the HTTP tunnel and the response inserted back in the response filter chain. Also notice that, due to the SSL session caching, the whole verification process above happens only in the first connection to the site or when the cache entry expires.

- **URL Rewriter:** This filterset is actually part of the HTTP Logon filterset described above. It acts only on `text/html` MIME types on requests through the HTTPS user agent. Its main purpose is to rewrite URLs of the form:

  ```
  https://example.com/
  ```

  as

  ```
  http://example.com/
  ```

  Supposing, of course, that "example.com" is in the encryption domain. If the site name in the above

URL is not in the encryption domain, it is left unchanged.

That way, any further requests initiated by consequences of the browser parsing the current HTML will again be sent to us – recall that the engager configured us as the browser's HTTP proxy only – we do not receive any HTTPS requests the browser generates. So this filter tries to ensure that all URLs in the HTML the browser receives point to URLs with the HTTP scheme what we can proxy.

The discussion omitted several details for the sake of clarity. In our implementation, it is comprised of two filters: a response header filter for rewriting URLs in the Location field during redirect messages; and a response body filter the that parses the HTML looking for tags with `src`, `href` and `action` parameters and rewrites only URLs within them – that way, any URLs within the readable text (outside the tags) won't be touched. We also made it rewrite URLs in `window.open` JavaScript instructions, since it occurs quite often in many websites.

The response body filter is the system's Achiles heel: since it is static, it misses any absolute URLs generated dynamically by embedded languages such as Java, JavaScript or VBScript, nor it sees absolute URLs embedded in Flash movies or other plugin-specific objects. However, things work remarkably well in sites where nearly all embedded URLs are relative.

- **Keygen Tag Mangler:** This filter replaces the `<KEYGEN...>` tag used by Netscape-derived browsers in web forms to generate a new keypair [12] by a combo box (a `<SELECT>...</SELECT>` sequence in HTML parlance) allowing the user to choose one of the allowed key sizes. The original name of the KEYGEN tag is prepended with "x-kapanga-keygen-", so that the Keygen Interceptor field described below can intercept it. This filter is only active when previously told so by the command parser.
- **Keygen Interceptor:** this filter acts on response bodies of POST requests and only when the mime-type is "application/x-www-form-urlencoded". It looks for form fields with the name starting with "x-kapanga-keygen-". Upon finding it, it starts the New Digital ID Wizard right in the point where the user chooses the passphrase (see Figure 4c). When the wizard is done generating the keypair, it is converted to an SPKAC and sent over the form field with its original name (i.e., the "x-kapanga-keygen-" previously prepended is removed).
- **Certificate Interceptor:** this filter grabs the response bodies in "application/x-x509-{user,ca,email}-cert" MIME types. It also looks for these content-types in each section of multipart MIME types as well – this is the mechanism used by web sites and commercial web-based CAs to install certificates and certificate chains. The data is decoded (DER/PEM-armoured detection is built in and both single certificates and PKCS#7 bags are supported) and inserted directly to the Certificate Store.

  If one of the inserted certificates matches a previously sent SPKAC, the automatic attestations are performed. If the inserted chain contains a Root CA but no automatic attestation has occurred, a dialog box pops up informing the user that he/she may be interested in performing a manual attestation.

2.3 Engagers

The engagers are responsible for setting up the data interception in each browser by inserting ourselves in the proxy chain through the following process:

- The browser's current proxy settings are detected and saved for later restoration;
- The address and port of the proxy the browser is currently using for sending HTTP requests is detected and the engager signals our default dispatcher to use this proxy. If the browser isn't using any HTTP proxy, we tell our default dispatcher to do the same and send the requests directly;
- The browser's HTTP proxy settings are overwritten with "localhost:ourport", where "ourport" is the port where we've previously started a server to listen to this specific browser's requests;
- The address and port of the proxy the browser is currently using for sending HTTPS requests is detected and the engager tells the HTTPS dispatcher to use this proxy. Unlike the HTTP proxy, however, we don't overwrite the browser's setting.

Implementing this seems simple, but each browser presented its own special cases.

The engager for Internet Explorer proved to be the simplest to implement because IE has simple API calls to change the settings and have its currently running instances instantaneously reload any changes made to it. Slight complications arise due to the several versions of IE and the API itself. The most severe is with IE versions 5 and above: since it supports per-dialup connection profiles, each with its own proxy settings, the process above has to be performed for each dialup profile. In the end, the IE engager we implemented works with all IE versions all the way

back to version 3. Version 2 and below didn't support proxies at all.

Implementing the Mozilla engager, on the other hand, proved to be quite a challenge because of the lack of a simple way (as far as we know) to signal its currently running instances of any changes in its settings. Mozilla's settings are read once during program startup and kept in memory. We can easily overwrite the configuration files and its format is quite simple (although figuring out where it is located means messing with `registry.dat /appreg` file [10]). This works well for inactive profiles, but not for the active ones – the running instance doesn't notice that Kapanga (an external process, from its point of view) changed the files and thus doesn't reload them. And it overwrites our changes when saving the settings back to disk as it finishes.

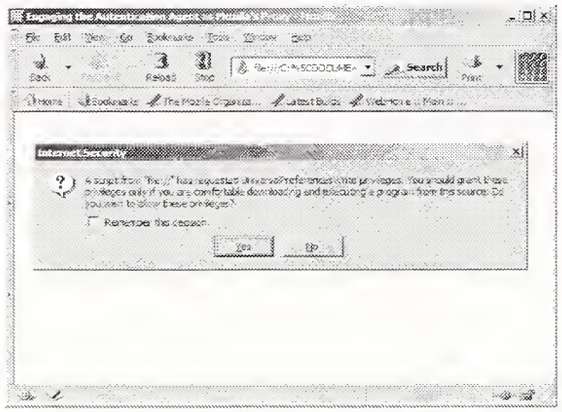


Figure 8: A JavaScript program is the only way to set the proxy settings in the currently running instances of Mozilla. However, since changing user settings is a privileged operation, its execution prompts a confirmation dialog box, somewhat thwarting the convenience of the engager.

The method we came up with works but is less than elegant: we generate a web page in a temporary, randomly-named file the local filesystem with a small JavaScript program that performs the changes. Then we direct the currenly running instances of Mozilla (via DDE [9] on Windows or X-Remote [8] protocol on Unix) to open this page. Since changing those settings requires granting special privileges to the script, the first time it is run Mozilla displays a confirmation dialog box, as shown in Figure 8 above.

The paranoid may regard this procedure as opening up a vulnerability itself – from there on, any local scripts (using the "file:///" scheme) may change Mozilla's preferences for that profile. It could have been worse, though: we considered and rejected the idea of avoiding the creation of a temporary file by sending the Javascript program over HTTP – that would mean the user should allow script execution over the "http://" scheme, which would open it up to abusive scripts from anywhere on the Internet.

A limitation of our current engager implementations is that they cannot handle Proxy AutoConfiguration [11], which is quite popular. Since implementing this support would require a quite capable JavaScript interpreter, we have chosen to deal with it in future versions; we felt that for the purposes of proving the concept, it was not essential.

Notice that engagers are just a convenience feature for users. They're obviously not necessary for the rest of the proxy to work, so long as the user changes the browser and Kapanga's proxy settings manually. That way, this whole system works even with browsers our implementation doesn't have specific engagers for. All that is required is that the browser must have proxy support. We've successfully run Kapanga in conjunction with many other browsers such as Konqueror, Opera and even Links (a console-based browser), just for kicks.

3 OTHER DESIGN ALTERNATIVES

Before settling for the particular set of design criteria and features we described, we considered and rejected a few other alternatives. While the reasons for some of them are pretty obvious, other are quite subtle and perhaps debatable. In the next subsections we describe a few choices we had to make and the rationalie behind tem.

3.1 Traffic Interception Method

Using the browsers' native proxy support was an obivous choice – web proxy technology was specifically design to intercept and forward HTTP traffic and it's widely deployed and matured. Not that we lacked choices:

- Internet Explorer has a feature called Browser Helper Objects [15] that could make interception a lot easier on that platform because we wouldn't have to deal with next-hop proxies and its particularities (PAC, multiple authentication methods, etc). However, we didn't want to confine Kapanga's applicability to Windows only; as previously mentioned, we wanted it to work with any browser on any platform;
- Implementing Kapanga as a SOCKS [20] proxy might also work, but it would involve guessing port numbers where HTTP traffic goes. Besides, not all proxies support SOCKS;
- Redirecting the socket API calls would not only require the same port number guesswork, but it would require a lot more system-dependent code and it wouldn't allow Kapanga and the browsers to run on different machines.

3.2 The Pure-Scheme vs. Cross-Scheme Dilemma

Kapanga uses what we call a *cross-scheme* system: when a site is in the Encryption Domain, we effectively map portions of the HTTPS scheme's

address space into the HTTP's address space. That is, the browser has no notion on whether the request is going through HTTP or HTTPS – this state information is in Kapanga's Encryption Domain. This has consequences:

- Bookmarks made when a site is in the encryption domain will probably not work when the site is not in the encryption domain or when Kapanga is not running at all (unless the site designer was very careful to handle this);
- We had to create the URL Rewriter Filter to force absolute URLs embedded in the HTML back to us. As previsouly mentioned, though, this fails with dynamically generated URLs.

Earlier in the design process, we considered – and rejected – what we called a *pure-scheme* system: we would actually implement two proxies, one strictly HTTP to HTTP and the other strictly HTTPS to HTTPS. Given that the namespaces don't collide, there would be no need for a URL Rewriter filter nor would we have problems with bookmarks.

This sounds like a good idea if we think only in terms of the pure HTTP proxy; however, given that SSL was specifically designed to be resistant to interception and tampering, the pure HTTPS proxy would have to be, in fact, a generic HTTPS spoofer/man-in-the-middle attack.

From a purely cryptographic point of view, this is quite easy to implement: during the initial SSL handshake, we send the browser a fake server certificate generated on the fly. From the user interface point of view, on the other hand, this has a problem: it triggers the browser's SSL warning dialogs, since the fake certificate isn't signed by a CA chain the browser trusts. This is clearly unnaceptable, not only in light of our philosophy of non-intrusiveness and minimum hassle for the users, but also because SSL-derived user interface problems are exactly what Kapanga was originally intented to solve in the first place.

We could make the SSL spoof work silently if we inserted a new root certificate in each browser's certificate store, but that would bring disadvantages: first, it would again limit Kapanga to run in the same computer as the browser (a restriction we didn't want to have); second, the exact mechanism for inserting new roots varies from browser to browser: IE stores trusted root CAs in the Windows Registry, while Mozilla-derived browsers use a Berkeley-DB file. This would increase the amount of platform-specific code Kapanga would have – something we've been trying to minimize all along –, not to mention that the process would fail if Kapanga runs without the proper privileges to write to those certstores.

There are other arguments against the SSL spoofer and the pure-scheme idea:

- Performance would suffer, since we'd have three encryption/decryption rounds: the browser encrypts the data, Kapanga would decrypt it, modify it and reencrypt it again;
- It wouldn't work on browsers without native SSL support; in contrast, the cross-scheme approach allows Kapanga to work even if the browser doesn't support SSL;
- Writing and releasing the code of a portable silent auto-engaging SSL spoofer would be more like giving a powerful weapon to the blackhats than a powerful protection to the average user.

Yet another advantage of the cross-scheme appoach is that we give the user the choice of not using Kapanga at all if he/she feels like, so we neither mess nor risk to break the user's web banking systems and other critical applications they already have running.

4 CONCLUSIONS AND FUTURE WORK

We described the architecture of a solution for perfoming the cryptographic and user interface aspects of HTTPS channel establishment and web form signature outside of the web browser. The key idea is to implement the crypto services in a proxy that rewrites the HTML on the fly and converts it to HTTPS when appropriate, so we can bypass the browser's and protocol limitations while retaining compatibility. Thus, any browser with proxy support can be used – the user is not forced to adopt any particular web browser. Another advantage is that our approach does not depend on any proprietary architecture such as ActiveX or Java.

Our primary motivation was to play with newer user interface concepts to make client-side PKI easier to use. A few results stand out: in other to make sites with client authentication that user's didn't hate, we had little choice but to address a few protocol and user interface gaps:

- A web site should be able to enumerate the user's certificate so as to offer assistance in registration as preparation for the HTTP-to-HTTPS transition (the SSL handshake with its certificate validation process);
- There had to be a way to redirect the user to an URL with a nice explanation, continuation options or alternative authentication methods when the SSL handshake fails. It's just not acceptable to break the connection and leave the user with a cryptic error message;
- The certificate issuance process shouldn't be so fragile as to break because of lack of ActiveX upgrades, different browser versions or the phase of the moon. Nor it should induce the user to store the private key without some effort to set up a decent passphrase. The process must be simple, reliable and hassle-free. Having it instantaneous is

a plus – with so many online services with instantaneous registration processes, it is hard to justify the severe identity validation procedures of most CAs;

- There really should be a simple way to do such a simple thing as signing a web form.

The implementation of those features in an external proxy enabled us to bypass the browsers limitations while providing the illusion that those features were "augmented" to the browser in an non-intrusive way. It also required minimal or sometimes no change to the server side at all: nothing needs to be changed for client-based HTTPS authentication; a simple change in the action URL in HTML forms enables form field signature (bigger changes may be needed if the application needs to validate the signatures); and small changes to the HTML page where the Nestcape vs IE issuance process decision is made is enough to support Web-based commercial CAs.

The price of this "backwards" compatibility is paid in the considerable complexity of the architecture and the horrible contortions our tool has to go about to implement them. Some problems may not have a good solution at all, such as the static nature of the URL rewriter filter not being able to handle dynamically generated URLs; this limits the tool's applicability to "well behaved" sites only.

There are many worthwhile future improvements on sight. Making the proxy work for generic TCP connections as well as HTTP may help extend the use of client-side authentication for several other protocols such as SMTP, POP3, VNC, X and many others. Extending the program to act as a Mail Transfer Agent (since every MTA is kind of a proxy) holds the potential for allowing us to make the same for mail clients: having the digital signature generation and verification be performed out of the mail client. Going further in this idea, we could also add support for encryption and decryption to allow message confidentiality. We are also working on making the CSM support PGP and SSH keys as well in order to achieve Jon Callas' concept of *format agnosticism* [21], consonant with our philosophy of bridging the PKIs together.

5 ACKNOWLEDGEMENTS

Thanks go to our co-developer Tiago Assumpção and to Felipe Nóbrega for implementing both the CAs used for the instantaneous certificate issuance and the public CSM server. We'd also like to thank Justin Karneges for writing the first versions of the Q Cryptographic Archictecture [19] on which our implementation is based. We are also grateful to the anonymous reviewers for their invaluable criticisms and suggestions for this paper.

Horacio and Theco are comic book characters from Mauricio de Sousa Produções that are arguably part of Brazilian pop culture, used here for entirely non-profit illustration purposes.

6 REFERENCES

1. Tim Berners-Lee et al., *RFC 2616: Hypertext Transfer Protocol – HTTP/1.1*, 1999, http://www.faqs.org/rfcs/rfc2616.html
2. Eric Rescorla, *RFC 2818: HTTP Over TLS*, 2000, http://www.faqs.org/rfcs/rfc2818.html
3. Eric Rescorla, *SSL and TLS: Designing and Building Secure Systems*, 2001, Addison-Wesley, ISBN 0201615983.
4. Marco Carnut, Cristiano Lincoln Mattos, Evandro C. Hora & Fábio Silva, *FreeICP.ORG: Free Trusted Certificates by Combining the X.509 Hierarchy and the PGP Web of Trust through a Collaborative Trust Scoring System*, 2003, Proceedings of the 2nd PKI Research Workshop, http://middleware.internet2.edu/pki03/presentations/02.pdf
5. Ari Luotonen, *Tunneling TCP based protocols through Web proxy servers*, 1998, http://www.web-cache.com/Writings/Internet-Drafts/draft-luotonen-web-proxy-tunneling-01.txt
6. Steve Lloyd et al, *Understanding Certificate Path Construction*, 2002, PKI Forum, http://www.pkiforum.org/pdfs/Understanding_Path_construction-DS2.pdf
7. Steve Lloyd, *AKID/SKID Implementation Guideline*, 2002, http://www.pkiforum.org/pdfs/AKID_SKID1-af3.pdf
8. Jamie Zawinski, *Remote Control of Unix Netscape*, 1994, http://wp.netscape.com/newsref/std/x-remote.html
9. MSDN Library, *About Dynamic Data Exchange*, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winui/WinUI/WindowsUserInterface/DataExchange/DynamicDataExchange/AboutDynamicDataExchange.asp
10. Daniel Wang, *Mozilla Profile registry.dat File Format*, http://wangrepublic.org/daniel/mozilla/registry
11. Nestcape Inc., *Navigator Proxy Auto-Config File Format*, 1996, http://wp.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html
12. Netscape Inc., *Netscape Extensions for User Key Generation*, http://wp.netscape.com/eng/security/comm4-keygen.html

13. Arnold G. Reinhold, *The Diceware Passphrase Home Page*, http://world.std.com/~reinhold/diceware.html
14. Sean Smith, *Effective PKI Requires Effective HCI*, ACM/CHI Workshop on Human-Computer Interaction and Security Systems, 2003, http://www.cs.dartmouth.edu/~sws/papers/hci.pdf
15. Dino Esposito, *Browser Helper Objects: The Browser the Way You Want It*, 1999, Microsoft Corp., http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebgen/html/bho.asp
16. RSA Data Security Inc, *RSA Keon WebPassport*, `http://www.rsasecurity.com/node.asp?id=1230`
17. Verisign Inc., *Go! Secure for Web Applications*, http://www.verisign.com/products-services/security-services/pki/pki-security/pki-solution/web-application/
18. John Marchesini, Sean. Smith & Meiyuan Zhao, *Keyjacking: the surprising insecurity of client-side SSL*, http://www.cs.dartmouth.edu/~sws/papers/kj04.pdf
19. Justing Karneges, *Q Cryptographic Architecture*, http://delta.affinix.com/qca/
20. M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas & L. Jones, SOCKS Protocol Version 5, http://www.faqs.org/rfcs/rfc1928.html
21. Jon Callas, *Improving Message Security With a Self-Assembling PKI*, 2003, Proceedings of the 2nd PKI Research Workshop, http://middleware.internet2.edu/pki03/presentations/03.pdf

Side-Effects of Cross-Certification

James L. Fisher, Ph.D.
Mitretek Systems, Falls Church VA
jlf@mitretek.org

While many organizations lean towards cross-certification with bridge certification authorities (BCAs) for wider PKI interoperability, there are many hidden details that can affect operating capabilities as well as legal standings. The purpose of this paper is to share lessons learned to help the reader to understand implications of choosing a cross-certificate based trust model. Topics covered include: X.500/LDAP certificate directory implementation and interoperability requirements; transitive and asymmetrical trust; choosing the proper trust anchor; cross-certificate policy mappings; and Certification & Accreditation requirements.

This paper does not examine the trust path discovery process—it focuses on the necessary, enabling configuration components that enable path discovery and path validation to be automated.

There are technical solutions to the issues presented herein. However, due to their inherent complexity, a discussion of alternative solutions is beyond the scope of this paper.

Benefits of Certificate Bridges

One of the primary advertised benefits of a certificate bridge is that it allows the relying party to enjoy the benefits of a larger trust domain while not being required to be an integral part of the certificate hierarchy of that other trust domain, all while trusting only one trust anchor—a public key which is within its own trust domain.

The other benefit is that the relying party can also be relatively sure of the certificate holder's identity, based on the trust placed in others to validate an organization's identity. It's not that cross-certification automatically grants that peace of mind, but rather that it is standard practice for each party considering cross-certification to scrutinize the other party's pre-issuance identity vetting policies, private-key protection policies, CA and directory infrastructure operational policies, etc. Thus, an issued cross-certificate represents a thorough background check with acceptable findings.

Issued cross-certificates can be used to dynamically assemble a chain of certificates called a trust path which spans the gap between the certificate issuer and the relying party. The complete set of certificates comprising the certificate trust path (including supporting time-specific validity statements) forms a tangible record of trust that can be stored for future evidence of due diligence. This might be necessary for institutional archival purposes or to satisfy National Archive and Records Administration (NARA) requirements.

Directory Interoperability

The operating authority of a BCA typically provides a publicly available X.500 or LDAP directory for publishing issued CA certificates, cross-certificates, and often certificate revocation lists (CRLs). Equally important, these X.500/LDAP directories are configured to facilitate trust path discovery and validation by providing chaining to, or referrals to, all other CA directories to which cross-certificates have been issued. The intent is to provide a one-stop-shop virtual directory from which all relevant certificates and CRLs can be retrieved during the path discovery and validation processes.

In practice, each cross-certifying organization typically has its own

X.500/LDAP directory to which its own users (certificate issuees and/or employees) point the LDAP clients in their local validation engines, and if the requested certificate or CRL does not fall under that local directory's base distinguished name (DN), a superior reference chains (transparently to the user) to the master BCA for retrieval. No matter where the certificate or CRL resides, it is returned to the LDAP client. Without such chaining, there is currently no way for a desktop LDAP client to discover what directory to connect to for retrieval of a certificate or CRL. This necessary processing has interesting implementation implications.

First, the BCA directory must be able to resolve any base DN that could *ever* form a trust path through that BCA. An illustration will help in understanding the details.

In the examples we use in this paper, assume the existence of the following fictitious PKI participants:

- A Government (Certification) Bridge Authority/Architecture (GBA)
- A Government Institution (GI)
- A Neighboring Nation Bridge Authority/Architecture (NNBA)
- A University Bridge Authority/Architecture (UBA)
- An older, legacy PKI system at the Enormous State University (ESU1)
- A newer PKI system at the same university (ESU2)
- A World Wide Council (WWC)
- A random transoceanic nation (RTN)

The following list describes the cross-certifications that have occurred between the above fictitious entities, and the resulting trust mesh appears in Figure 1:

- The Government Bridge—GBA—(with base DN `ou=GBA, o=Upper Government, c=US`) has cross-certified with only:
  - GI (with base DN `ou=GI, o=Upper Government, c=US`)
  - UBA (with base DN `o=edu, c=US`)
  - Neighboring Nation (with base DN `c=NN`)
- The University Bridge (UBA) has also cross-certified with:
  - The original Enormous State University infrastructure, ESU1 (with base DN `o=ESU Provosts, c=us`)
  - The new Enormous State University infrastructure, ESU2 (with base DN `o=ESU Provosts, c=us, dc=esu, dc=edu`)
- The Neighboring Nation has also cross-certified with:
  - The World Wide Council (with base DN `dc=int`)
- The World Wide Council cross-certifies with:
  - The Random Transoceanic Nation (with base DN `c=RTN`)

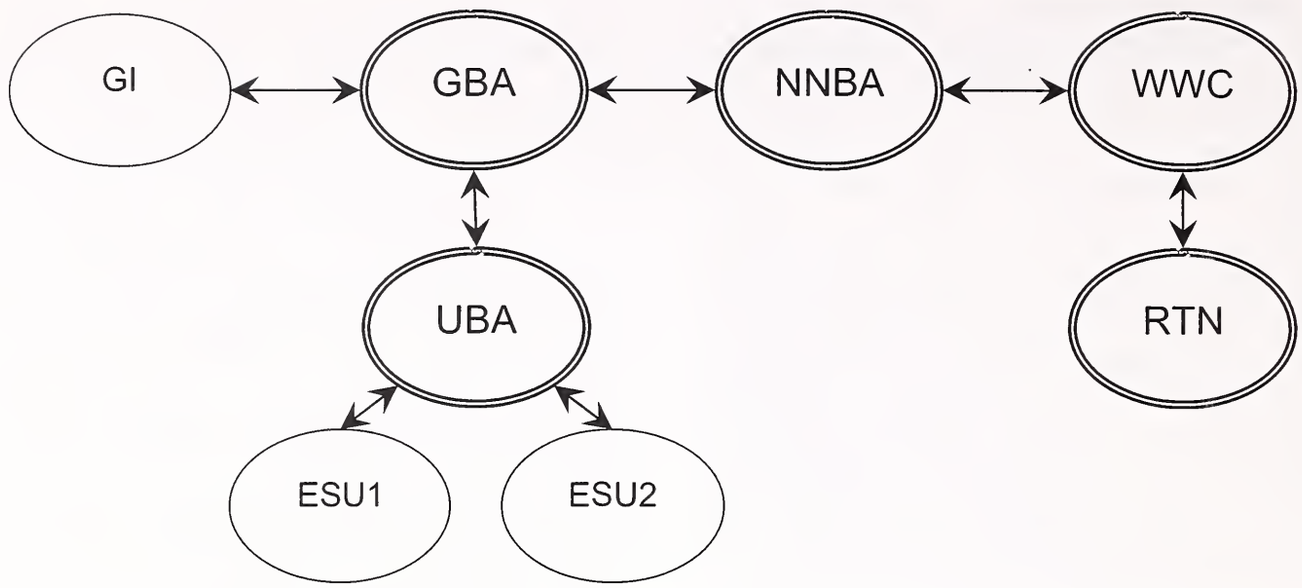


Figure 1: Cross-Certification Between Trusting Partners

Figure 1 allows us to see more clearly the many and varied trust paths that can be formed. For instance, through certificate bridges and cross-certificates, the Government Institution should be able to trust digitally signed messages from the Enormous State University, regardless of whether the signer's certificate was issued from ESU's old or new PKI infrastructures. Additionally, cross-certificates would also allow the Government Institution to trust digitally signed messages from the Random Transoceanic Nation—this trust path may or may not be intentional or desired, as we will discuss later. Let us now look at the certificate/CRL directory configurations required to enable a relying party to easily discover and validate a trust path.

Before discussing the supporting certificate directories, it will be helpful to review the definition of key X.500 directory knowledge references and concepts:

- Directory Server Agent (DSA): the software providing the X.500 directory service for a particular hierarchal directory information base (DIB)
- Name context: a subtree of a directory, and is identified by the DN of the topmost entry (the "base DN"); in many commercial databases, a DSA's database can contain more than one name context
- Cross-reference: specifies a name context (usually within a remote DSA) that is not a child of this DSA's name context; a cross-reference typically points directly to the entry in the name context hierarchy
- Subordinate reference: specifies a name context (usually within a remote DSA) that is a child of this DSA's name context
- Superior reference: specifies the parent DSA that holds entries outside this DSA; only one superior reference per DSA is permitted

For complete directory chaining, the (fictitious) US-based certificate directories are configured as follows, with Figure 2 representing the same information pictorially:

- The GI directory DSA is rooted at `ou=GI, O=Upper Government, c=US`, and has one superior reference to the GBA directory

- The ESU directory has:
  - One DSA rooted at `o=ESU Provosts, c=us`
  - A second DSA (or a second naming context under the first DSA) rooted at `o=ESU Provosts, c=us, dc=esu, dc=edu`
  - One superior reference to the UBA directory
- The UBA directory has:
  - One DSA rooted at `o=edu, c=US`
  - A cross-reference from the DN `o=ESU Provosts, c=us` to the original ESU directory
  - A second DSA (or a second naming context under the first DSA) rooted at `dc=edu`
  - A subordinate reference under the second DSA from the DN `o=ESU Provosts, c=us, dc=esu, dc=edu` to the new ESU directory
  - A superior reference to the GBA directory
- The GBA directory has:
  - One DSA rooted at `c=US`
  - A subordinate reference from the DN `ou=GI, o=Upper Government, c=US` to the GI
  - A subordinate reference from the DN `ou=UBA, o=edu, c=US` to the UBA directory
  - A subordinate reference from the DN `o=ESU Provosts, c=us` to the original ESU directory
  - A cross-reference from the DN `c=NN` to Neighboring Nation's border directory
  - A cross-reference from the DN `dc=edu` to UBA's second DSA
  - A cross-reference from the DN `dc=int` to the WWC's border directory
  - A cross-reference from the DN `c=RTN` to the RTN's border directory
  - (no superior references)

(We leave it as an exercise to the reader to consider the cross-references needed at the WWC and RTN border directories.)

Notice this very important fact: in order to be able to retrieve (via chaining) certificates issued by RTN's CA, the GBA directory must include a knowledge reference for the `c=RTN` DN (pointing to the RTN directory) *even though the GBA did not directly cross-certify with the RTN*. This requirement has potentially serious implications on how easily a BCA can dynamically expand to accommodate indirect trust agreements.

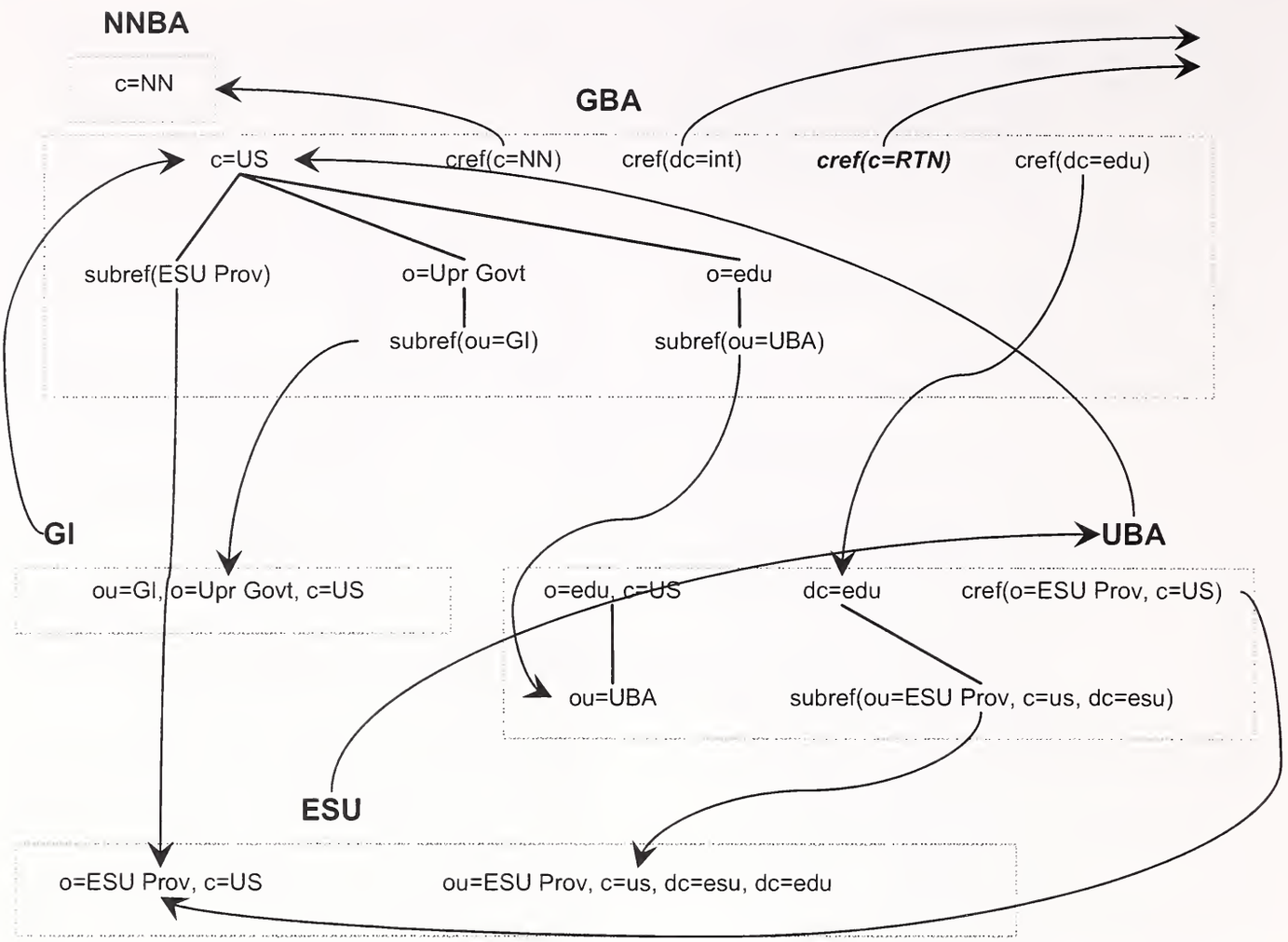


Figure 2: Directory Chaining Agreements, and Subordinate and Superior References

Notice that because of DN naming conventions chosen, and how each DSA is rooted, both GBA and UBA directories need a cross-reference from the DN `o=ESU Provosts, c=us` to the original ESU directory.

One can easily see how quickly border directory configuration becomes complicated when multiple BCAs cross-certify. Each BCA must be configured with knowledge of all possible directories traversed along a trust path, even if the BCA did not cross-certify directly with the corresponding CA.

As seen, each border directory needs the capability of supporting:

- multiple root DSAs or multiple name contexts within a single DSA
- cross-references
- subordinate references
- superior references

Fortunately, these capabilities are found in a number of commercial X.500 directories, such as: i500 by PeerLogic; eTrust by Computer Associates; M-Vault by ISODE. However, Microsoft Active Directory does not support superior references, non-local cross-references, or non-local subordinate references. Therefore, organizations wishing to provide a publicly accessible certificate directory (often called a border directory) *to their relying parties* suitable for use in automated path discovery should export all necessary certificates and CRLs from their Microsoft Active Directory and import into a directory supporting the aforementioned types of references.

Alternatives:

The BCA directory was required to provide a multitude of cross-references and subordinate references because directory chaining was desired. One design alternative is to have the BCA directory simply return *referrals* to only those directories with which it is directly cross-certified. But this presents two problems in today's environments:

- Many LDAP clients still can not process directory referrals
- Trust paths traversing more than one bridge would not be discovered if the pertinent certificate fields contained only DN-formatted information and not URI-formatted information

Directory chaining would not be necessary if all certificates had properly formatted AIA fields, and the local validation client could understand all AIA formats. However, one missing or improperly formatted AIA field would destroy the ability to discover a trust path.

Transitive Trust

When CAs cross-certify, the phenomenon of transitive trust comes into play. Such indirect trust may or may not be intended or welcomed. An internationally oriented illustration will make the side-effects clearer.

Let us extend the example of the previous section to include one additional cross-certificate pair. Assume that the very recently formed country of Forbiddenstan has also cross-certified with the World Wide Council for the purpose of discussing commercial tradė. Consequently, there is now a new trust path from the GI, through the GBA, through a Neighboring Nation, through the World Wide Council, to Forbiddenstan. Let us also assume that "the United States maintains a broad embargo against trading with Forbiddenstan, and most commercial imports from Forbiddenstan are prohibited by law." And finally, to ensure compliance with federal regulations, let us assume that GI directors would prefer that there be no valid certificate trust path from GI to Forbiddenstan.

To prevent the formation of such a trust path, cross-certificates must be re-issued to specify new name constraints or path length constraints. There are two logical places in the trust chain where such a filter could be positioned. Since this is a federal law, the GBA's cross-certificate issued to the NNBA could contain a name constraint to filter out Forbiddenstan's `c=FB` DN. However, since other domestic humanitarian-oriented government agencies might have legitimate needs to trust Forbiddenstan-signed documents, a nationwide filter might not be appropriate. Therefore, a GI itself might need to insert name constraints into the cross-cert it issues to the GBA. Additionally, all other relying parties would similarly need to be aware of this political situation and place appropriate name constraints in their cross-certificates.

While this method works, it is very difficult to envision all necessary path constraint needs—expressed either as path permitting or path inhibiting rules—before cross-certificate issuance. And, re-issuance of a cross-certificate is not without its labor costs. Revoking previously issued cross-certificates will also be necessary to force validation engines that pre-cache the validation paths to refresh themselves. More importantly, the need to restrict certain trust paths is typically not realized until after an "inappropriate" trust path is formed.

Choosing the Proper Trust Anchor

Another challenge of a multiple cross-certificate environment is choosing the correct trust anchor and certificate policies on which to filter. Complicating factors include:

- Asymmetrical policy mappings
- The possibility of multiple trust paths

- Unidirectional cross-certification (e.g., a GBA may issue a cross-certificate to a military to facilitate GBA-centric trust path discovery, but the military BCA may choose to not issue a cross-certificate to that GBA)

Policy mappings are often placed in cross-certificates for the purpose of declaring how the levels of assurances (LOAs) in the subject's domain translate to the LOAs in the issuer's domain. In order to make use of these policy mappings, RFC3280 indicates that path discovery and validation algorithms must specify a trust anchor and a set of acceptable certificate policies (in the trust anchor's domain). Additionally, if the initial set of acceptable certificate policies is a subset of those mapped, then the effect is to filter out any trust paths involving certificates with LOAs that do not map to that initial set.

Consider the following example. The (fictitious) State of Algonk has one CA (with one private key) issuing end-entity certificates with one of four levels of assurance (LOAs): Level A, Level B, Level C, and Level D. Independently, GSA has three LOAs: Small, Medium, and Large. But when the State of Algonk cross-certified with the GBA, Algonk submitted documentation describing only their Level C LOA, therefore the cross-certificate issued by GBA to Algonk contains only one policy mapping: Algonk Level C maps to GBA Medium.

Furthermore, the cross-certificate pair between the GBA and the Algonk contains asymmetrical policy mappings. Why? Because each party's Policy Authority (PA) could independently evaluate the other party's Certificate Policy/Certification Practice Statement (CP/CPS), and there is no guarantee the parties will view each other's policies equally. Consequently, according to the policy mappings found in the cross-certificate issued by the GBA to Algonk, the GBA views Algonk Level C LOA as mapping to the GBA Medium LOA. Conversely, according to the policy mappings found in the cross-certificate issued by Algonk to the GBA, the State of Algonk views GBA Medium as mapping to Algonk Level B.

Typically, the trust anchor of choice is a public key within one's own issuing hierarchy. However, let us consider the case were a centralized, organization-independent validation service (employed by a GBA) is being established to validate only certificates that are GBA Medium or equivalent. Let us examine two trust anchor options and their implications.

If the trust anchor is a GBA root public key, then the certificate policy OIDs on which to filter should be GBA Medium. In this case policy-filtered trust paths can be found from the GBA to (a) Algonk Level C certificates, but not to other Algonk LOAs, and (b) any other issuers' certificates that GBA maps to GBA Medium LOA.

Alternatively, the validation service operator could reason that since Algonk certificates are validated so often, the dynamically discovered trust path for Algonk certificates should be made as short as possible for reasons of efficiency. To shorten the trust path, the trust anchor is chosen to be the State of Algonk's root public key. Since the GBA views only Algonk Level C certificates as equivalent to GBA Medium LOA, the initial set of acceptable certificate policy OIDs is just the policy OID representing Algonk Level C. Obviously, trust paths will be found to Algonk Level C certificates. However, the policy mapping in the Algonk-issued cross-certificate (i.e., the cross-certificate going in the opposite direction) states that GBA Medium maps to Algonk Level B. Since Algonk's Level B certificate policy OID is not in the initial set of acceptable certificate policies, no other issuer's certificates mapping to GBA Medium (as determined GBA's point of view) will be accepted (i.e., no valid trust path will be discovered for those certificates). And the configuration decision complications increase as more BCAs become involved. It is therefore

recommended to explicitly state the validation service's acceptable certificate policy set—and not include the phrase "or equivalent"—and use only the corresponding trust anchor.

In practice, such asymmetrical mappings can be easily avoided. Both parties should explicitly state and agree on how each of the applicant's LOAs relates to the issuing party's LOAs. Continuing with our example, when applying for cross-certification, the State of Algonk should explicitly request that the GBA PA map Algonk's Level C LOA to GBA's Medium LOA.

Post-Issuance CA Subordination

One technique for reducing the number of certificate bridges is to establish one root under which all other certificates are issued. Recently, there have been discussions of establishing such a common root (CR) that would subordinate existing commercial CAs that meet government requirements.

A common root would also offer the advantage of needing to distribute only one self-signed public key in popular web browsers.

For proper policy OID representation, one of following two items must occur:

- The new CR must assert all policy OIDs of all subordinated CAs, or
- The end-entity certificates of the subordinated CAs must be re-issued to include the new CR policy OID

Figure 3 depicts one phase of the proposed hierarchy. Assume the GBA has previously cross-certified with one of the commercial vendor's CAs (CVCA). The root CVCA has issued a proper subordinate CA A1, and A1 issues only special-audience end-entity certificates. These end-entity certificates, when processed through the certificate mappings in the GBA/CVCA cross-certificate, map to a GBA Medium LOA. Assume the common root, CR, was subsequently cross-certified with the GBA.

Then, in the final phase, the proposed technique for demonstrating that A1 qualifies as a Scrutinized Provider is for the CR to subordinate the CA A1. In practice, this would result in a new subordinate CA A1*. A1 and A1* have the same public key and subject DN (to validate the already-

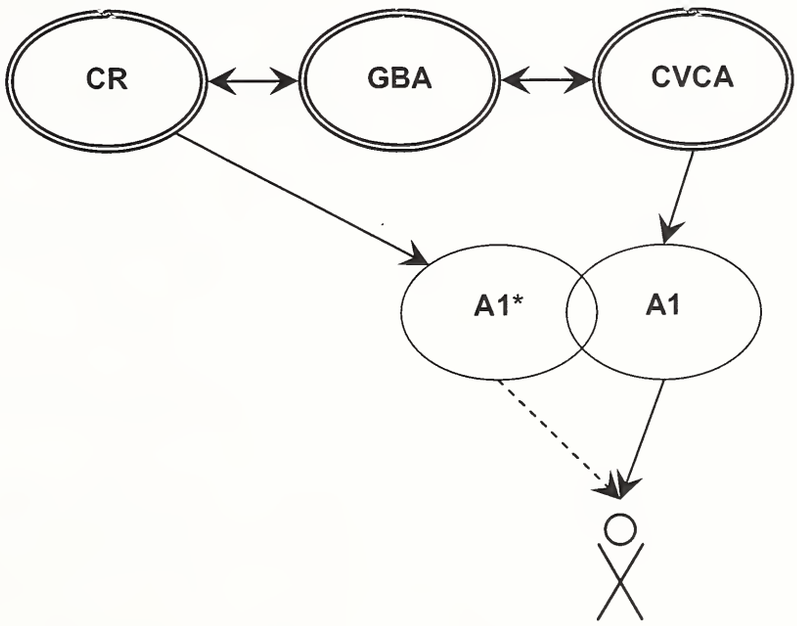


Figure 3: Subordinating Operational CA A1

existing end-entity certificates issued by A1), but different issuer DNs. This results in two possible trust paths from the CR root to the A1-issued certificate:

- One directly from the CR through the subordinated A1* CA certificate, and
- The second from the CR through the CR/GBA cross-certificate, through the GBA/CVCA cross-certificate, and finally through A1

When processed through the second trust path, the policy mappings in the cross-certificates will ensure the end-entity policies are properly mapped into the CR's certificate policy OID space. However, in direct trust hierarchies, such as from CR through A1* to the end-entity certificate, no policy mappings can take place. Therefore, A1-issued certificates must include two sets of certificate policy OIDs—one set for the CR policy name space, and the other set for the CVCA policy OID space—thus reflecting A1's two superiors.

Interestingly enough, if a relying party were given just the CR root, the subordinated A1* CA certificate issued by the CR root, and an end-entity certificate issued by CA A1, the relying party would find a perfectly valid hierarchical issuance path from the CR to the end-entity cert. However, if CVCA revoked A1 (say, due to a compromise of A1's private key), and the organization behind CVCA did not notify the GBA Program Management Office (PMO), then the above direct trust path through A1* would still appear valid when, in reality, it should be declared as “revoked.”

Therefore, extreme caution should be used if such a "two master" topology is used.

Operations Policies

Typically, one focuses on technical and policy issues within one's own security domain. In this section we consider operations policies requirements such as Security Certification and Accreditation (C&A).

A popular misconception in writing system security plans (SSPs) is that when a BCA is cross-certified with the root CA of a certificate issuer hierarchy, the BCA PMO is required to see only the C&A report corresponding to the remote organization's root CA, and that organization can be trusted to silently perform C&As on their internal hierarchy. However, a careful review of NIST Special Publication (SP) 800-37, "Guide for the Security Certification and Accreditation of Federal Information Systems," reveals more stringent requirements.

SP 800-37 defines a certification agent as "an individual, group, or organization responsible for conducting a security certification, or comprehensive assessment of the management, operational, and technical security controls in an information system to determine the extent to which the controls are implemented correctly, operating as intended, and producing the desired outcome with respect to meeting the security requirements for the system." (p.15) Additionally, SP 800-37 states that since the certification agent "provides an independent assessment of the system security plan to ensure the plan provides a set of security controls for the information system that is adequate to meet all applicable security requirements," the certification agent needs to be independent from:

- "persons directly responsible for the development of the information system and the day-to-day operation of the system"
- "individuals responsible for correcting security deficiencies identified during the security certification"

Given a certification agent's independence from the managerial and operational chains of a CA, the resulting report cannot remain solely within the organizations chain of command—the report must be delivered externally. The organization most interested in and most impacted by such a

report is the BCA PMO, and therefore should be the recipient of the certification report.

Therefore, the PMO of each BCA should regularly see the C&As of *every* issuing CA to which the BCA can form a trust chain.

Conclusions

As we have discussed, "bridging" for PKI interoperability is not the panacea that many thought it to be. It is extremely complex and requires careful attention to detail. It is easy to structure unintended and difficult-to-detect consequences. Such complexity often results in significant opportunities for undetected errors that the security community often points out as exploitable vulnerabilities. We also implore each organization to consider these inherent issues when performing security C&A and Certificate Policy/Certification Practice Statement (CP/CPS) Compliance Audits.

References

NIST Special Publication (SP) 800-37, "Guide for the Security Certification and Accreditation of Federal Information Systems" [http://csrc.nist.gov/publications/nistpubs/800-37/SP800-37-final.pdf]

RFC3280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" [http://www.ietf.org/rfc/rfc3280.txt]

Evaluating the Performance Impact of PKI on BGP Security

Meiyuan Zhao*and Sean W. Smith
Department of Computer Science
Dartmouth College

David M. Nicol
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Abstract

The Border Gateway Protocol is central to making the Internet work. However, because it relies on routers from many organizations believing and passing along information they receive, it is vulnerable to many security attacks. Approaches to securing BGP typically rely on public key cryptography, in various encodings, to mitigate these risks; to work in practice, these approaches usually require public key infrastructure. This cryptography and the PKI may both potentially impact the performance of this security scheme; however, evaluating how these effects may scale to large networks is difficult to do analytically or empirically.

In this paper, we use the tools of simulation to evaluate the impact that signatures, verification, and certificate handling have on convergence time, message size, and storage, for the principal approaches to securing BGP.

1 Introduction

By distributing and maintaining routing information, the *Border Gateway Protocol (BGP)* [32, 39] plays a central role in making the Internet work. However, BGP relies on hearsay information. BGP speakers trust the messages they receive and they completely trust other BGP speakers to follow the protocol specification reliably. Consequently, BGP—and the Internet it routes—is vulnerable to many potential attacks by malicious players [26]. To mitigate these risks, many researchers have proposed security mechanisms to authenticate the routing information transferred between BGP speakers [1, 8, 13, 17, 35, 40, 41]. *S-BGP* is the dominant scheme here.

Because of the need to authenticate information passed among parties spanning a large set of domains, these security mechanisms typically rely on public key cryptography. Implicitly or explicitly, public key *infrastructure* thus also becomes a critical component—otherwise, how do the parties know what public keys to use and whether they are still valid?

Neither public key cryptography nor public key infrastructure come for free. However, when designing and analyzing these large information-distribution systems, it's easy to overlook these implementation details, and the performance impact they can have on the overall protocol. Furthermore, given the large, messy nature of Internet routing, it can be hard to evaluate this impact: analytic techniques may fail to capture the complexity, and empirical techniques may require a prohibitively large testbed.

In previous work [27], we used the tools of *parallel simulation* to evaluate the performance impact of basic signing and verification on route attestations—and proposed and evaluated an improved way of generating and encoding this information. In this paper, we extend this work:

- to consider two new aspects of performance: *message size* and *memory cost*;
- to consider the PKI impact of recent proposals for in-band *origin authentication*;
- to consider the performance impact of standard PKI *revocation* schemes; and
- to consider the potential improvement of using recent *aggregate signature* schemes in place of standard signatures in assertion chains.

We find that among the half dozen techniques studied there is no clear best solution. Compared to the technique that uses the least memory, the technique that supports the fastest convergence time is three times faster but uses twice the memory. Signing cost is what matters for speed

*contact author, zhaom@cs.dartmouth.edu

(and BGP convergence) but this comes at a price, memory and message size.

This Paper Section 2 reviews BGP and S-BGP. Section 3 reviews some alternate encoding and cryptographic approaches. Section 4 presents our evaluation methodology. Section 5 presents our experiments and results for path authentication. Section 6 presents our experiments and results for origin authentication. Section 7 reviews related work, and Section 8 concludes with some thoughts for future research.

2 BGP and S-BGP

The Border Gateway Protocol (BGP) [32, 39] is the routing protocol for maintaining connectivity between *autonomous systems (ASes)* in the Internet. Each AS is assigned a unique integer as its identifier, known as its *AS number*. An AS manages subnetworks expressed as IP *prefixes*—a range of IP addresses. A BGP *speaker*—a router executing BGP protocol—constructs and maintains *forwarding tables* that enable packet forwarding. A BGP speaker maintains connections with neighboring speakers, known as its *peers*, and sends an **Update** to announce a new preferred route to prefix p. The route is a `(prefix, AS path)` tuple. The *AS path* is a sequence of AS numbers that specifies a sequence of autonomous systems through which one can traverse the network; last AS in the sequence is the *originator* of this route. For instance, if the autonomous system AS_k owns IP prefix p, the autonomous system AS_0 might send out an **Update** $(p, \{AS_0, AS_1, \ldots AS_k\})$ to announce its preferred route to p. Each BGP speaker keeps received routes in its *routing table*; for each prefix, the speaker tags one route as its preferred one.

Typically, a speaker's routing table changes when it adds a new route, deletes a preferred route, or replaces a previously preferred route with a new one. BGP speakers incrementally send **Update** messages to announce such changes to their peers. When speakers establish (or reestablish) a *BGP session*, they share their own routing table with each other via a large number of **Update** messages announcing routes in their routing tables. If it results in new preferred routes, processing of an **Update** message may create a number of new **Updates**. If the speaker chooses to announce a new preferred route, it extends the existing AS path by perpending its AS number to it and sends it to all of its peers, except the one who sent the route earlier. When a speaker announces a route to prefix p, it implicitly *withdraws* the last route it announced to p. The recipient, understanding this implicit route withdrawal, decides whether it prefers the new route. A withdrawal can also be an explicit announcement, with no mention of an alternative preferred route. In this case, the recipient may examine the previously received routes to the same prefix and decide whether there is an alternative to announce to its peers. If no such route found at hand, it simply withdraws the route as well.

BGP rate-limits the sending of **Update** messages with parameter called the *Minimum Route Advertisement Interval (MRAI)*, which is basically the minimum amount of time that must elapse between successive batches of **Updates** sent to a neighbor. BGP speakers have output buffers to keep waiting **Update** messages, and send them in batches when reaching the MRAI. A speaker may have a different MRAI for each of its peers or may have one MRAI that controls all peers. In practice, throughout the Internet, the default value the MRAI is 30 seconds.

Any change of network reachability will be reflected in the routing table of some BGP speaker. BGP will then propagate this change via **Update** messages through the entire network, like a wave. The *convergence time* measures the length of time for such wave of announcements to die out completely—in other words, for the network to return to a stable state. During the transient period of convergence, the continual changing of preferred routes degrades the effectiveness of packet forwarding. Longer convergence times thus reflect increased network instability and may cause severe network performance problems. Studies of BGP have considered convergence [10, 20, 34] and possible optimizations to control and accelerate it [11, 19, 21, 23, 30, 38].

Because BGP is central to Internet functionality and is vulnerable to malicious actors, we need to secure the information that BGP distributes. We consider each component:

- *Origin authentication* considers whether the originating AS really controls a claimed IP address range.
- *Path authentication* considers whether a claimed path to reach some IP prefix is in fact valid.

The dominant security solution, *Secure BGP (S-BGP)* [17] focuses on the **Update** messages. The first step of S-BGP is to set up public key infrastructures to help establish the authenticity of the involved parties. S-BGP uses X.509 [12] public key certificates and puts BGP-related information into certificate extensions. Speakers digitally sign the **Update** messages they announce to peers; with these X.509 certificates, recipients can verify the signatures to authenticate the received routes.

More specifically, each speaker uses *address attestations (AAs)* for origin authentication, and *route attestations (RAs)* for path authentication.

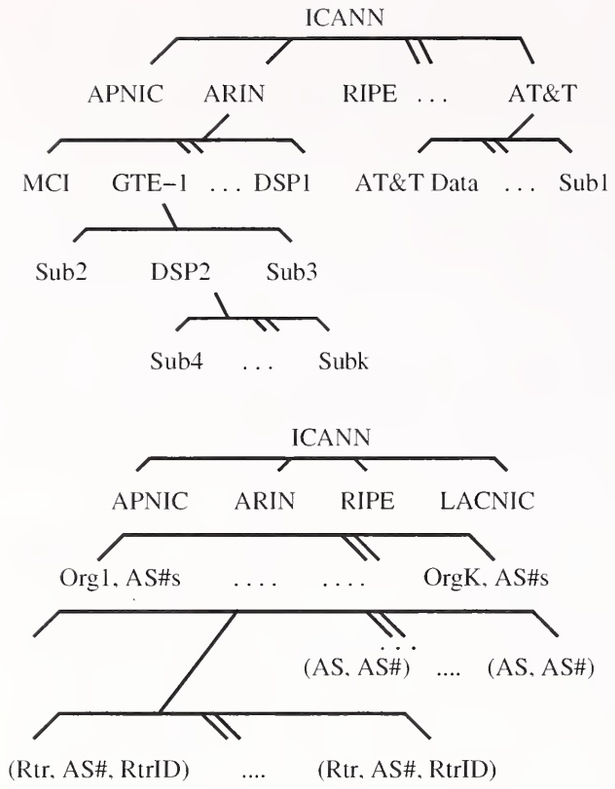


Figure 1: Sketch of the S-BGP PKIs.

2.1 S-BGP PKIs

To enable validation of attestations, S-BGP proposes two X.509 public key infrastructures. The first PKI contains certificates to authenticate the *owners of portions of the IP address space*. The second PKI is to authenticate BGP speakers, ASes, and the owners of ASes. Figure 1 illustrates the structures of these PKIs. Both PKIs are hierarchies rooted at ICANN [15]. ICANN issues itself self-signed certificates and further issues certificates to the the first tier of organizations, typically *Regional Internet Registries (RIRs)* such as ARIN, RIPE, APNIC, and LACNIC.

For the address allocation PKI, ICANN issues itself a certificate claiming the ownership of entire IP address space on the Internet. Consequently, it issues certificates to RIRs as it assigns IP address blocks to them. The certificate contains an extension that specifies the set of address blocks ICANN is allocating to that RIR. Each RIR further assigns portions of its address blocks and issues corresponding certificates to the third tier organizations of the hierarchy. The process continues until it reaches end subscribers. A typical certification path for an address block is similar to the following:

"ICANN→Registry→ISP/DSP... →Subscribers".

The second PKI contains certificates for AS number assignments, as well as identity certificates of organizations, ASes, and BGP speakers. The AS number authentication is similar to address allocation authentication. At the top, ICANN assigns AS numbers to RIRs. Then, each RIR assigns some of its AS numbers and issues certificates to the third tier organizations (also called AS owners). These AS owners, in turn, issue certificates for authenticated ASes. AS owners also issue certificates for BGP speakers; each such certificate binds the router name to an AS number and router ID, testifying that the speaker is authorized to represent certain AS. Typical certification paths in AS number and BGP speaker identification PKI are as follows:

"ICANN→Registry→AS owners→AS numbers"
"ICANN→Registry→ISP/DSP... →BGP speakers".

2.2 S-BGP Attestations

As noted earlier, S-BGP uses two forms of attestations.

For origin authentication, an address attestation (AA) establishes that an AS (the subject in the AA) is authorized by an organization Org_x (the signer of the AA) to announce certain IP blocks of address space [17]. BGP speakers use AAs together with corresponding address allocation certificates to ensure that the origin AS in the route announcement is authorized to originate routes to the IP prefixes.

For path authentication, a *route attestation (RA)* is signed by a BGP speaker to authenticate the existence and position of an AS number in an AS path [17]. Figure 2 demonstrates the structure of RAs. Such attestation is nested: each BGP speaker signs the AS path in sequence, as it joins. That is, first the origin BGP speaker signs the

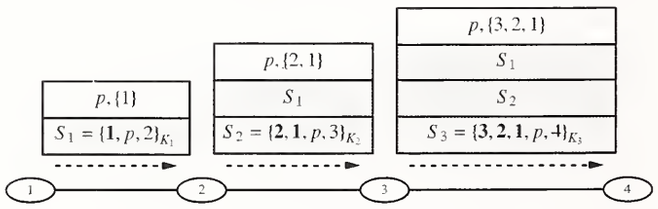


Figure 2: This figure sketches the process of sending route announcements and their route attestations. We have four ASes numbered as 1, 2, 3, and 4. `AS 1` initiates the process by sending announcement `(p, {1})` stating that it owns prefix p and it is reachable. It generates the corresponding route attestation by signing `{1, p, 2}` using its private key K_1. It puts its AS number as the AS path first, then the prefix, then the intended recipient. The other ASes continue this process by signing the new AS path, the prefix, and the indendend recipient. The new attestation is sent together with all previous ones, so that they are effectively chained together. The figure shows the AS path components in bold.

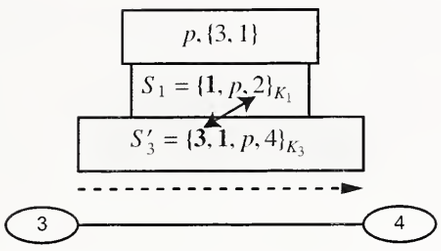


Figure 3: This figure sketches how S-BGP would stop an attempt by AS 3 to forge a route announcement. AS 1 had told AS 2 it would accept messages to p, and AS 2 told that to AS 3. However, AS 3 is trying to strip away 2 and fool AS 4 into believing a fraudulent 2-hop route. However, since AS 1 included the name of AS 2 in its signed statement about that link, AS 4 will detect the forgery.

AS number of the origin autonomous system and the intended receiver (in the form of AS number). The next signer is the receiver of this RA; it computes and signs the concatenation of the new AS path, the prefix, and intended receiver. The process goes on until the entire AS path is signed.

The inclusion of the intended recipient and the prefix in each signature is necessary to prevent against "cut-and-paste" attacks. To continue the earlier example, consider Figure 3. AS 3 is not able use the attestations it has received to forge an attestation for route (p, {1,3}) that AS 4 will accept. To do so, AS 3 would need a signed statement from AS 1 offering to route information to p directly from AS 3. However, the signed link that AS 3 has from AS 1 explicitly specifies that AS 1 links to AS 2, not AS 3. To facilitate validation, BGP speakers send the new RA together with all the previous RAs associated with it. This way, the receiver can authenticate the entire AS path. However, receivers need certificates for BGP speakers to validate these signatures.

2.3 Performance Issues of Path Authentication

Several factors affect the performance of path authentication in S-BGP, given the structural properties of RAs.

First, BGP speakers consume extra CPU cycles when signing and verifying RAs and when handling and validating certificates. Each **Update** message involves one signing operation by each signer and k verification operations by each verifier (where k is the number of RAs for this AS path). Moreover, for each signature verified, the verifier needs to validate the certificate of the alleged signer. Second, RAs and certificates increases message size. Each message with an AS path of length k carries k nested RAs. Finally, to decrease the number of signing/verification operations, one could cache the signed or/and verified routes in memory. Therefore, memory cost becomes another issue.

Researchers have introduced a number of optimizations for S-BGP [16], mainly focusing on caching signed and verified routes and applying DSA pre-computation. These optimizations reduce the computational cost related to cryptographic operations in the cost of extra memory cost and computation complexity.

3 Alternate Signature Approaches

Besides caching, other studies suggest different cryptographic schemes that may potentially reduce the overhead of S-BGP route announcement authentication. We discuss three: signature amortization, sequential aggregate signatures, and origin authentication.

3.1 Signature Amortization

In our previous analysis [27], we proposed *Signature Amortization* (S-A).

Looking at the details of the path authentication process, we observed two important facts. First, BGP speakers verify RAs more often than creating RAs themselves. Hence, making verification faster could potentially decrease the overall computational latency. Second, when the BGP speaker sends identical routes to its neighbors, it has to create distinct RAs; moreover, BGP speakers keep outgoing **Update** messages in buffers and, using MRAI timers, send them in bulk. This bulk send creates the potential for getting more "bang" from each private key operation.

Our S-A scheme exploits these two facts. To speed up the verification processing, we use RSA, since RSA verification is significantly faster than DSA (used by S-BGP). Then, we amortize the cost of signing operation in two steps.

In step one, when a BGP speaker sends the same route announcement to multiple recipients, we collapse it to literally the same announcement—using a bit vector (or a more space-efficient equivalent) to express which of the speaker's peers are the recipients. Thus, the speaker only needs to generate one signature, instead of one for each recipient; the verifier of this RA uses the bit vector to check the intended receiver. To do this, the speaker needs to preestablish an ordered list of its neighbors, and distribute this to potential verifiers; however, we can put this information in the speaker's X.509 certificate, since the verifier needs to obtain that anyway to verify the signature itself.

In step two, when an MRAI timer fires and a BGP

speaker sends the messages accumulated in its out buffers, we have it collect all "unsigned" messages, build a Merkle hash tree [24, 25] on them, and signs the root of the tree—thus generating one signature for all unsigned messages, instead of one for each. A leaf of the tree is the hash of the pair of a route and its recipients. The resulting RA consists of the RSA signature on the root, the the hash path from the root to that leaf, the route, and the recipient bit vector. A verifier of the RA can use these hash values and information in the route announcement to construct the root of the tree correctly. There are trade-offs, however. The verifier needs to perform a few extra hashing operations when verifying a RA, and the message size grows (due to the hash path).

With our S-A approach, we speed up the security operations of S-BGP at the cost of more memory and longer **Update** messages.

3.2 Sequential Aggregate Signatures

Recently, *aggregate signature* schemes have emerged that save signature space when multiple parties need to sign messages [2, 3]. The *sequential aggregate signature (SAS)* scheme by Lysyanskaya et al. [22] combines n signatures from n different signers on n different messages into one signature of unit length. In SAS, each signer, in an ordered sequence, incrementally signs its new message and incorporates it into the aggregate signature σ. A party with knowledge of the n messages, the public keys of the n ordered signers, and the final σ is able to verify that each signer s_i has correctly signed his message M_i and σ is a valid sequential aggregate signature. The major advantage is that the signature of n messages is the same as the length of an ordinary signature. Furthermore, an SAS scheme can be built from RSA, with small modifications, easing implementation.

Applying SAS scheme to path authentication of S-BGP, we generate σ along the AS path similar to nested RAs. Since one aggregate signature is enough to authenticate entire AS path, this scheme shortens message size.

3.3 Origin Authentication

Aiello et al. [1] consider the semantics, design, and costs of origin authentication in BGP, and propose an *OA* scheme.

The authors formalize semantics for IP address delegation, which is similar to the address allocation PKI by S-BGP. The proofs of the IP address ownership establish a tree-like hierarchy rooted at IANA [14]. The next tier are the organizations that receives `/8` IPv4 address blocks directly from IANA. These organizations further delegate sub-block addresses; delegations continue until we reach autonomous systems.

In the OA scheme, the BGP speakers send ordinary BGP **Update** messages together with *origin authentication tags (OATs)*. Each OAT contains a delegation path, a set of *delegation attestations* (one for each edge in the path) and an *ASN ownership proof*. The structure of a delegation attestation is similar to an S-BGP address allocation certificate. The signer authorizes that the subject is delegated some address blocks as recorded in an extension. The ASN ownership proof is a certificate issued by ICANN; it attests that some AS numbers are granted to a particular organization.

The OA scheme considered four possible constructions on delegation attestation. A *Simple Delegation Attestation* contains a signature by an organization on a tuple `(p, org)`, where `p` is the prefix delegated to `org`. An *Authentication Delegation List* combines all `(p, org)` tuples by the same organization into single list and generates one signature. A compromise of these two approaches, an *AS Authentication Delegation List* breaks up the long list into several sublists (each containing the delegation tuples specifying the address delegations made to the same organization and autonomous system) and signing each. An *Authentication Delegation Tree* constructs a Merkle hash tree on an organization's delegation list, and signs the root of the tree. We denote these variations by the terms *OA-Simple*, *OA-List*, *OA-AS-List*, and *OA-Tree*, respectively.

4 Evaluation Methodology

As Section 1 notes, this paper reports research examining the performance impact of public key cryptography and public key infrastructure on BGP security. Section 4.1 describes the metrics we use. Section 4.2 describes the various BGP security approaches on which we take these measurements. Section 4.3 discusses the tools we use to carry out these experiments.

4.1 Performance Metrics

We use a set of metrics to evaluate performance in terms of time and space.

For time, we measure the number of cryptographic operations involved, the resulting CPU cycles, and the BGP convergence time: the time it takes the system to reachieve a stable state after a perturbation, such as a new route announcement, a route withdrawal, or a router reboot. For each security scheme, we compare its convergence time with convergence time that original BGP achieves for the same perturbation. (Given the distributed

nature of BGP, convergence time is very difficult to be predicted using analytical techniques.)

For space, we measure both the message size and the storage cost in memory. Similar to other studies, our experiments relax the current BGP *maximum transfer unit (MTU)* (4096 bytes) limitation, to be able to understand the efficacy of any possible optimization.

4.2 Experimental Approaches

Our previous work evaluated the time impact of S-BGP and S-A on path authentication. We now measure the space impact as well, and both space and time impacts of SAS on path authentication. We measure the time impact of CRL and OCSP revocation schemes on fully optimized S-BGP.

We also examine the origin authentication scheme of Aiello et al. We measure time and space impacts of all four variations, as well as the time impact of CRL and OCSP revocation on the OA-AS-List variation (since it's the closest to S-BGP origin authentication).

4.3 Simulation

We use discrete-event simulation to understand the performance of BGP origin and path authentication schemes in a large-scale environment. As with our earlier work, our experiments uses SSFNet [5, 28], a discrete-event simulator that provides a comprehensive model of basic BGP operations [31]. Our earlier work added hooks for variants of processing models of BGP security schemes [27].

Throughout this study, we evaluate security schemes in the same network topology and same BGP activity settings. We use a 110-AS topology, with one operating BGP speaker per AS. For modeling simplicity, each BGP speaker announces two prefixes. In our model, each AS also possesses *virtual BGP speakers* that don't actually run a simulated BGP protocol. We use the number of such BGP speakers to represent the size of an AS; its size affects the time it takes for one **Update** message to be propagated through an AS.

We use the public data provided by RouteViews project [33] to generate a graph of AS connectivity of the Internet, then reduce the size to 110 ASes using a collapsing procedure. This reduced graph still preserves certain macroscopic properties [6] seen on the entire Internet. Moreover, we incorporate our estimation of route filtering policies into the topology using a method, similar to the one proposed in [7].

During normal BGP activities, we let one BGP speaker crash and reboot. We evaluate the performance of the entire system during router rebooting process. The workload on BGP speakers could be much higher than normal BGP activities, since the rebooting BGP speaker receives routing table dumps in a short period of time from each its peers, via a large amount of route announcements. To maximize the effects, we let the rebooting BGP speaker to be the one with the most peers.

Besides the common settings, we also have specific parameters for each of the security schemes. Table 1 summarizes the benchmarks and measurement numbers we use in our simulation. The running time benchmarks of cryptographic operations are from OpenSSL [29] library. For those algorithms not directly implemented by the library (such as DSA pre-computation, SAS aggregate signing and SAS aggregate verification), we decompose the involved operations and sum up the benchmarks of each step as an estimation. In addition, the numbers are normalized to a 200MHz CPU, which is a common CPU speed of BGP routers. We use a real system to measure and estimate latencies of processing plain **Update** messages, of sending a OCSP request and receiving a response, and of fetching CRLs. To take into account other factors that could potentially affect the numbers, the simulation decides these values by uniform distribution within certain ranges. S-BGP studies [16, 18] give the numbers for sizes of S-BGP certificate and attestations.

5 Path Authentication Performance Analysis

We compare performance impact of S-BGP, S-A, and SAS. We examine the performance on signatures and PKIs respectively. This section gives detailed results and analysis.

5.1 Signatures and Verifications

Before examining details, we enumerate our major findings on convergence time, message size, and memory cost in Table 2. S-BGP performs badly on convergence time, but is fairly efficient on memory cost. S-A outperforms the other two on convergence time, but is significantly more costly than the other two schemes on message size and memory cost. SAS generates the shortest **Update** messages, but results in the longest convergence time.

We also studied the efficacy of strategies for caching validated (or generated) signatures. In simulation experiments, we explored S-BGP with several variations of DSA optimizations. For the presentation of experiment results, we use *cDSA* to denote S-BGP with caching, *pDSA* to denote S-BGP using DSA pre-computation, and *cpDSA* for S-BGP with both optimizations. In our model, these

	SHA-1 hash	Attestation	S-BGP X.509 Certificate	Identifier
Length (bytes)	20	110	600	4

	RSA	DSA	DSA (p-c)	SAS
Verify Time (*ms*)	2.5	31.0	31.0	2.5
Sign Time (*ms*)	50.0	25.5	0.015	50.0
Signature Length (bytes)	128	40	40	128

	OCSP request	CRL fetching
Operation Latency (second)	0.5–1.0	0.5–1.0

Table 1: Constants and benchmarks used for simulation. RSA, DSA, and SAS algorithms are based on 1024-bit keys.

	Convergence	Message Size	Memory
S-BGP	long	moderate	best
S-A	shortest	worst	worst
SAS	longest	best	best

Table 2: Performance rankings for the path authentication schemes we examined

caching strategies store both validated signatures and generated signatures; we use $10\mu s$ (with a uniformly distributed delta of $5\mu s$) to model the lookup time. The S-A scheme will not speed up by caching hash trees with signatures, because the trees, and hence the signatures, are constantly changing even for the same route announcement (since the trees depend on the context of what else is being signed at that time). Therefore, we only examined S-A scheme without caching, when studying processing latency and convergence time. However, we model a variation for S-A caching merely to understand potential memory cost it might result. Finally, all the experiment results are average numbers from 20 runs of the simulation. The standard deviation is less than 5%.

Time We examine the convergence time by looking at the counts of cryptographic operations. Figure 4 through Figure 7 summarize the results. All the schemes without caching optimization generate relatively the same number of signature verifications, proportional to the total number of AS numbers encountered in AS paths in route announcements. Similarly, caching optimization by each of the schemes achieves relatively the same number of saving percentage.

The story of signing operations remains the same for S-BGP and SAS schemes. The S-A scheme can dramatically save as many as 98.3% of signing operations. Our experiments show that the average hash tree size by S-A is about 60.1, indicating that S-A is able to amortize the cost of 60 signing operations into only one signing and a few hashing operations.

The CPU cycles and convergence time reflect this difference in the number of cryptographic operations. We sum up the total CPU time on all BGP speakers, and also track the portion consumed by cryptographic operations, including signing, verification, and hashing ("crypto," in Figure 6). SAS requires 1723.2 seconds extra time for aggregate signing and aggregate verification, which is much shorter than 4002.2 seconds by S-BGP. This difference results mainly because aggregate verifications are much faster than DSA verifications. Caching optimization to S-BGP and SAS scheme can significantly reduce total CPU time. Although S-BGP (pDSA) uses much faster signing operations, the net speed-up is limited, because the number of verification operations dominates the number of signing operations. Compared with S-BGP and SAS, S-A improves both aspects—fewer signing operations and faster verifications. Our experiments confirm it is the most efficient on CPU cycles.

Next, we look at convergence time. Among the three major schemes, SAS is the worst. Compared with plain BGP, it converges three times slower. S-BGP comes next, with a slowdown of about 2.3 times. Even with optimizations, S-BGP still takes 46.05% longer to converge. (Our previous work [27] showed better S-BGP numbers, but that turned out to be due to a bug in our simulation code.)

Such slowdowns lead to routing fluctuations that create

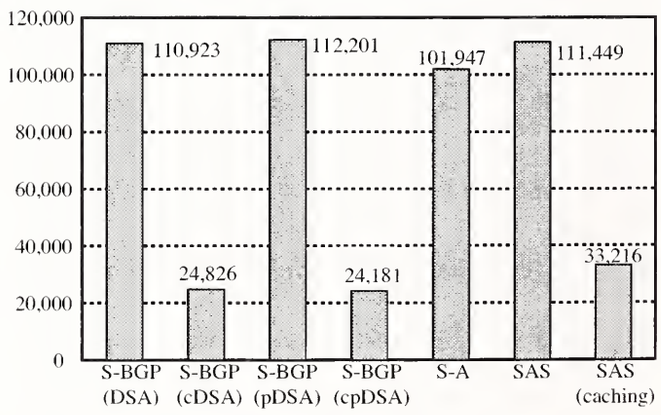


Figure 4: Verification operations in path authentication

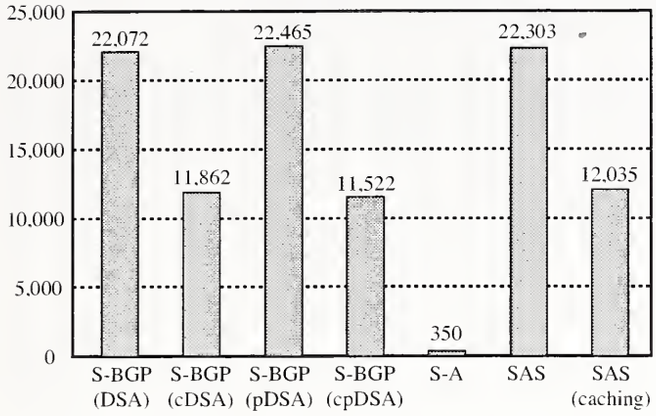


Figure 5: Signing operations in path authentication

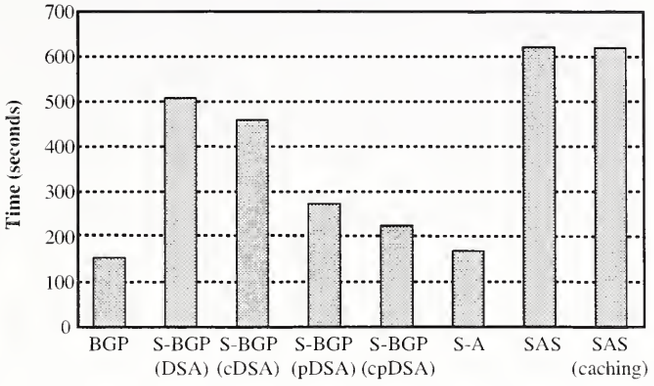


Figure 7: Convergence time in path authentication

all sorts of network problems, such as increased packet loss rates, increased network latencies, increased network congestion, and even disconnections. In our experiments, router reboots by BGP even without any security protection already cost the network 153.7 seconds to converge. Extending the period to another several minutes is not a good option.

Fortunately, our S-A scheme increases the convergence only by a few seconds, with no burden on caching large amount of data in memory.

Our experiments revealed that, counter-intuitively, convergence time is not proportional to the CPU time spent by BGP speakers. In fact, the data suggests that the latencies in the message sending process (therefore, signing overhead) could be the dominant factor. For instance, if we consider only the CPU time consumed by signing operations, SAS costs the most, about 92% of the total CPU time, which could explain why SAS is the slowest on convergence. One might reach a similar conclusion from the inconsistency between S-BGP (cDSA) and S-BGP (pDSA). Although S-BGP (pDSA) requires more CPU cycles, almost all of these CPU cycles are spent for signature verifications. As the result, it converges faster.

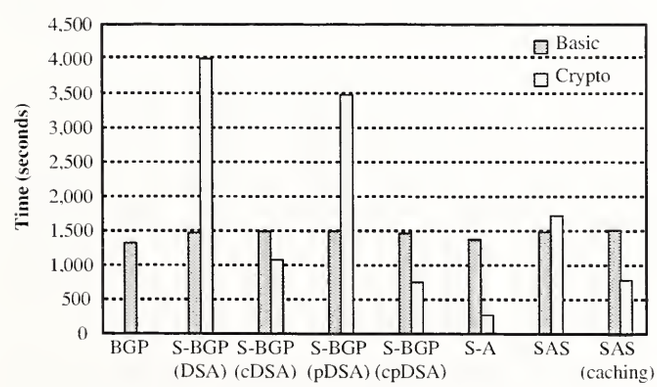


Figure 6: Total CPU time in path authentication

Memory Figure 8 shows the average memory cost and maximum memory cost for individual BGP speakers. We start with a baseline of 9KB memory at each speaker, for plain BGP. On average, S-BGP increases this requirement to 112.25KB, SAS to 121.95KB, and S-A to 314.38KB. We assume that BGP speakers record all cached routes in memory (e.g., RAM). In the simulation, we count the bytes of the IP prefix, AS path, and related cryptographic data structures (signatures, hash values, and bit vectors).

Frequent changes of hash trees prevent S-A from saving processing latency by caching signatures. To explore the memory impact of caching, we tried letting S-A cache more stable data, the leaf information: **Update** messages, signatures, and associated bit vectors (assuming neighboring relationship between ASes stays unchanged during simulation). For this experiment, we dispensed with hash trees, but the resulting convergence time of a variant that used hash trees and this caching would not be worse than

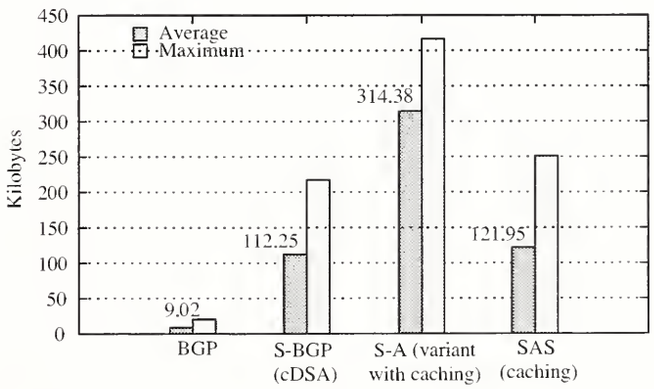


Figure 8: Comparison of memory costs for caching. The S-A scheme in this comparison is a variant that does not use hash trees, instead, amortizing signing costs by using bit vectors only. The bit vectors are relatively stable, which allow the BGP speakers to cache them with the routes to reduce duplicated verifications.

	BGP	S-BGP	S-A	SAS
Average message size (bytes)	36.09	318.61	1107.08	184.29
Increase		8.83	21.57	5.11
Maximum message size (bytes)	42.6	527.7	1915.4	191.2
Increase		13.77	47.75	4.40

Table 3: Message size. The increase numbers are based on the message size by original BGP.

the numbers shown in Figure 7 and 8.

One of the leading factors that affect this memory cost is signature length. Here, S-BGP outperforms S-A because a DSA signature is much shorter than a RSA signature (e.g., 40 bytes vs. 128 bytes). Secondly, SAS is able to save memory by caching only one signature for an AS path of any length. Even with RSA signatures, SAS is as efficient as S-BGP.

Although not shown in Figure 8, edge routers consume the most memory for caching routes, statistically. We posit two reasons. First, as a pure customer in the network, an edge router may receive more route announcements than the ones in the core of the network. Second, and most importantly, the AS paths recorded by edge routers are significantly longer, so these routers will cache more signatures.

In ongoing work, we are exploring using cryptographic hashing to further reduce cache size.

Update Message Size SAS produces one signature for an AS path; it wins the competition on message size. S-BGP is next, again, because of shorter signature length. Our experiment results, shown in Table 3, confirm that S-A generates the longest messages. For both S-BGP and S-A, number of signatures in messages grows as the length of path increases.

For SAS, since each **Update** message contains only one aggregate signature for the entire AS path, the maximum message size is close to the average size. On the other hand, the longest **Update** message for the S-BGP and S-A schemes is about two times as long as average messages.

Our experiments measured shorter message sizes than the number measured in the Internet, because we only considered the fields (AS path, signatures, hashes, and bit vectors) that would vary between the schemes. Since the ignored portions are the same for each of the schemes, the simulation still results in a fair comparison of the message size.

5.2 Certificate Revocation

Bringing the PKI one step closer to reality requires considering the costs of checking the validity of a signer's certificate, when verifying a signature. Recall that BGP speakers use their private keys to sign and create RAs on route announcements. We use simulation to model the case that BGP speakers validate BGP speakers' public keys in certificates before using them to verify RAs.

In our revocation simulation, we assume that the 110 ASes belong to different organizations (also called PKI *domains*), with each organization having its own CA issuing certificates for that organization's BGP speakers. Each PKI domain has a repository of certificates and CRLs, offered by an LDAP server. When we model revocation by OCSP, we assume an organization has an online OCSP responder; when we model CRLs, we assume the organization's LDAP server also offers CRLs.

We then examine the convergence time for S-BGP with all optimizations, using OCSP or CRLs for certificate validation. The OCSP approach provides fresh information of certificate status, at the cost of network and processing latencies. The CRL approach is less aggressive: the verifier downloads CRLs periodically, checks certificate status with these local copies, and (when the local copies expire) get fresh CRLs from the appropriate repositories via the LDAP protocol.

For simplicity, we assume that BGP speakers can validate OCSP responses and fetched CRLs by verifying signatures on them. In other words, we do not model the process of discovering trust path for them. The rest of this section discusses and compares the performance impact that checking certificate status has on S-BGP.

OCSP The model we use to study OCSP is close to typical PKI practice in the real world. In a practical PKI, one or more OCSP responders connect to a certificate database operated by local CAs to serve the status information of the certificates issued by local CAs. Optionally, the responders can set up SSL connections to enhance privacy for the client.

The OCSP response is a signed data structure that contains the real-time status of a requested certificate. OCSP introduces latencies, from setting up an SSL connection, from network delays, from real-time signing, and from signature verification. According to measurements we made with real-world OCSP implementations, the latency of one round is about 0.5–1.0 seconds, the majority of which is from network latency.

If a client has multiple certificates to validate, it can send OCSP requests in sequence or in parallel. A proxy, such as a *Certificate Arbitrator Module* (CAM) [37], can

Protocol	# Ann.	# Vrf.	# Sign	# OCSP Rqst.	Basic CPU (s)	Crypto CPU (s)	Convergence (s)
BGP	19571.8	–	–	–	1310.6	–	153.7
S-BGP (cpDSA)	21898.9	24180.6	11521.9	–	1464.1	755.4	224.4
Sequential OCSP	22542.9	113859.9	11663.2	89912.5	1501.7	70990.2	2720.4
Parallel OCSP	21707.8	110429.3	11290.5	87004.0	1448.5	3971.0	344.3

Table 4: Performance of validating certifi cates using OCSP for S-BGP path authentication

contact multiple OCSP responders throughout the network and send requests in parallel for the client. In our simulation, we model both sequential and parallel cases.

Table 4 shows that checking certificate status using OCSP for S-BGP is intolerably expensive. Sending sequential OCSP requests is an especially bad idea. We put performance numbers of BGP and S-BGP (cpDSA) in the table for comparison, and show both the basic CPU time, the processing latencies related to cryptographic operations, the latencies by OCSP requests and responses, and network latency in between. Even the resulting convergence time for parallel OCSP requests is 344.3 seconds.

CRLs For CRLs, we assume that each BGP speaker has a local cache of CRLs. Since signature verification requires an up-to-date copy of the CRL from the relevant CA, the BGP speaker pays the price of fetching and validating fresh ones before verifying RAs, if some CRLs are missing or expired.

To evaluate the cost of fetching CRLs, we let BGP speakers have a certain fraction of the CRLs in their local cache be expired, and then measure the resulting convergence time. The experiments assume that it costs 0.5–1.0 seconds on average for BGP speakers to fetch a CRL. We also assume that CRLs are valid for 12 hours.

Figure 9 shows the measurement data from simulation. It is clear that more expired CRLs cause the convergence times to increase linearly. These times range from 224.4 seconds to 287.7 seconds. Hence, even with all CRLs expired, validating certificates against CRLs is still a more efficient approach than OCSP, which costs 344.3 seconds to converge with the fast option, parallel OCSP requests.

6 Origin Authentication

Our approach to studying origin authentication is similar to the approach we took for path authentication. We first look at the performance impact of signatures and verifications, then examine the certificate validation cost on top of that. We add one model in simulation for experiments—the approximated address delegation graph. As mentioned earlier, the semantics of IP address delegation start from IANA. Aiello et al. [1] expressed IP addresses of the Internet using such semantics. Using publicly available Internet measurements, these researchers generated an approximated address delegation graph, a tree rooted at IANA. The structure is very similar to the address allocation PKI by S-BGP (not surprising, since it essentially solves the same problem).

For each prefix in route announcements, the **Update** message should carry an address delegation path for authentication. The scheme of Aiello et al. [1] uses in-band address delegation attestations carried in **Update** messages, because these attestations are much smaller in size than the S-BGP address allocation certificates. We use simulation to re-visit this issue.

We model address delegation using this approximated complete graph of the Internet and size it down so that it is suitable for our 110-AS simulated network. In practice, ASes could announce many prefixes, each of which could have its own address delegation path in the graph. Our simulation model is much simpler; each AS only announces two prefixes. We add randomness in the model to capture the diversity of the real world. First, we put the address delegation graph into the configuration of simulation, so that BGP speakers can recognize all delegation paths for each origin AS. Next, we let BGP speakers randomly choose a path for the prefix based on the origin AS. We limit the path length to seven (since address delegation paths are reported to be no longer than 4-5, in practice).

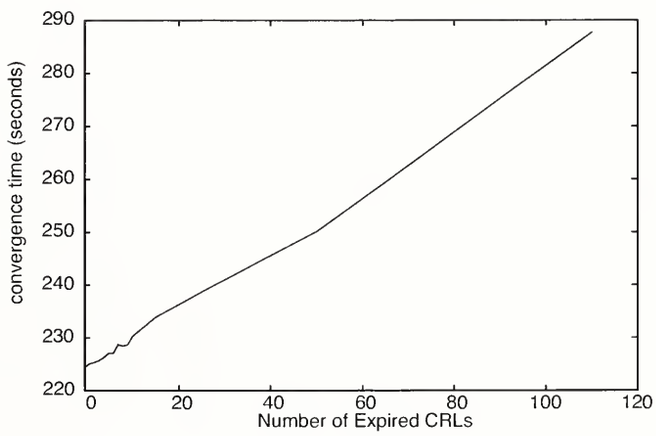


Figure 9: Convergence times by S-BGP using CRLs to validate certifi cates.

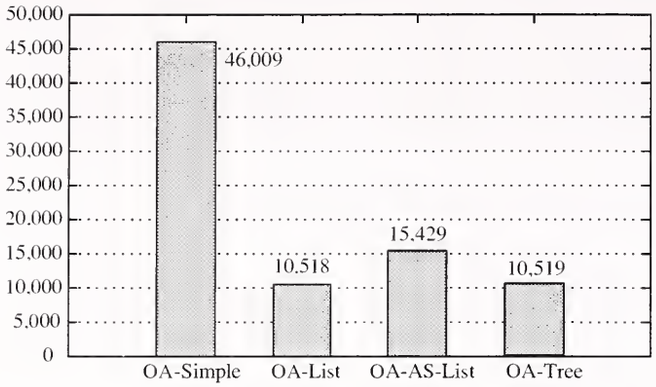


Figure 10: Number of verification operations by OA address delegation attestation constructions.

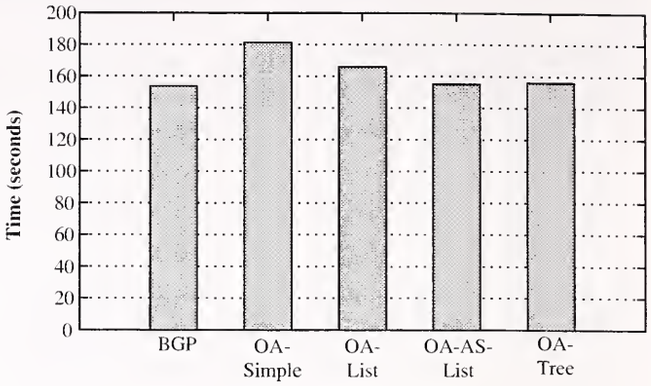


Figure 12: Convergence time by OA address delegation attestation constructions.

This randomly chosen path determines what address delegation attestations are involved.

6.1 Signatures and Verification

Time Figure 10 through Figure 12 show the processing latency. We assume that organizations prepare the delegation attestations offline; the simulation only counts signature verifications and hashing latencies accordingly. The OA-List and OA-Tree approaches greatly reduce the number of signature verifications required. Compared with path authentication schemes, the increase of convergence time by all delegation attestation constructions are manageable. This result, again, implies the verification overheads are a minor factor to convergence time compared to signing operations.

The resulting convergence time of OA confirms the conclusion made by Aiello et al. [1]—the efficiencies afforded by OA designs make in-band delegation attestation verification possible. However, as Aiello et al. also mention, in-band delivery of delegation attestation is susceptible to replay attacks, unless we introduce short-lived tokens or make delegation attestations short-lived. Thus, a trade-off exists between the period of vulnerability and the overhead of administration and computation.

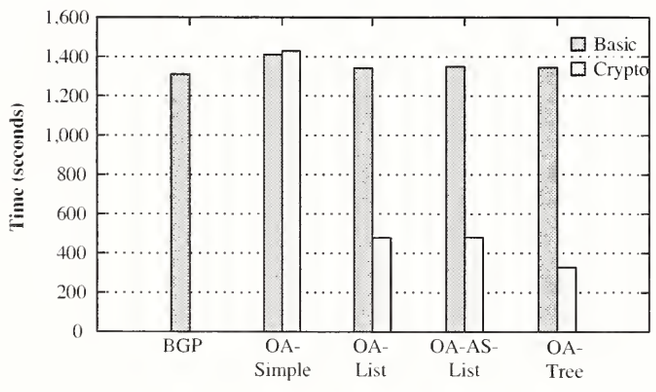


Figure 11: Total CPU time by OA address delegation attestation constructions.

Memory We let BGP speakers cache verified attestations and associated prefixes; we then measure the average memory cost and message size. Table 5 shows that the OA-List scheme is more costly than other schemes, mainly because the list construction produces extremely long delegation attestations. In the approximated address delegation graph, the average number of delegations made by organizations is about 56.96. Moreover, about 16 organizations make 80% of the address delegations. Obviously, this graph has high connectivity and the delegations are concentrated on very small portion of organizations. These features are the reason why the AS-List approach can produce long lists of prefixes in address delegation attestations. According to Figure 10, the AS-Tree approach handles the least number of signatures; however, its memory cost and message size are worse than OA-AS-List, mainly because the AS-Tree approach involves hash values, which are much longer than organization identifiers.

6.2 Certificate Revocation

The above analysis shows that the OA-AS-List attestation construction is fairly efficient. It is the most efficient one on memory cost and message size, and it does not put significant pressure on BGP processing and convergence. In fact, the OA-AS-List construction—the delegation list grouped by different delegatees—is very similar to the design of address allocation certificates of S-BGP. Thus, we next consider the case that BGP speakers send S-BGP ad-

Attestation Constructions	OA-Simple	OA-List	OA-AS-List	OA-Tree
Storage for Attests. (KB)	42.80	666.27	13.23	30.22
Message Size (Bytes)	496.97	36293.37	575.35	1029.24

Table 5: Average memory cost and message size by OA address delegation attestation constructions.

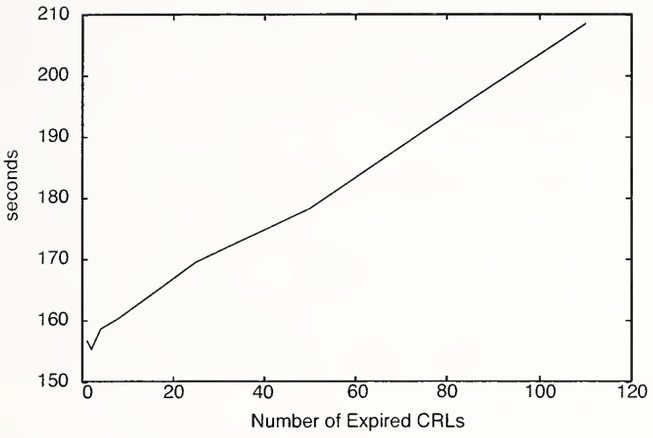


Figure 13: Convergence times by origin authentication using CRLs to check certifi cate status.

dress allocation certificates, instead of delegation attestations in **Update** messages. In other words, the sender encloses the complete certification chain for the verification of address attestation (AA). We assume that each speaker sets ICANN and the CA in their local PKI domain as its trust anchors.

Again, bringing this PKI one step closer to reality requires considering the costs of checking the validity of the certificates. We consider each approach in turn.

OCSP As before, we first consider OCSP, both in sequence and in parallel. Table 6 shows the experiment results on processing latency. The most important conclusion we can draw is that, as for path authentication, OCSP processing for origin authentication can greatly slow down the BGP convergence. For either part of BGP route authentication, using OCSP to validate real-time certificate status does not appear to be feasible in practice.

CRLs Again, we carried out experiments assuming different sets of CRLs expire at the routers, and examined performance. Figure 13 shows the results. The curve is similar to the convergence time by path authentication with CRL fetching. The convergence time is relatively unaffected if each of the BGP speakers needs to fetch fewer than eight CRLs during rebooting.

6.3 Certificate Distribution

In addition to processing latency and convergence time, the experiments also measure message size. Carrying certificates in **Update** messages would require 2KB on average. The maximum message size is about 4KB. Given the BGP message MTU, carrying these certificates does not appear to be feasible in practice. On the other hand, if BGP speakers record all certificates locally, our simulation shows that certificates consume about 6KB storage on each BGP speakers, on average. The relatively small scale of the simulated network prevents us from directly inferring potential storage issues in the real world. IP address allocations, AS number assignments, and router assignments on the full Internet produce much more certificates. The CIDR BGP report from AS1221 (Telstra) [4] shows that there are 181,031 active BGP entries in a routing table. To validate ownerships of these prefixes, we need roughly the same number of address allocation certificates. Besides, this report also concludes that there are about 18,233 unique ASes and 50,000 organizations. Considering both PKIs by S-BGP, each BGP speaker needs about 190MB in total to store all certificates.

7 Related Work

The performance studies in [16, 18] offer detailed discussions on deploying S-BGP in the real world. The authors collected a variety of data sources to analyze S-BGP's performance impacts on BGP processing, transmission bandwidth, and routing table size. These studies concluded that the memory requirements of holding route information and related cryptographic data are a major obstacle to deployment of S-BGP. Unlike our work, all of the discussions are based on static measurement of BGP.

The origin authentication study by Aiello et al. [1] designed a simulator, *OASim*, to model the operations of a single BGP speaker. This simulator accepts timed BGP **Update** streams and computes the costs associated with the validation and storage of the related origin authentication proofs. The simulation results show that in-band distribution of origin authentication proofs is possible. Our simulation is more powerful than OASim in that we model and simulate a network and study the convergence time.

Our previous study [27] used a packet-level detailed

Protocol	# Ann.	# Vrf.	# Attest.	# OCSP Rqst.	Basic CPU (s)	Crypto CPU (s)	Convergence (s)
BGP	19571.8	–	–	–	1310.6	–	153.7
OA-AS-List	20131.2	15429.1	10364.1	–	1349.7	480.4	155.1
Sequential OCSP	22800.5	73586.7	5071.0	68515.65	1522.1	53665.7	2420.9
Parallel OCSP	22408.6	72635.2	5071.2	67564.00	1494.8	19060.2	938.7

Table 6: Convergence impact of OCSP on in-band address attestation.

simulation model of BGP to understand the processing overhead by S-BGP. We discovered that, due to public key cryptography, S-BGP is expensive on operational latency and thus greatly increases convergence time. We further proposed a more efficient scheme (signature amortization, S-A) for BGP path authentication. Our simulation experiments conclude that the new approach has minimal impact on BGP convergence.

There are also other studies on more efficient mechanisms for securing BGP. One challenge in the adoption of any inter-domain routing security solution is its integration with existing infrastructure. In the *Inter-domain Routing Validation (IRV)* project [8], participating ASes host servers called IRVs. Each IRV maintains a consistent corpus of routing data received and advertised. Remote entities (e.g., routers, other IRVs, application) validate locally received data by querying source AS IRVs using an out-of-band (and potentially secure) protocol. This approach has the advantage that the query responses can be tailored to the requester for optimization or access control.

A recent effort that attacks the scalability issue of S-BGP is *psBGP* [40]. The major goal is to increase practicability of security solutions on BGP. The psBGP protocol contains four main components—authentication of AS numbers, authentication of IP prefix ownership, authentication of BGP speakers, and integrity of AS path. Essentially, this proposal combines aspects of S-BGP and soBGP.

Besides public key cryptography, there are efforts on securing BGP using symmetric key algorithms [9, 13, 42]. These proposals are more efficient on the operational latency, but require more storage, loose time synchronization, and complex key-pair pre-distribution.

Subramanian et al. [36] proposed the *Listen* and *Whisper* protocols to address the BGP security problem. The Listen protocol helps data forwarding by detecting "incomplete" TCP connection; the Whisper protocol uncovers invalid route announcements by detecting inconsistency among multiple update messages originating from a common AS. The Listen and Whisper approach dispenses with the requirement of PKI or a trusted centralized database, and aims for "significantly improved security" rather than "perfect security."

8 Conclusions

Implementation details of securing BGP have significant impact on BGP's behavior, and on the capacity of routers to actually use the algorithms. BGP's detailed time and memory consumption is too complex to analyze purely with mathematics, and so we turn to large-scale discrete-event simulation to examine the impacts of cryptographic operations and standard PKI certificate validation schemes on recent proposals to secure BGP.

We compare several major security proposals with S-BGP. Our simulation results have shown that it is possible to apply more efficient cryptographic operations to improve the performance in terms of convergence time, message size, or storage costs. Tradeoffs exist. Different proposals have their own strengthens and weakness. In particular, Signature Amortization achieves fast convergence at the cost of longer message size and more memory. Sequential Aggregation Signatures can decrease the message size, but slowing down the BGP convergence significantly. The Origin Authentication scheme can achieve instant origin proofs with in-band distribution of attestations, at the cost of exposing vulnerabilities to attackers.

We also analyzed the impacts of standard certificate revocation/validation mechanisms. The OCSP approach greatly slows down convergence. On the other hand, if BGP speakers rely on CRLs for certificate validation, the extra overheads by CRL handling operations are insignificant to affect convergence. Of course, such choices trade performance with security.

Besides BGP routing system, a variety of other large-scale distributed systems assume an underlying PKI—but neglect to consider its performance impact. Understanding the impact of the underlying PKI systems is a challenging task. In the future, we plan to analyze broader issues of PKI design and deployment that satisfy the security and performance requirements by these large-scale distributed systems and applications.

In ongoing work, we are also exploring new aggregated path authentication protocols that further improve performance [43].

Acknowledgments

The authors are grateful to Patrick McDaniel, Kevin Butler, William Aiello, Steve Kent, Scot Rea, B. J. Premore, and Hongsuda Tangmunarunkit for their valuable suggestions. This research has been supported in part by Sun, Cisco, the Mellon Foundation, NSF (CCR-0209144, EIA-9802068), AT&T/Internet2 and the Office for Domestic Preparedness, Department of Homeland Security (2000-DT-CX-K001). This paper does not necessarily reflect the views of the sponsors.

References

[1] William Aiello, John Ioannidis, and Patrick McDaniel. Origin Authentication in Interdomain Routing. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pages 165–178. ACM Press, October 2003.

[2] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. A Survey of Two Signature Aggregation Techniques. *RSA CryptoBytes*, 6(2):1–10, 2003.

[3] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Proceedings of Eurocrypt 2003*, number 2656 in LNCS, pages 416–432. Springer-Verlag, 2003.

[4] CIDR BGP Reports from AS1221 (Telstra), October 2004. `http://www.cidr-report.org/as1221/`.

[5] James Cowie, David Nicol, and Andy Ogielski. Modeling the GIobal Internet. *IEEE Computing in Science and Engineering*, 1(1):42–50, Jan.–Feb. 1999.

[6] Michalis Faloutsos, Petrof Faloutsos, and Christos Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proceedings of ACM SIGCOMM'99*, pages 251–262. ACM Press, 1999.

[7] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 9(6):733–745, December 2001.

[8] Goeffrey Goodell, William Aiello, Timothy Griffin, John Ioannidis, Patrick McDaniel, and Aviel Rubin. Working around BGP: An Incremental Approach to Improving Security and Accuracy in Interdomain Routing. In *The 10th Annual Network and Distributed System Security Symposium*, San Diego, California, February 2003.

[9] Michael Goodrich. Efficient and Secure Network Routing Algorithms. provisional patent filing, `http://www.cs.jhu.edu/~goodrich/cgc/pubs/rout-ing.pdf`, January 2001.

[10] Ramesh Govindan and Anoop Reddy. An Analysis of Internet Inter-Domain Topology and Route Stability. In *Proceedings of INFOCOM 1997*, pages 850–857, April 1997.

[11] Timothy G. Griffin and Gordon Wilfong. An Analysis of BGP Convergence Properties. In *Proceedings of SIGCOMM 1999*, pages 277–288, August 1999.

[12] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC3280, `http://www.ietf.org/rfc3280.txt`, April 2002.

[13] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. SPV: Secure Path Vector Routing for Securing BGP. In *Proceedings of SIGCOMM 2004*, pages 179–192. ACM Press, August 2004.

[14] IANA: Internet Assigned Numbers Authority. `http://www.iana.org`.

[15] Internet corporation for assigned names and numbers. `http://www.icann.org`.

[16] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure Border Gateway Protocol (S-BGP) – Real World Performance and Deployment Issues. In *The 7th Annual Network and Distributed System Security Symposium (NDSS'00)*, San Diego, California, February 2000.

[17] Stephen Kent, Charles Lynn, and Karen Seo. Secure Border Gateway Protocol. *IEEE Journal of Selected Areas in Communications*, 18(4):582–592, April 2000.

[18] Steve Kent. Securing the Border Gateway Protocol: A Status Update. In *Seventh IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, October 2003.

[19] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet Routing Convergence. In *Proceedings of SIGCOMM 2000*, pages 175–187, August 2000.

[20] Craig Labovitz, Abha Ahuja, and Farnam Jahanian. Experimental Study of Internet Stability and Wide-Area Backbone Failures. In *Proceedings of the International Symposium on Fault-Tolerant Computing*, June 1999.

[21] Craig Labovitz, Abha Ahuja, Roger Wattenhofer, and Srinivasan Venkatachary. The Impact of Internet Policy and Topology on Delayed Routing Convergence. In *Proceedings of INFOCOM 2001*, pages 537–546, April 2001.

[22] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential Aggregate Signatures from Trapdoor Permutations. In *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 74–90. Springer-Verlag, 2004.

[23] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, and Randy H. Katz. Route Flap Damping Exacerbates Internet Routing Convergence. In *Proceedings of SIGCOMM 2002*, August 2002.

[24] R. Merkle. Protocols for Public Key Cryptosystems. In *Proc 1980 Symposium on Security and Privacy, IEEE Computer Society*, pages 122–133, April 1980.

[25] R. Merkle. A Certified Digital Signature. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1990.

[26] S. Murphy. BGP Security Vulnerabilities Analysis. Internet-Draft, draft-murphy-bgp-vuln-00.txt, NAI Labs, February 2002.

[27] David M. Nicol, Sean W. Smith, and Meiyuan Zhao. Evaluation of Efficient Security for BGP Route Announcements using Parallel Simulation. *Simulation Practice and Theory Journal, special issue on Modeling and Simulation of Distributed Systems and Networks*, 12(3–4):187–216, July 2004.

[28] Andy T. Ogielski and James H. Cowie. SSFNet: Scalable Simulation Framework - Network Models. `http://www.ssfnet.org`. See `http://www.ssfnet.org/publications.html` for links to related publications.

[29] OpenSSL: The Open Source toolkit for SSL/TLS. `http://www.openssl.org`.

[30] Dan Pei, Xiaoliang Zhao, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. Improving BGP Convergence through Consistency Assertions. In *Proceedings of INFOCOM 2002*, June 2002.

[31] Brian Premore. *An Analysis of Convergence Properties of the Border Gateway Protocol Using Discrete Event Simulation*. PhD thesis, Dartmouth College, June 2003.

[32] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC1771, `http://www.ietf.org/rfc1771.txt`, March 1995.

[33] The Route Views Project. `http://www.antc.uoregon.edu/route-views/`.

[34] Aman Shaikh, Anujan Varma, Lampros Kalampoukas, and Rohit Dube. Routing Stability in Congested Networks: Experimentation and Analysis. In *Proceedings of SIGCOMM 2000*, pages 163–174, August 2000.

[35] B. Smith and J.J. Garcia-Luna-Aceves. Efficient Security Mechanisms for the Border Gateway Routing Protocol. *Computer Communications (Elsevier)*, 21(3):203–210, 1998.

[36] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy H. Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proceedings of First Symposium on Networked Systems Design and Implementation (NSDI 2004)*, March 2004.

[37] MitreTek Systems. Certifi cate Arbitrator Module. `http://cam.mitretek.org/cam/`.

[38] Hongsuda Tangmunarunkit, Ramesh Govindan, Scott Shenker, and Deborah Estrin. The Impact of Routing Policy on Internet Paths. In *Proceedings of INFOCOM 2001*, pages 736–742, April 2001.

[39] Iljitsch van Beijnum. *BGP: Building Reliable Networks with the Border Gateway Protocol*. O'Reilly, 2002.

[40] Tao Wan, Evangelos Kranakis, and P.C. van Oorschot. Pretty Secure BGP (psBGP). In *The 12th Annual Network and Distributed System Security Symposium (NDSS'05)*, San Diego, California, February 2005.

[41] Russ White. Architecture and Deployment Considerations for Secure Origin BGP (soBGP). IETF Internet Draft `http://www.ietf.org/internet-drafts/draft-white-sobgparchitecture-00.txt`, May 2004.

[42] K. Zhang. Efficient Protocols for Signing Routing Messages. In *The 5th Annual Network and Distributed Systems Security Symposium (NDSS'98)*, San Diego, California, March 1998.

[43] Meiyuan Zhao, Sean W. Smith, and David M. Nicol. Aggregated Path Authentication for Efficient BGP Security. Technical Report TR2005-541, Computer Science Department, Dartmouth College, May 2005.

Observations from the Deployment of a Large Scale PKI

Rebecca Nielsen
Booz Allen Hamilton

1 Background

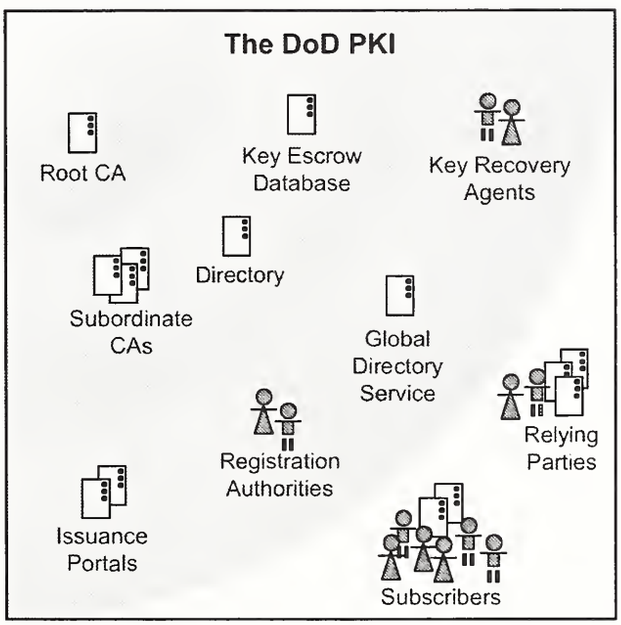
The United States Department of Defense (DoD) has been investigating the use of public key technology to help meet its information assurance goals since 1997. The DoD implemented a pilot Public Key Infrastructure (PKI) in 1998, and began a mass rollout of the current DoD PKI in 2000. Since then, the DoD has successfully issued digital certificates on Common Access Cards (CAC) to over 85% of its 3.5 million user population. While the deployment of the DoD PKI has not always been smooth, the issuance of digital certificates has been one of the first truly enterprise-wide standard technology implementations within the DoD.

This paper provides insight into some of the technology and organizational lessons learned in deploying the world's largest PKI from the perspective of a DoD contractor.

2 Managing the "I" in PKI

2.1 The DoD PKI Architecture

The DoD PKI consists of a single Root Certification Authority (CA) and multiple subordinate CAs. The Root CA only issues subordinate CA certificates. Subordinate CAs issue five types of certificates: identity, signature, encryption, component, and code signing. Identity and signature certificates can be used to authenticate to applications or digitally sign forms or email messages. Because many DoD email addresses change when individuals move from one location to another, the DoD issues the primary identity certificate with no email address. The signature and encryption certificates contain email addresses to support Secure/Multipurpose Internet Mail Extensions (S/MIME) version 2 and 3. New email certificates can be issued based on the presentation of a valid identity certificate. Component certificates are issued to web servers and other devices. Code signing certificates are issued to specific entities within the DoD that approve mobile code.



Other DoD PKI core components include the internal directory servers and the Key Escrow Database (KED). All private keys associated with encryption certificates are escrowed prior to the issuance of the certificate.

The DoD PKI supports two interfaces for certificate issuance, hardware and software. Hardware certificates are issued on the CACs to all DoD personnel. The CAC is a Java smart card that has been validated against Federal Information Processing Standard (FIPS) 401 Level 2 requirements. The Java card was selected both to allow for the inclusion of additional functionality beyond PKI on the card and to enable multiple vendors to provide card stock to the DoD. Identity proofing and certificate issuance on the CAC take place using the DoD's existing personnel identification card issuance system. Since the CAs do not have a direct method for interfacing with the CAC, issuance portals are used to facilitate key generation, certificate request generation, and insertion of issued certificates.

Software certificates can be issued to people and web servers. Although the CAC is the primary issuance process for personnel, software certificates are used to support some legacy applications that do not yet support hardware tokens, and in some environments where CAC issuance is difficult. Software certificates are requested via a Hyper Text Transfer Protocol, Secure (HTTPS) interface and verified by Registration Authorities (RA) and Local Registration Authorities (LRA).

For publication of PKI information, the DoD PKI interfaces with the Global Directory Service (GDS). GDS is an internal enterprise directory that supports both HTTPS and Lightweight Directory Access Protocol (LDAP) interfaces. Subordinate CA certificates, Certificate Revocation Lists (CRL), and encryption certificates are published to the GDS from the DoD PKI. Subordinate CA certificates and CRLs are also

published to an external X.500 directory for access outside of the DoD.

Certificate revocation is performed by RAs using an HTTPS interface. Key recovery is performed by key recovery agents who access the KED using an HTTPS interface.

2.2 *Certification Authority (CA) Scalability*

> Take into account all of the required tasks when developing architecture requirements.

When designing the infrastructure, the DoD performed load testing to determine how many certificates a given CA could issue. However, this initial load testing did not account for the many other functions that CAs must be able to perform at the same time as certificate issuance, including the following:

- validating credentials of trusted personnel,
- publishing certificates,
- generating CRLs,
- publishing CRLs,
- revoking certificates,
- responding to requests to search for specific certificates.

The DoD PKI has over 2,000 approved RAs. Because of personnel turnover, the list of approved RAs changes frequently. To ensure that certificates can still be issued if one or more CAs are unavailable, the DoD PKI is configured so that all RAs are authorized on all CAs. The interface provided by the vendor to manage trusted personnel did not support the large number of RAs or the requirement for frequent updates. To minimize the impact to CAs, the DoD has developed custom scripts that allow changes in RA personnel to be quickly uploaded to all CAs.

The process of generating CRLs requires significant CA processing time, both in determining which certificates have been revoked, and, as the CA ages, determining which revoked certificates have expired and should not be placed on subsequent CRLs. For CAs that issue a large number of certificates, the requirement to check each revoked certificate for its expiration date can cause the total time to generate a CRL to be greater than the next update period of the CRL. While CRL generation requires less processing time if expiration date checks are not performed, continuing to include expired certificates on CRLs increases the overall CRL size.

Although the GDS is the primary interface for applications to retrieve CRLs and for end users to search for encryption certificates, direct searches of the CA internal databases are still required, primarily for certificate revocation. When requesting certificate revocation, most users and supervisors do not know the CA and serial number for the certificate that needs to be revoked. As a result, the RA must search multiple CAs to locate the correct certificate prior to authorizing its revocation.

2.3 *Hardware and Software Maintenance*

> PKI cycle times are significantly different then hardware and software cycle times.

2.3.1 *CA Hardware and Software*

The DoD PKI was designed for the long term. The DoD Root CA has a validity period of thirty-six years. Each subordinate CA has a validity period of six years. Subordinate CAs issue certificates for the first three years of their validity period and are then "retired" so that they only issue CRLs for the remaining three years. CAs are only taken completely out of service once all certificates issued by the CA have expired. CACs issued to Government personnel are valid for three years, while those issued to contractors are valid for up to one year.

In contrast, hardware life cycles are one to three years, and software product cycles can be eighteen months or less. As a result, neither the software nor the hardware in use for the DoD Root CA are still supported by their respective vendors. Older subordinate CAs are also operating on non-supported versions of hardware and software, increasing both the requirement for and the cost of maintenance.

2.3.2 *Key Length*

In addition to product life cycles, the basic technology behind PKI is also changing. When the Root CA was established, 512-bit keys were still in use, and 1024-bit keys were the longest supported by vendors. Today, 1024-bit keys are standard, and the Federal Government has published guidelines that all certificates that will expire later than 2008 should be issued with 2048-bit keys.

As a result, the DoD PKI is currently working on a solution to upgrade the Root CA. There are two options for upgrading, migrating the current Root CA to newer hardware and software versions, or establishing a new Root CA and issuing a rollover certificate from the current Root CA to the new Root CA. Migrating the existing Root CA is simpler for

the short term, but does not solve the problem of the 1024-bit Root CA signing key length.

Establishing a new Root CA with a 2048-bit signing key will require pushing down the new key to all applications relying on certificates from the DoD PKI. In addition, a new Root CA will require maintaining two infrastructures for three to six years, depending on whether all current subordinate CAs are retired when the new Root CA is established.

2.3.3 Certificate Profile

Another issue with the long term nature of the PKI is that changes to certificate profiles require over three years to implement. For example, the DoD PKI initially did not support the extensions required to use digital certificates to authenticate to Microsoft Windows-based networks. Windows requires the following extensions in certificates[1]:

- CRL Distribution Point must be present,
- Key Usage must be set to Digital Signature,
- Extended Key Usage (EKU) must contain the Smart Card Logon Object Identifier (note that if the EKU extension is populated, it must also contain the identifiers for all other uses for the certificates, such as Client Authentication),
- Subject Alternative Name must contain a User Principal Name (UPN) of the format user@name.com.

Once the requirements were determined and the changes implemented at the subordinate CAs, all new CACs contained a signature certificate with the additional information. However, there are still some subscribers within the DoD that do not have the required extensions on their CACs.

Another example is the Authority Information Access extension. Research by the Federal Bridge Path Discovery and Validation Working Group has indicated that path discovery is facilitated when certificates contain the Authority Information Access (AIA) extension[2]. The DoD PKI does not currently include the AIA extension. If the DoD makes a decision to modify its certificate profiles to include the AIA extension, the change will take three years to be reflected in all DoD PKI issued certificates.

2.3.4 Smart Card Technology

In addition to CA and certificate profile updates, the DoD must manage user smart card migration issues. Since most CACs are valid for three years, CAC middleware must concurrently support three years of smart card technology. The DoD is investigating upgrading new card stock to a 64k chip, instead of the 32k chip currently supported. This new chip will support additional capabilities beyond PKI. The additional space may also support better security protections, which would enable users to perform more card maintenance, such as certificate update, from their own workstations instead of having to return to an issuance station. However, all DoD users will not be able to take advantage of these new capabilities until all cards have been replaced through normal expiration.

2.4 Personnel

> Integrating PKI rollout with existing processes is a requirement for success.

The initial DoD PKI rollout was planned as a software-based implementation. The DoD would centrally manage the PKI core components, and each DoD service or agency would provide personnel to act as RAs and LRAs who would register individuals. When early adopter applications tried to get their users registered to get certificates, however, they found that RAs and LRAs were not available. Local commands resisted the requirement for additional personnel, and the travel costs for sending RAs and LRAs to training.

At the same time the PKI was performing initial rollout, the personnel office was developing a new identification (ID) card to be rolled out to all DoD military and civilian employees. In November 1999, the DoD made a decision to combine the two programs and use the new ID card as a hardware token for digital certificates. As a result of this decision, the PKI and personnel offices worked together to design a process that used existing ID card issuance stations to verify identity and issue certificates in conjunction with ID cards.

This new process did increase the personnel requirements for ID card issuance stations. Prior to the CAC, DoD ID cards were only issued to military personnel, but CACs are issued to military personnel, civilian employees, and on-site contractors. Also, the time to issue CACs is longer then the time to issue the old ID cards. However, combining certificate issuance with ID card issuance allowed the PKI to take advantage of the existing ID card infrastructure and minimized the personnel requirements for services and agencies. Also, the requirement for personnel to get a new ID card facilitated the issuance of certificates.

3 Technology Challenges

Building the capability to support the DoD enterprise is not sufficient for the success of the DoD PKI. Since PKI is an infrastructure technology, it does not solve any operational requirements unless public key technology is integrated into applications. This section explores the two most significant challenges that the DoD PKI has experienced in gaining acceptance from the functional community for the use of PKI.

3.1 Certificate Status Checking

> Checking certificate revocation status is the most difficult technical challenge of PKI.

CRLs are theoretically elegant. They provide a mechanism for a CA to state that it no longer asserts the binding between the identity in the certificate and the associated key pair for a set of certificates that it issued. CRLs provide only a minimum set of data, the certificate serial number, the date of revocation, and optionally a reason for revocation. Because they are digitally signed, the transmission mechanism does not itself have to be trusted in order to accept the information contained in the CRL.

In practice, however, relying on CRLs has not worked well. Products that are enabled to use digital certificates only provide minimal support for CRLs. In some cases, no provision is provided to automate the downloading of CRLs. Vendors who do provide an automated update capability may not allow setting when the attempt to retrieve a new CRL occurs, which can result in multiple applications attempting to access the CRL repository at the same time. At least one vendor treats the next update field of a CRL as an expiration date, and will not validate any certificate issued by a CA for which a current CRL is not available. Finally, the information contained in a CRL is only as current as the time the CRL was published, which results in significant latency issues.

The scale of the DoD PKI results in an additional problem, the overall size of CRLs. The combined size of the CRLs from all of the DoD PKI CAs is approaching 40 megabytes. It is not feasible for every application on DoD networks to download this amount of data every day without having a significant impact on available bandwidth.

Although CRLs are an efficient way of publishing revocation information for the entire PKI, no single application has all subscribers as users, so each application only needs a subset of the information. However, the enterprise PKI does not know in advance which subset is needed by each application.

The DoD PKI has examined two alternate CRL approaches, partitioned CRLs and delta CRLs. A CA creating partitioned CRLs divides certificates into blocks of a preset size based on information contained in the certificate such as the certificate serial number. Instead of issuing one CRL, the CA issues multiple CRLs, one for each preset block of certificates. When an application attempts to validate a certificate, it checks to see if a current CRL for the block the certificate is contained in is locally cached, and downloads the CRL partition if it is not. While partitioned CRLs allow applications to only retrieve limited CRL information, the DoD has not developed a solution involving partitioned CRLs, partially because of the lack of support from either CA or application vendors.

A CA supporting delta CRLs issues a full CRL once or periodically, and then only issues delta CRLs that contain certificates that have been revoked since the last delta CRL was issued. As a result, delta CRLs are significantly smaller than full CRLs. However, applications must have a mechanism of ensuring that they have downloaded all delta CRLs, because no single CRL can be considered an authoritative source for information on the revocation status of any given certificate. Although the DoD PKI does not support delta CRLs, one agency within the DoD has successfully piloted a delta CRL approach for transmitting revocation information in a severely bandwidth constrained environment.

In general, CRLs have been an effective method of transmitting revocation information across enterprise networks with high bandwidth availability, but are too cumbersome to use to get this information down to individual applications.

As a result of the continued issues with performing certificate validation using CRLs, the DoD PKI is deploying an infrastructure to support revocation status checking using the On-line Certificate Status Protocol (OCSP). To meet the requirements for decentralization, availability, and scalability, this infrastructure will not interface directly with the DoD PKI CAs. Instead, it will provide a capability to download CRLs and provide real-time OCSP responses from multiple locations across the DoD network. Instead of downloading CRLs, applications that support OCSP will able to get real-time responses for specific certificates from this global robust certificate validation system. Although the use of CRLs as the authoritative source for revocation information does not address the latency issues of CRLs, this hybrid approach of CRLs and OCSP will take advantage of the efficiency of CRLs

and provide an interface for applications that is easier to implement and maintain.

3.2 Key Recovery

> The person most likely to need key recovery capability is the subscriber.

The DoD PKI key escrow and recovery solution was designed when the DoD PKI was primarily issuing software certificates. The solution was designed to support situations when a private key needed to be recovered by a manager or law enforcement agent to access encrypted data, and occasionally by subscribers who had lost their private keys. As a result, the process was manual and personnel intensive, requiring first that requestors verify their own identities and provide justification for the request, and then that two key recovery agents together retrieve the escrowed keys out of storage.

The transition to the CAC as the token for key generation and storage created a significant change in the key recovery requirement. Since the private key associated with each subscriber's encryption certificate is stored only on the CAC, subscribers lose access to their own private keys at CAC expiration because the old CAC is surrendered at the time of new CAC issuance. In order to access files previously encrypted, all subscribers must recover their old private keys.

The manual process which was designed to prevent abuse of key recovery is too costly to support for a large number of individuals requesting their own escrowed private keys. Since individuals are assumed to be authorized to have access to their own keys, a more automated process is being developed that will allow subscribers to use their identity certificate to authenticate to the key escrow system and request retrieval of their own private keys.

4 Organizational Challenges

Although the implementation of the DoD PKI has experienced technical challenges, overcoming organizational obstacles has sometimes proved a harder task. Some of these obstacles are independent of PKI, such as issues relating to coordinating common processes across the worldwide enterprise, developing working relationships between disparate commands within the enterprise, and determining which organizations within the enterprise should have primary responsibility for each element of the overall architecture. This section highlights some of the organizational challenges specific to PKI implementation for users, managers, and developers.

4.1 The Users

> Provide users with new capabilities that help them to get their jobs done, not just PKI certificates.

Most users will embrace new technology if they see a clear benefit to its use. However, PKI was initially marketed as a technology, not as a mechanism for getting the job done. For example, "PKI 101" training often starts by stating the concepts of public key technology, then introduces the user to "Alice" and "Bob" who are exchanging signed and encrypted email messages. By this time, attendees have decided that PKI is very complicated and since they can send email without PKI (and have been doing it for years without problems), they leave the training having decided that PKI is too difficult.

The DoD PKI was originally targeted as a pilot that would provide better assurance for a new electronic travel system. Through the use of digital signatures, travel claims could be processed significantly faster, resulting in shorter times for employee reimbursements. Once deployed, the PKI could then be used with other systems. However, delays in the rollout of the travel system and the decision to implement a smart card-based PKI meant that many users were issued a CAC months prior to receiving a smart card reader and without any application requiring use of their new certificates. As a result, most users' experience with PKI consisted of waiting in line to get a CAC and then using the CAC the same way they had used the ID card they had prior to the CAC. These users did not see any real benefit to the new technology.

Although smart card readers are being deployed and applications are beginning to incorporate support for public key technology, the DoD PKI continues to struggle to attain widespread user acceptance.

4.2 The Managers

> Application owners need policy, budget guidance, and a business justification for adopting public key technology.

Within the DoD, funding for PKI core components and card stock is centrally managed. However, individual applications, including email, networks, and web servers, are very decentralized. Therefore, rolling out the PKI required a few decisions by policy makers, but integrating public key technology into applications requires many decisions by many application owners. These managers must consider

multiple demands when determining how to allocate limited resources:

Getting support from application managers requires providing managers with the information they need to make decisions including the following:

- Ensure that published policy is consistent with the organization's goals for integrating public key technology. Policy enables early adopters of new technology to justify their investments.
- Provide direction for requesting funding for public key enabling as a part of the standard budget cycle. Getting out of cycle funding to meet security driven requirements is difficult and almost always means that other planned functionality must be sacrificed to meet PKI requirements.
- Use specific examples when presenting security requirements to application owners to show what vulnerabilities exist in current systems and how the use of PKI can help to mitigate them.
- Define business case benefits for PKI in addition to the "better security" case. Because PKI acceptance has been primarily in the security community, explanations for why to integrate public key technology tend to be heavily security focused. The business case for PKI, including more efficient user management, decreased password management, and new functionality capabilities, should be more clearly stated.

Getting the acceptance of application owners is a critical step in showing a return on investment in PKI. However, most application owners will not commit to enabling existing applications until the users have digital certificates. DoD applications that made initial investments in using PKI in the late 1990s all delayed public key enabling because of the inability of their internal DoD users to register for certificates. Now that the DoD has invested resources to issue certificates to eligible users, these and other applications are finally beginning the transition to using public key technology.

4.3 The Developers

Better training is needed to assist developers in public key enabling.

Ultimately, the success of PKI is dependent on developers performing system integration to public key enable applications. DoD applications usually involve some components, such as web servers, that have native support for some PKI capabilities and other components that do not support PKI. Public key enabling, therefore, can require upgrading software to later versions that support required capabilities, replacing components with similar components from different vendors that support required capabilities, and building interfaces between enabled components and those that do not support public key technology.

Unfortunately, there are relatively few individuals who understand both PKI and application architectures. Available PKI training consists primarily of lessons on how to stand up the infrastructure; it does not focus on how to integrate PKI into existing applications. Vendors provide some guidance, but this information is usually limited to the vendor's own products.

For example, a web server vendor will provide instructions on how to request and install a server certificate and how to turn on client certificate-based authentication. The vendor may also provide instructions on how to perform certificate validation. However, nothing is provided on how to integrate the authenticated identity information from the certificate with access control to a database or other back end component.

Training targeted at developers on how to integrate PKI into real-world applications should become much more widely available.

5 Conclusion

As the DoD has rolled out PKI, it has experienced technical challenges. However, PKI has been more successful than many technologies in meeting the scalability demands of the DoD enterprise. By integrating certificate issuance with existing personnel processes, the DoD PKI has been able to perform in-person identity verification to over three million users. Technology challenges have been met using a combination of redundant systems and customized interfaces developed by DoD, contractor, and vendor personnel working together. The only basic building block of PKI that has not scaled successfully is the CRL. However, the DoD is working to overcome this barrier through the use of OCSP.

The DoD PKI rollout has also encountered issues surrounding the enterprise nature of PKI. PKI implementation has required that existing business process problems be resolved prior to the success of PKI.

The more difficult challenges have been in getting acceptance from the user, application owner, and developer communities. A primary reason for this issue is the lack of good training available targeted specifically to the interests of these communities.

The DoD is committed to continuing down the path of PKI deployment and public key enabling of applications. The implementation of PKI is a long term investment, since application owners do not want to commit to using certificates until they believe that their user population has the capability to get certificates. Unfortunately, the return on investment in PKI does not become measurable until applications have started to use the technology. Staying the course has presented challenges, but the potential of public key technology, both in current architectures and in the next generation Net-Centric environment, is critical to meeting the DoD's information assurance goals and improving its business processes.

---

[1] "Guidelines for Enabling Smart Card Logon with Third-Party Certification Authorities," Microsoft Knowledge Base Article 281245. http://support.microsoft.com/default.aspx?scid=kb;en-us;281245

[2] "Functional Requirements for Path Validation Systems," Path Validation Working Group, Draft Version 0.8, March 2004. http://www.cio.gov/fbca/pdvalwg.htm

Language based Policy Analysis in a SPKI Trust Management System

Arun K. Eamani and A. Prasad Sistla§
University of Illinois at Chicago
{aeamani, sistla}@cs.uic.edu

Abstract— **SPKI/SDSI is a standard for issuing authorization and name certificates. SPKI/SDSI can be used to implement a Trust Management System, where the policy for resource access is distributively specified by multiple trusted entities. Agents in the system need a formal mechanism for understanding the current state of policy. We present a first order temporal logic, called FTPL for specifying properties of a given SPKI/SDSI policy state. We also present algorithms to check if a SPKI/SDSI policy state satisfies a property specified in FTPL.**

I. Introduction

A. Motivation and Use of the Language

SPKI/SDSI (simple public key infrastructure/simple distributed security infrastructure) is a mechanism for specifying policy in a distributed access control system [EFL+99], [Ell99]. With standardized semantics it can also be viewed as a Trust Management Language [BFK99]. There has been much work done on analyzing fixed a priori defined properties of such an authorization system [JR01], [CEE+01]. In this paper we introduce a language based on general purpose temporal logic for specifying properties of such a system. This language, called FTPL: First order Temporal Policy analysis Logic, could be used by the agents to reason about the state of the policy. Such analysis is useful for understanding the current state of policy, auditing the past policy statements, to point out any violations in trust between agents and for aiding policy refinement/management. We present efficient methods for automatically checking if a given SPKI/SDSI certificate set satisfies a policy question specified in the logic. The logic that we use is an extension of the standard real-time temporal logic extended with quantifiers over the principals in the system. Many important properties such as the following can be expressed in our logic - *can two principals K_1 and K_2 access the resource R simultaneously at some point in future?* We can also specify queries in this logic, for example *retrieve all the principals who were able to access a resource R at some point in a time interval.*

§ This research is partially supported by the NSF Grants CCR-0205365and CCR-9988884

There have been a string rewriting system [CEE+01] and a push down system(PDS) [JR01] to model certificate analysis problems in a SPKI/SDSI framework. We propose that temporal logic be used as a language to specify the issues of authorization, naming and validity. The policy analysis problem would be specified as a temporal logic formula f and we evaluate the formula on a given set of certificates $\mathcal{C}$ and a particular time instance t. The formulas of the logic could be interpreted not only to evaluate the truthness of a statement but also to reason about the state of the system by finding all instantiations of free variables which would make the formula true in the given state of system at a particular instant of time.

In access control models the usual analysis consists of safety analysis [HRU75]: does there exist a reachable state where a untrusted principal has access to the resource? A state in the system corresponds to a set of signed policy statements i.e certificates. We change the policy state by adding or deleting a certificate. Li et al [LWM03] propose security analysis where the state can change in a restricted manner. Security analysis is a study of security properties such as safety, availability, containment etc. Availability requires that in every reachable state a particular principal will be able to access a resource. Containment requires that in every reachable state if a particular principal has a property, say being able to access a resource, then that principal also satisfies a condition, like being member of a particular role. The above type of analysis is useful to understand how the current policy state can evolve.

We propose *policy analysis* in distributed access control models, where we analyze properties of a particular policy state. We consider a policy state as consisting of a set of policy statements each labeled with a validity interval. Each policy statement is valid during

the time interval associated with it. At any point of time only a subset of the given set of certificates are valid. Policy analysis can also be viewed as a special case of security analysis where all the possible state transitions are know in advance, given that certificate issuers specify the time period during which the certificate is valid. We can reformulate the problems of security analysis to reason over time instead of reachable states. For example the availability property could be restated as: *at all times in future does a principal K have access to a resource R.* While, for practical reasons, we may want to reason about bounded availability: *at all times during a time interval, say a school semester is a student K able to access the school ftp server R.*

In a SPKI/SDSI system, given certificates containing validity intervals, authorization and name relationships vary with time. In a delegation based system where multiple agents make policy statements regarding access of a particular resource, no agent is aware of the overall state of the policy. There being a large number of policy statements, manual analysis is not an option. Many policy specification languages including SPKI/SDSI cannot specify constraints such as negative credentials, mutual exclusion, etc. necessitating a mechanism to make sure that agents didn't make policy statements which violate the unspecifiable constraints. In addition, analysis of current policy state is useful to formally reason whether the current policy state satisfies user defined properties and to gain knowledge for making decisions about changing the current policy state. Based on current policy state, we can also reason about issues of authorization and naming in future time.

We also propose *policy audit*, where we check for policy violations over a set of all the policy statements which were valid during the time of audit. In access control audit, the problem is of type: *did principal K access a resource R?*, while in policy audit we check *whether a principal K had permissions to access a resource R?* or *whether principal K_1 gave authorization regarding a resource R to principal K_2*. Policy audit is a means to ascertain whether trusted agents were really trustworthy with regards to policy specification. By suitably shifting the timeline, we can use FTPL for purpose of reasoning about current policy state as well as for policy audit. For purposes of policy audit, we collect the set of all policy statements corresponding to the time of audit and evaluate the formulas of the logic over the static collection of policy statements labeled with validity periods.

The logic we propose would provide the resource administrator with a formal and a high level mechanism for specifying policy properties. The language could also be useful for the other types of users, like a client to reason about properties such as *"is there an authorization chain leading to the client from a particular principal?"* etc.

The paper will consist of five other sections, section [2] presents short description of related work. Section [3] consists of introduction to SPKI/SDSI and previous models for certificate analysis problems. Section [4] will consist of our proposed logic and examples of policy problems specifiable by the language and section [5] will contain algorithms to evaluate the formulas of the logic. Section [6] consists of conclusion and future work.

II. Related Work

There are other logics for SPKI/SDSI which model the name resolution, authorization and other features of SPKI/SDSI: Martin Abadi's logic to model local name spaces of SDSI [Aba97], Halpern and van der Meyden's Logic of Local Name Containment to characterize SDSI name resolution which was extended to deal with other SPKI issues like revocation, expiry dates, and tuple reduction [HvdM01]. Ninghui Li [Li00] proposes a logic program to describe the name resolution algorithm which also handles authorization certificates and threshold subjects. The purpose of our logic is neither to model the features of SPKI/SDSI nor to provide for its semantics but for policy analysis. Jha and Reps [JR01] propose using temporal logic to reason about certificate chain derivations in a given SPKI system, while we propose using temporal logic as a language for specifying certificate set analysis problems involving authorization, naming and validity.

There have been many languages, logic and semantic frameworks to express authorization policy in a distributed system. Datalog and its variants seem to be the language of choice to specify authorization policy [LM03]. A specific class of distributed access control systems attracting much attention are the Trust Management Systems [BFK99], [BFIK99]. The concept of Trust Management is that there exists a standard mechanism for expressing the access control policy and also a standard method to verify that an access request complies with the policy. This is called "Proof of Compliance". SPKI/SDSI was not intended by the authors to be a Trust Management System in that Proof of Compliance can be application dependent, but it can be viewed as a Trust Management System with

standardized certificate chain reduction rules [EFL+99]. Though lot of work has been done in specifying policy and checking whether an access request is compliant with the policy in a distributed access control system, not much literature exists regarding a language based mechanism for detailed analysis of policy beyond simple authorization problems. To our knowledge FTPL is the first such language for high level policy analysis in not just a SPKI/SDSI system, but in the distributed access control framework. While we give a logic to formulate policy analysis problems in the context of SPKI/SDSI we feel that full fledged languages for policy analysis in other distributed access control systems and trust management systems are of much use to the users of the system.

We give a list of specifiable problems that could be of interest in SPKI/SDSI policy analysis including some given by Jha and Reps [JR01]. The problems they list can be qualified by time. For example Authorized Access 1:"Given resource R and principal K, is K authorized to access R?", can be more specifically written in the context of time as:"Given Resource R and principal K is K authorized to access R at a particular instant of time, or at some point in a time interval?" While Jha and Reps provide a lattice based time structure which can be used to model such problems, we give a logic based formalism to specify such problems.

III. SPKI/SDSI

A. Introduction

SPKI and SDSI are certificate based mechanisms for authorization and naming in distributed systems respectively. SPKI 1.0 : simple public key infrastructure, was a mechanism for expressing authorizations and delegations, where it was proposed that permissions be given directly to the public keys of entities. The most important contribution of SDSI: simple distributed security infrastructure, was to create local namespaces distinguished by the unique public key of the entity defining the names, instead of trying to create a globalized namespace. Features of SPKI 1.0 and SDSI 1.0 were integrated into SPKI/SDSI 2.0 [EFL+99]. In the future if we say SPKI we mean SPKI/SDSI 2.0 unless mentioned otherwise. One of the features of the SPKI is that every principal is free to issue certificates signed by himself unlike in X.509 PKI framework [ADT02] where there are separate set of principals called certificate authorities(CA) who are trusted to issue certificates bearing their signature.

In SPKI/SDSI resource owners can delegate access rights to trusted entities and they can issue certificates authorizing others, leading to a chain of certificates from the resource owner to the end user. In a SPKI/SDSI system principals and resources are identified by their corresponding public keys. We can still associate names with the principals. Every principal has a local name space and the principal is free to define local names in his domain independent of others. For example for a UIC student John, university may be defined as K_{UIC}, while for a UIUC student Jim, university may be defined as K_{UIUC}. K_{UIC}, K_{UIUC} are the public keys of the universities UIC and UIUC respectively. Public keys being unique throughout the system, a name qualified by the public key of the principal defining the name, is also unique.

Definition 1: An identifier is a word over a given alphabet. The set of identifiers is denoted by $\mathcal{I}$. The set of keys is denoted by $\mathcal{K}$.

Definition 2: A local name is a sequence consisting of a key followed by a single identifier. A local name K *friend* may be defined as K_{bob}, K_{jim},... etc

There is another kind of name in SPKI/SDSI called extended name which increases the level of indirection in the naming scheme.

Definition 3 : An extended name is a sequence consisting of a key followed by two or more identifiers. An example of an extended name is K *brother friend*, where the meaning of *friend* would be evaluated in the context of $K's$ *brother*.

Definition 3: A Name is either a local name or an extended name. We denote the set of all names as $\mathcal{N}$.

Definition 4: A Term is either a key or a name. We use $\mathcal{T} = \mathcal{K} + \mathcal{N}$ to denote set of all terms.

Definition 5: We define a fully qualified name also to mean a public key or a local name or an extended name.

B. Certificate Structure

There are two types of certificates or certs in SPKI/SDSI: name certs and authorization(auth) certs. The function of a name cert is to define a local name as another term. Only the principal, in whose namespace the local name is defined, may issue the corresponding name certificate. In an auth cert a principal can grant or delegate permissions regarding accessing a resource to another term. We now describe the logical structure of the SPKI/SDSI certificates.

There are four fields in a name certificate (K, A, S, V) which is signed with the private key of the issuer:

K^{-1}. K is the public key whose namespace is being considered, A is an identifier in $\mathcal{I}$ and S is an element of $\mathcal{T}$, i.e. it can be another public key, or a local name or an extended name. S is the term implied by the local name KA . V is a validity specification which states the validity condition for the certificate. The basic validity specification is of form $[t_1, t_2]$ where t_1, t_2 are absolute time constants. The certificate is said to be valid from time t_1 *to* t_2. If either t_1 or t_2 are not given we assume $-\infty$ or $+\infty$ respectively. Validity specification could also be a certificate revocation list or an online check [CEE+01]. In our model of SPKI we consider a validity specification to be a certain time period in the interval 0 to ∞.

The authorization certificates consist of five fields $(K, S, D, \mathcal{E}, V)$ signed by the private key of the issuer: K^{-1}. The main purpose of an SPKI authorization certificate is to grant or delegate a set of authorization actions as specified by $\mathcal{E}$ to the subject $S \in \mathcal{T}$.

K is the public key of the certificate issuer.

S is the Subject which can be either a key or a name.

D is the boolean delegation bit which controls the delegation rights.

$\mathcal{E}$ is the set of authorization rights which are granted or delegated to the subject by the issuer. Note that authorization actions have address of the resource encoded in them. In our logic when we say *read* we mean *read* of a particular resource R defined in the context.

V is the Validity specification and the observations made above for name certs are applicable here also.

Delegation bit D implies that when the bit is set to one then the subject can access and also delegate the rights specified by set $\mathcal{E}$ to other users in the system, if the bit is zero then the subject only gets the right to perform actions specified by $\mathcal{E}$, but cannot further delegate these rights to any other user.

C. Tuple Reduction

In a delegation based distributed access control system the resource owner permits others to specify policy regarding the access of the resource. Entities entrusted with delegation rights further propagate the access rights. This leads to a chain of certificates for accessing a resource. The principal requesting access would present the resource administrator with a requested authorization action and a chain of certificates which should prove that the requesting principal has the right to perform the requested action. Given a chain of certificates, one must deduce authorization and validity information from the chain. The rules which specify the inference conditions are called the tuple reduction rules. When an entity receives a set of certificates it verifies them for integrity and stores them in an unsigned form called the tuples. We use the terms certificates and tuples interchangeably as far as there is no confusion.

Consider a certificate issued by K_1 to a key K_2 with authorization actions $\mathcal{A}_1$ and and validity specification V_1, suppose delegation bit D_1 is also true. Then the certificate is logically represented as 5-tuple

$$C_1 : (K_1, K_2, D_1, \mathcal{A}_1, V_1)$$

Given D_1 to be true, let K_2 delegate authorization actions $\mathcal{A}_2$ to subject S_2, with validity condition V_2 and delegation bit D_2.

$$C_2 : (K_2, S_2, D_2, \mathcal{A}_2, V_2).$$

According to the tuple reduction rules the result of composition of the certificates C_1 and C_2 in that order is another certificate C' equivalent to the chain

$$C_1 \circ C_2 \equiv (K_1, S_2, D_2, AIntersect(\mathcal{A}_1, \mathcal{A}_2),$$
$$VIntersect(V_2, V_2)).$$

These equivalent certificates can also be used as a regular certificates in the SPKI system and are called Certificate Result Certificates (CRC).

$AIntersect()$ is the intuitive intersection of authorization action sets. Howell and Kotz [HK00] note that the intersection may not always be well defined. For the purpose of our logic we consider authorization actions to be simple constants. Date range intersection $VIntersect()$ is the intersection of corresponding validity intervals.

D. Models for SPKI Certificate Analysis.

There are two primary models for tuple reduction problems, one is the string rewriting model of Clarke et al [CEE+01] and the other is the push down system(PDS) based semantics of Jha and Reps [JR02]. We present the PDS based model of [JR02]

1) Push Down System based Semantics: Jha and Reps [JR01] model tuple reduction problems as configuration reachability problems in a pushdown system. A push down system is similar to a push down automata but without the input alphabet. They view the authorization problem: *can a principal* K_p *access resource a* R as a configuration reachability problem in the pushdown automata and use the PDS model checking algorithms [EHRS00], [BEO97] to answer the problem. The configuration of a PDS contains information about the control state and stack state of the PDS at a particular point in the computation. A configuration of PDS corresponds to a term in a SPKI system. If

a configuration G_T reaches another configuration G_S using the state transition rules, then the corresponding term T can resolve to term S, defining the "term reachability semantics" for the SPKI/SDSI system. PDS State transition rules correspond to the SPKI/SDSI certificates. The PDS formalization is also more expressible then the rewriting system of Clarke et al [CEE+01] to formulate various certificate analysis problems. We now describe the method to build a push down system from a given set of certificates and then model the certificate analysis problems using configuration reachability relation of the PDS. The primary issues of certificate analysis are that of name resolution and authorization derivability. For purposes of the algorithm we convert name certs of form (K, A, S, V) into the rewriting rule $KA \rightarrow S$, auth cert $(K, S, d, \mathcal{E}, V)$ is written as rule $K\Box \rightarrow S\Box$ if $d = 1$ else as $K\Box \rightarrow S\blacksquare$ if $d = 0$. V and $\mathcal{E}$ are ignored for now, we consider that each resource has only one type of access right.

A pushdown system is formally defined as a triple $\mathcal{P} = (Q, \Gamma, \Delta)$, where Q is a finite set of state variables, Γ is a finite set of stack symbols, and $\Delta \subseteq Q \times \Gamma \times Q \times \Gamma^*$ is the finite set of state transition rules. If $(q, \gamma, q', w) \in \Delta$ then we write $(q, \gamma) \hookrightarrow (q', w)$. The PDS $\mathcal{P}$ when in state q and γ on top of stack, uses the above transition rule to goto state q', popping γ and writing string w onto the stack. The $\hookrightarrow$ corresponds to the rewriting relation between various elements of a SPKI certificate. A configuration of $\mathcal{P}$ is a pair $\langle q, w \rangle$ where $q \in Q$ and $w \in \Gamma^*$. Here q denotes the control state and string w the stack state. The set of all configurations is denoted by $\mathcal{G}$. Corresponding to the transition relation of the PDS states we have the (immediate) reachability relation of the configurations. If $(q, \gamma) \hookrightarrow (q', w)$ then for all $v \in \Gamma^*$, $\langle q, \gamma v \rangle \Rightarrow \langle q', wv \rangle$, i.e. configuration $\langle q, \gamma v \rangle$ is said to be the immediate predecessor of $\langle q', wv \rangle$. The reflexive transitive closure and the transitive closure of the immediate reachability relationship($\Rightarrow$) are denoted by $\Rightarrow^*$ and $\Rightarrow^+$ respectively. A *run* of a PDS is a sequence of configurations $c_0, c_1, ...c_n$ such that c_i is an immediate predecessor of c_{i+1}. The run of a PDS corresponds to a SPKI certificate chain reduction.

Given a set of certs $\mathcal{C}$ with principals(keys) represented by $\mathcal{K}$ and identifiers by $\mathcal{I}$, we construct a PDS $\mathcal{P}_\mathcal{C}$ (Q, Γ, Δ) as follows.
The set of control states is the set of keys, $Q = \mathcal{K}$. Note that we only analyze authorization problems concerning a particular resource which is identified by a special key R in $\mathcal{K}$. The stack alphabet is set of identifiers and the delegation symbols, $\Gamma = \mathcal{I} \cup \{\Box, \blacksquare\}$. The symbol $\Box$ implies permission to delegate while $\blacksquare$ just grants access. The set Δ of state transitions contains a transition $(K\ \gamma \hookrightarrow K'\ \omega)$ for every cert rule $(K\ \gamma \rightarrow K'\ \omega)$. The term $K_1\ A\ B$ corresponds to the PDS configuration $\langle K_1, A\ B \rangle$, term K corresponds to $\langle K, \epsilon \rangle$. When the PDS is in configuration $\langle K_1,\ A\ B \rangle$ and there is a state transition rule(cert) $K_1\ A \hookrightarrow K_2$ then it goes to configuration $\langle K_2,\ B \rangle$. The reachability relation $\Rightarrow^*$ defines set of all terms which a given term can resolve to. For authorization derivability, the PDS configuration is of the form $\langle K, \Box \rangle$ which means principal K can access and delegate permissions, or of the form $\langle K, \blacksquare \rangle$ which means K can just access. A principal K can access the resource R if there is a run of the PDS from configuration $\langle R, \Box \rangle$ to either of the configurations $\langle K, \Box \rangle$ or $\langle K, \blacksquare \rangle$. Whether a term can resolve into another term can be decided in the PDS system in time $O(n^2 L_\mathcal{C})$ where n is the total number of keys in the certificate system $\mathcal{C}$, and $L_\mathcal{C}$ is the sum of lengths of the right hand side(rhs) of all the certificates in the system. Length of the rhs of a cert is the number of non-key symbols on the rhs of the rule corresponding to the cert.

We use term reachability semantics of the PDS model to define the FTPL semantics. For that purpose we modify the PDS described above by adding an authorization variable v to the system to form a augmented PDS $\mathcal{P} = (Q, \Gamma, \delta, v)$. The authorization rights a principal grants to another principal are computed as the side-effects of the run of the PDS using the authorization variable v. At the start of a computation v is initialized to $\mathcal{A}$, the set of all authorization rights for the resource being considered. Each auth cert rule is labeled with the authorization rights granted to the subject of the cert. Accordingly, we also label the corresponding state transition rule in δ. In the PDS $\mathcal{P}$ whenever we go from one configuration to another configuration using a state transition rule labeled with a set of authorization rights $\mathcal{E}$, we update the authorization variable v as $v = v \cap \mathcal{E}$. If the PDS goes from configuration $\langle K_1, \Box \rangle$ to $\langle K_2, \Box \rangle$ with the value of v as $\mathcal{A}_1$ at the end of the run, then K_1 directly/indirectly delegates to K_2 the authorization rights $\mathcal{A}_1$.

IV. Logic for Policy Analysis

A. Definition of the Logic

We define a polyadic First order Temporal Policy analysis Logic (FTPL) as a language for specifying properties of a SPKI/SDSI certificate system. The formulas of the logic are to be evaluated on a semantic model of the

SPKI/SDSI system. The policy analysis problem would be specified as a FTPL formula f which is interpreted on a given set of certificates at a particular instance of time. Users can choose application dependent semantics for the language, but we use the standard tuple reduction semantics [CEE+01] with the view of SPKI as a trust management system. The constructs of the logic deal with the issues of authorization, naming and validity. Though we model time as discrete in this paper, we can easily extend FTPL to interpret over continuous time.

The authorization reachability issue is modeled by the predicate $authorize(i, j, d, e)$, called the authorization reachability predicate, where i is a key or a variable and j is a fully qualified name or another variable. The components of *authorize* predicate d, e specify the delegation and authorization rights respectively. The boolean delegation control $d \in \{0, 1\}$, with 1 implying permission to access and delegate and 0 just granting access. The authorization control e is a subset of a set of constants $\mathcal{A}$, which specifies the set of authorization actions for a particular resource. The predicate $authorize(i, j, d, e)$ is true at a particular instance of time t with respect to a given set of certificates $\mathcal{C}$, if there is a chain of certificates, where the principal denoted by i directly/indirectly transfers the rights d, e to the principal or name denoted by j. Name certificates are not just mechanisms to resolve names to principals but can also be used to delegate permissions [Li00]. To reason exclusively about name resolution we define the predicate $resolve(p, q)$ where p is either a local or an extended name and q is either a fully qualified name or a variable. The predicate reasons whether the name denoted by p resolves to the term denoted by q using the name certificates given in $\mathcal{C}$. The type of a variable in case of both $authorize$ and $resolve$ is the set of all principals in the system. The only atomic formulas of the logic, are the authorization reachability predicate $authorize(i, j, d, e)$ and the name resolution predicate $resolve(p, q)$. The rest all formulas are combinations of the predicates with the associated FTPL operators. The operators of the logic consist of the usual boolean connectives $\neg$ *and* $\wedge$, the temporal nexttime operator $\mathbf{X}$, the bounded until operator $\mathbf{U}_{[t_1,t_2]}$ where t_1, t_2 are positive integers and $t_1 \leq t_2$, the principal quantifier $\exists$ where the universe of discourse is the set of principals(keys) present in the certificate system $\mathcal{C}$, over which the formulas of the logic are interpreted. These are the basic operators. We assume we have set $\mathcal{V}$ of variables. All the variables range over the set of principals present in the given certificate set $\mathcal{C}$. Let $\mathcal{K}$,$\mathcal{T}$ be the set of keys and terms present in certificate set $\mathcal{C}$ respectively and $\mathcal{A}$ be the set of authorization actions for a particular resource R.

The syntax of the logic is inductively defined as follows:

1)) If $i \in \mathcal{K} \cup \mathcal{V}$, $j \in \mathcal{T} \cup \mathcal{V}$, $d \in \{0, 1\}$ and $e \subseteq \mathcal{A}$ then $authorize(i, j, d, e)$ is a FTPL formula.

2) If $p \in \mathcal{N}$ and $q \in \mathcal{T} \cup \mathcal{V}$ then $resolve(p, q)$ is a FTPL formula.

3) If f and g are formulas of the logic, then $\neg f$ and $f \wedge g$ are also FTPL formulas.

4) If f and g are formulas of the logic $\mathbf{X}f$, $f\ \mathbf{U}_{[t_1,t_2]}\ g$, are also FTPL formulas.

5) If f is a formula of the logic then $(\exists i f(i))$is also a FTPL formula.

A variable i is bound in a formula f, if every occurrence of i in f is in the scope of a quantifier over i. If i occurs in f and is not bound then we call i a free variable of f. We use free variables in the formula to formulate queries over a SPKI system. The formula will return a set of evaluations which when substituted for the free variables would make the logical formula true in the context of the given certificate system $\mathcal{C}$ at a particular instant of time t. For a formula f, let $free\text{-}var(f)$ denote the set of free variables of f. An evaluation say ρ for the formula f is a mapping from $free\text{-}var(f)$ to the set of principals, $\rho : free\text{-}var(f) \rightarrow \mathcal{K}$. We extend the domain of ρ to $(free\text{-}var(f) \cup \mathcal{T})$ so that for every $T \in \mathcal{T}$, $\rho(T) = T$. An interpretation for a formula f is a triple $(\mathcal{C}, t, \rho)$ where $\mathcal{C}$ is a set of certificates, $t \geq 0$ is a time instance and ρ is an evaluation.

We give the semantics of a formula f over an interpretation $(\mathcal{C}, t, \rho)$. For any certificate C let $C.val$ be the validity time interval of C. For a given set $\mathcal{C}$ of certificates and time t, let $\mathcal{C}_t = \{C \in \mathcal{C} | t \in C.val\}$. $\mathcal{C}_t$ denotes the set of certs valid at time t and $\mathcal{P}_{\mathcal{C}_t}(Q, \Gamma, \delta, v)$ denotes the PDS constructed from the set of certs $\mathcal{C}_t$ as described in previous section.

We define the satisfiability relation $\models$ between an interpretation and a FTPL formula as follows:

$(\mathcal{C}, t, \rho) \models authorize(i, j, d, e)$ if the PDS constructed from set of certificates $\mathcal{C}_t$, $\mathcal{P}_{\mathcal{C}_t}$ can go from configuration $\langle \rho(i), \square \rangle$ to the configuration $\langle \rho(j), \square \rangle$, with authorization variable $v \supseteq e$, at the end of the computation, when $d = 1$ i.e. $\langle \rho(i), \square \rangle \Rightarrow^* \langle \rho(j), \square \rangle$. If $d = 0$ the PDS $\mathcal{P}_{\mathcal{C}_t}$ can reach either the above configuration or the configuration $\langle \rho(j), \blacksquare \rangle$.
Note that if the key $\rho(i)$ directly/indirectly issues term $\rho(j)$ with greater delegation and authorization rights

than specified by d and e respectively, the $authorize$ predicate still holds true.

$(\mathcal{C}, t, \rho) \models resolve(p, q)$ if the PDS $\mathcal{P}_{\mathcal{C}_t}$ can go from configuration corresponding the name p, to the configuration corresponding the term $\rho(q)$.

$(\mathcal{C}, t, \rho) \models \neg f$ iff $(\mathcal{C}, t, \rho) \not\models f$

$(\mathcal{C}, t, \rho) \models f \wedge g$ iff $(\mathcal{C}, t, \rho) \models f$ and $(\mathcal{C}, t, \rho) \models g$

$(\mathcal{C}, t, \rho) \models \mathbf{X}f$ iff $(\mathcal{C}, t+1, \rho) \models f$.
$\mathbf{X}f$ is true at an instance of time t iff f is true in the next instance of time $t+1$.

$(\mathcal{C}, t. \rho) \models f\ \mathbf{U}_{[t_1,t_2]}\ g$ iff $\exists t' \in [t+t_1, t+t_2]$ such that $(\mathcal{C}, t', \rho) \models g$ and $\forall t'' :\ t \leq t'' < t'\ (\mathcal{C}, t'', \rho) \models f$.
$f\ \mathbf{U}_{[t_1,t_2]}\ g$ is true at time t, iff formula f is true from time t to t', where at t', g will be true and $t' - t$ should be within bounds of $[t_1, t_2]$.

$(\mathcal{C}, t, \rho) \models \exists i f(i)$ iff there exists an $K_1 \in \mathcal{K}$ such that $(\mathcal{C}, t, \rho') \models f(K_1)$ where ρ' is a restriction of the domain of ρ to $(free\text{-}var(f(K_1)) \cup \mathcal{T})$.

For convenience of expression we introduce derived operators $\vee, \Diamond, \Box, \forall, \triangleleft$ which are extensions of the basic FTPL operators.

$$f \vee g \equiv \neg(\neg f \wedge \neg g)$$

$$\Diamond_{[t_1,t_2]} f \equiv True\ \mathbf{U}_{[t_1,t_2]}\ f$$

$$\Box_{[t_1,t_2]} f \equiv \neg\Diamond_{[t_1,t_2]}\neg f$$

$$\forall i f(i) \equiv \neg\exists i \neg f(i)$$

$$\exists i \triangleleft G\ f(i) \equiv \exists i(resolve(G, i) \wedge f(i))$$

$$\forall i \triangleleft G\ f(i) \equiv \forall i(resolve(G, i) \supset f(i))$$

$\vee$ is the standard propositional operator. $\Diamond, \Box$ are the standard temporal logic operators. $\forall$ is the universal quantifier. $\triangleleft$ is the name resolution operator. $\Diamond_{[t_1,t_2]} f$ states whether f is true during any time instance from t_1 to t_2. $\Box_{[t_1,t_2]} f$ states whether formula f is true throughout time interval t_1, t_2. Intuitively the name resolution operator $\triangleleft$ resolves a name (local or extended) to its principals according to the SDSI name resolution principles and then evaluates the formula over the restricted domain.

Given a formula f with free variables and a set of certificates $\mathcal{C}$ and a time instance t, we interpret the formula f as a query returning a set of tuples, which are evaluations satisfying the formula f. For example $authorize(R, x, 1, \{r\})$, where x is a free variable, denotes a query which returns all the principals x who have the permission to delegate right r of the resource R at the current instance of time.

B. Using the Logic to analyze the state of a given SPKI system

All the formulas are evaluated in the context of a given certificate set $\mathcal{C}$ and with respect to a particular resource R unless otherwise specified. Constructing the required set of certificates $\mathcal{C}$ for evaluating a formula in a distributed environment is non-trivial. Like others [CEE+01], [JR02] we assume that we have the required set of certificates to evaluate the formulas of the logic. Distributed database lookup and goal-based certificate discovery methods seem to be promising approaches in retrieving the required set of certificates dynamically, especially for the SDSI part of the system [LWM01].

From the perspective of a resource administrator a practical guideline seems to be to cache all the certificate chains which are presented as a proof of compliance by the access requesting clients. The resource administrator may cache these certificates in a secure storage and use the FTPL logic as a means of offline Policy Audit. Though the view that certificates don't exist till they are used seems to be suited for policy audit, it can have unanticipated results as we show with an example later in the section. The language FTPL can be used in two analysis modes with respect to time. By defining the origin of time "0" to be current time instance, we can reason about the current state of the policy. For auditing a set of policy statements made from a particular time t in the past we can time shift the origin of time to t. We now illustrate the use of language to specify policy analysis problems.

This formula specifies that at every instance during the period $[t_1, t_2]$ atleast one principal in the set defined by fully qualified name G has 'w' access to resource R (Group availability property):

$$\Box_{[t_1,t_2]}\exists i \triangleleft G\ authorize(R, i, 0, \{w\})$$

This formula specifies whether a blacklisted principal K_B ever had 'read' access to a resource R during the time of audit $[0, t_a]$. Here "0" refers to the start time of the audit, and "t_a" the end time of the audit:

$$\Diamond_{[0,t_a]} authorize(R, K_B, 0, \{read\})$$

This formula specifies that principal K will be able to delegate 'read' right for the file R in future:

$$\Diamond_{[0,\infty]} authorize(R, K, 1, \{read\})$$

This formula specifies that given resource R and name G, there is an authorization chain granting G permissions to perform action 'w' on R at current time:

$$authorize(R, G, 0, \{w\})$$

This formula specifies that given resource R and name G, all the principals denoted by G are authorized to perform action 'w' on R at current time:

$$\forall i \triangleleft G(authorize(R, i, 0, \{w\}))$$

Note that the later formula implies the former, but not vice versa.

This formula specifies that both the principals K_1 *and* K_2 have simultaneous 'write' access to a resource R at some time in the future(mutual exclusion):

$$\Diamond_{[0,\infty]}[(authorize(R, K_1, 0, \{write\}) \wedge$$
$$authorize(R, K_2, 0, \{write\}))]$$

This formula specifies that both the principals K_1 *and* K_2 have 'write' access to a resource R at some time in the future:

$$\Diamond_{[0,\infty]}(authorize(R, K_1, 0, \{write\}) \wedge$$
$$\Diamond_{[0,\infty]} authorize(R, K_2, 0, \{write\}))$$

This formula specifies that a principal belonging to local name $K_{foo}Accountant$ is directly/indirectly granting write permissions for a resource at some point of time to a principal of local name $K_{foo}\,Purchaser$ (conflict of interest).

$$\Diamond_{[0,\infty]} \exists i, j(authorize(i, j, 0, \{write\}) \wedge$$
$$resolve(K_{foo}\,Accountant, i) \wedge$$
$$resolve(K_{foo}\,Purchaser, j))$$

Given resource R , this query returns all the principals authorized to perform actions 'r' and 'w' on R at time t_a:

$$\Diamond_{(t_a, t_a)} authorize(R, x, 0, \{r, w\}).$$

Where x is the free variable which returns all valuations satisfying the above formula.

This formula specifies whether principal K_1 can access resource R before K_2:

$$\neg authorize(R, K_2, 0, \{r\})\ \mathbf{U}_{[0,\infty]}$$
$$authorize(R, K_1, 0, \{r\})$$

All the previous properties and queries assumed that we evaluate the formula against a single certificate system $\mathcal{C}$. In the following query we need to consider different set of certificates for different subcomponents of the formula.

This query returns all the principals who will be excluded from performing action 'w' currently on the resource R if all the certificates issued by a compromised key K, $\mathcal{C}_K \subseteq \mathcal{C}$ are revoked now.

$$\{x|\ (\mathcal{C}, 0, \emptyset) \models authorize(R. x, 0. \{w\})\} -$$
$$\{x|\ (\mathcal{C} - \mathcal{C}_K, 0, \emptyset) \models authorize(R, x, 0, \{w\})\}$$

Certain problems like *"is there a authorization chain from the resource R to a principal K without involving the compromised key K'"*, are not directly expressible in the logic FTPL. By using the above mentioned method we can still answer such questions.

Reasoning about roles in a SPKI system.
Li [Li00]states that local names in the SPKI framework can be interpreted as "distributed roles". Distributed in the sense that they are not specified by a centralized authority in the traditional sense of the roles. Roles are thought of as an abstraction for a set of users and a set of permissions, they can include other roles too [San98]. Roles can be implemented in a SPKI system using local names by giving permissions directly to local names instead of principals. Principals inherit privileges by virtue of being members of a "local name". This approach is useful to simplify policy specification in many real-world scenarios. We can use FTPL to reason about the interpretation of local name as roles. The predicate $resolve$ can be used to reason about role membership and role hierarchy specified in a distributed fashion. The following formula specifies that role corresponding local name R_2 dominates that of R_1 : $resolve(R_1, R_2)$

Static separation of duty: This formula specifies that two roles R_1 and R_2 have the same user as a member in both the roles at same time.

$$\Diamond_{[0,\infty]} \exists i((resolve(R_1, i) \wedge resolve(R_2, i))$$

Containment: This formula specifies that there is a principal not contained by role R having 'r' access to a resource P at any point of time:

$$\Diamond_{[0,\infty]} \exists i\ (authorize(P, i, 0, \{r\}) \wedge \neg\, resolve(R, i))$$

We can also use FTPL to reason about trust violations in a SPKI system. Consider the following scenario :
A resource K_{Univ} in a university, which the students in CS and ECE dept need to have read access, some teaching assistants and special students(say research assistants) in CS dept also need to have a write access.

But no non-CS student is supposed to have a write access to the resource. Given that it is not possible to specify negative permissions in a SPKI system the resource administrator has to trust that the principals with delegation authority will not violate the university policy.

$(K_{Univ}, K_{CSDept}students, \{read\}, 0, \{t_1, t_2\})$
$(K_{Univ}, K_{ECEDept}students, \{read\}, 0, \{t_1, t_2\})$
$(K_{Univ}, K_{CSDept}, \{read, write\}, 1, \{t_1, t_2\})$: So that the CS dept admin can give write permissions to selected TA's and some special CS students.
$(K_{CSDept}, K_{TA1}, \{read, write\}, 0, \{t_1, t_2\})$
$(K_{CSDept}, K_{TA2}, \{read, write\}, 1, \{t_1, t_2\})$: TA2 can give write permission to some special CS students.
$(K_{CSDept}, students, K_{studenti}, \{t_1, t_2\})$: Name Certificate for CS student 'i' including TA's and special CS students.

In the above system the CS admin or TA2 can violate the "university policy" by giving permissions to a non-CS student.

The university resource admin wants to check whether there is a Non-CS student having write access to the resource.

$$\Diamond_{[0,\infty]}\exists i\,[(authorize(K_{Univ}, i, 0, \{write\}) \wedge \neg resolve(K_{CSDept}students, i)]$$

Evaluating a FTPL formula on a partial set of certificates.
In the case where we evaluate the above formula over a set of certificates obtained by caching the previous access request chains we may get a false-positive. Consider the scenario where TA2 from start uses the following chain to get resource authorization,

$$[(K_{Univ}, K_{CSDept}, \{read, write\}, 1, \{t_1, t_2\}) \circ (K_{CSDept}, K_{TA2}, \{read, write\}, 1, \{t_1, t_2\})]$$

then the name cert validating TA2 to be a student of CSDept is not cached by the resource administrator leading to a false positive. When the resource administrator catches a possible policy violation he can use manual resolution or a goal-directed procedure to confirm the result. The other type of violations are caused by lack of authorization certs, for example when a principal grants permissions to a blacklisted principal, but the blacklisted principal doest not access the resource and no chain involving the blacklisted principal is sent as a proof of authorization, then the corresponding formula evaluated over the cached set of certs returns a false answer. But this is a benign violation in the sense that though a certificate has been issued in violation of the policy, that certificate has not been used. In this case the serious problems which arise because of evaluating a formula on a partial state of the system are due to lack of name certificates. Goal directed algorithms for retrieving the distributed set of SDSI name certificates already exist in the literature [LWM01].

V. Evaluating the formulas

In this section, we give an algorithm that takes a certificate system $\mathcal{C}$ and a FTPL formula f and outputs a set of pairs of the form (ρ, t) such that f is satisfied with respect to the evaluation ρ at time t, i.e., $(\mathcal{C}, t, \rho) \models f$. For ease of presentation of the algorithm, we assume that there is only one resource and one access right. It can be easily generalized to the case of multiple access rights. If f has k free variables, the algorithm actually computes a relation R_f which is a set of $(k+1)$-tuples such that the first k values in each tuple define an evaluation ρ and the last value in the tuple is a list of time intervals such that for all instances t in these time intervals $(\mathcal{C}, t, \rho)$ satisfies f. Actually, we compute the relations R_g for each subformula g of f. These relations are computed inductively in increasing lengths of g. In the base case, g is an atomic subformula which is of the form $authorize(i, j, d, e)$ or is of the form $resolve(p, q)$. Recall that for the atomic formula $authorize(i, j, d, e)$, i can be a variable or a key, and j can be a variable or a term. We assume that j is a variable or a key (if j is a term then by introducing new keys and new rules we can reduce it to a case where j is a key. For example in order to evaluate a predicate $authorize(K_1, K_2\,A_1\,, d, e)$ over certificate set $\mathcal{C}$, we add new key K_3 and a new rule $K_2 A_1 \rightarrow K_3$ and evaluate $authorize(K_1, K_3, d, e)$ in the augmented system. The total number of new symbols and rules we introduce is bounded by the sum of right hand sides of the certificate rules). Similarly we assume that in $resolve(p, q)$, p can be a name and q can be a key or a variable. In the first step of our algorithm, we compute the relations R_g for the case when g is an atomic formula. In the second step, we compute the relations R_g for the case when g is not atomic.

In the first step, the algorithm operates as follows. Recall that each certificate in the set $\mathcal{C}$ is associated with a valid interval. We assume that t_{min} and t_{max}, respectively, are the minimum of the beginning points and the maximum of the end points of the validity intervals of certificates in $\mathcal{C}$. Let m be the number of certificates in $\mathcal{C}$. Using the validity intervals of the certificates, we compute a sequence of increasing times $T_0, T_1, ..., T_{l-1}$(These sequences can be easily computed from the sequence obtained by taking the begin and end

times of all the validity intervals of certificates in $\mathcal{C}$ and sorting them in increasing order) and a sequence of sets of certificates $\mathcal{C}_0$,...,$\mathcal{C}_{l-2}$ such that $l \leq 2m$, $T_0 = t_{min}$, $T_{l-1} = t_{max} + 1$ and for each i, $1 \leq i < l$, the set of certificates in $\mathcal{C}$ that are valid at every time instance in the interval $[T_i, T_{i+1} - 1]$ is the same and is given by $\mathcal{C}_i$. We can see that the complexity of the above procedure is dominated by the complexity for sorting the time points and hence is $O(m \log m)$.

For each $i = 0, ..., l-2$ we do as follows. Using the set of certificates $\mathcal{C}_i$, for each atomic formula g of f we do as follows. Consider the case when g is $authorize(p, q, d, e)$. In this case, both p, q are either a variable or a key. We compute the set $Eval_{g,i}$ of evaluations ρ for g such that the following property is satisied: if $d = 1$ then the the PDS $\mathcal{P}_{\mathcal{C}_i}$ can go from the configuration $\langle \rho(p), \square \rangle$ to the configuration $\langle \rho(q), \square \rangle$; if $d = 0$ then the PDS can go to the above configuration or to the configuration $\langle \rho(q), \blacksquare \rangle$. Note that if p is not a variable then $\rho(p) = p$; similar condition holds for q. If neither of p, q is a variable then ρ is the empty evaluation; in this case either $Eval_{g,i}$ contains the empty evluation indicating the satisfaction of g at every time instance in the interval $[T_i, T_{i+1} - 1]$ or it is the empty set indicating the non-satisfaction of g in the above interval. If g is $resolve(p, q)$ then we compute the set $Eval_{g,i}$ of evaluations ρ such that the above PDS can go from the configuration corresponding to the name p to the configuration corresponding to the term $\rho(q)$ (recall that p has to be a name and q can be a variable or a term; also, recall that the configuration corresponding to the name of the form $K\ A\ B$ is $\langle K, A\ B \rangle$).

Now it should be easy to see how R_g can be calculated from the sets $Eval_{g,i}$ for $i = 0, ..., (l-2)$. Recall that, if g has k free variables then R_g is a set of $(k+1)$ tuples (here $k \leq 2$). For each evaluation ρ that appears in at least some $Eval_{g,i}$, R_g has a single tuple whose first k components give the evaluation ρ and whose $(k+1)^{st}$ component is the list of all time intervals $[T_i, T_{i+1} - 1]$ such that ρ is in $Eval_{g,i}$.

In the above procedure, we defined the sets $Eval_{g,i}$ using a PDS $\mathcal{P}_{\mathcal{C}_i}$. However, we plan to employ a And-or graph from which the sets $Eval_{g,i}$ can be computed for all g using a single fix point computation.

Now we describe the And-or graph construction for a set $\mathcal{D}$ of certificates. First using the certificate set $\mathcal{D}$. we first define a nondeterministic normalized pushdown system $\mathcal{P}$. from which we define an equivalent context free grammar $\mathcal{G}$, which is converted in to an And-Or graph $\mathcal{H}$. The advantages of using the And-Or graph over the PDS model of Jha and Reps [JR02] is that we have the reachability relationships between various principals in one structure from which the required relations can be computed efficiently. The And-or graph $\mathcal{H}$ can be computed directly from $\mathcal{C}$ without constructing either of $\mathcal{P}$ or $\mathcal{G}$. However, their definition gives a better intuition leading to an easy proof of correctness of the algorithm.

The given certificate system $\mathcal{D}$ is expressed as a set of rewriting rules, the name certs correspond to the rules of form $K_1\ A \rightarrow K_2\ A_1\ A_2\ ...A_n$. The auth certs correspond to rules of the form $K\square \rightarrow K_1 A_1 A_2 ..A_n \square$. Corresponding to each key K we introduce two symbols $K^{\square}, K^{\blacksquare}$. $\mathcal{K}^{\#}$ is the augmented set of keys consisting of the original keys and their associated symbols.

Given the set $\mathcal{D}$ of cert rules, all the rules will be converted to a normalized transition function δ of a pushdown system. The pushdown system $\mathcal{P}$ we consider is a three tuple (Q, Γ, δ). Q is the set of states as given below, Γ is the stack alphabet containing identifiers and delegation symbols of a SPKI system, $\delta \subseteq Q \times \{\Gamma \cup \{\epsilon\}\} \times Q \times \{\Gamma \cup \{\epsilon\}\}$ is the transition function. δ can have transitions either of type $(K_1, A_1, K_2, \epsilon)$ or $(K_1, \epsilon, K_2, A_1)$, which means that PDS can go from state K_1 to state K_2 either by popping or pushing a symbol $A_1 \in \Gamma$ on top of the stack. ϵ is the empty string character. Thus in each transition a symbol is pushed or is popped from the stack and both do not occur in the same transition. Let l_i be the length of right hand side of rule i, $L_{\mathcal{D}}$ the sum of the lengths of the right hand side of the cert rules.

$Q = \mathcal{K}^{\#} \cup \{(i,j) |\ 1 \leq i \leq |\mathcal{D}|, 1 \leq j \leq \ell_i\}$, $\Gamma = \mathcal{I} \cup \{\square, \blacksquare\}$. Each rewriting rule is converted into a set of four tuples in δ. If we have the i'th cert(name or auth) with length ℓ_i as $K_i\ A_i \rightarrow K_i'\ m_{(i,1)} ... m_{(i,\ell_i)}$; $A_i, m_{(i,1)} ... m_{(i,\ell_i)} \in \Gamma$, then it will be converted to normalized tuples of set δ as follows:

For each K, the PDS when in state $K^{\square}$ pushes $\square$ onto the stack and goes to state K.

$$(K^{\square}, \epsilon, K, \square) \in \delta.$$

In a state K, the PDS non-deterministically chooses a rule i whose left hand side key is K and and the left hand side symbol(identifier or delegation bit) matches the symbol on top of the stack, it pops the symbol currently on top of the stack and goes to state (i, l_i) if $l_i > 0$ or else if $l_i = 0$ to state K_i' given by the right hand side key of rule i.

$$(\ K, A_i, (i, \ell_i), \epsilon\ ) \in \delta.$$

$$(\ K, A_i, K_i', \epsilon) \in \delta$$

In a state of the form (i, j) it pushes the j'th identifier of the right hand side of the i'th cert and goes to state

$(i, j-1)$ if $j > 1$, else if $j = 1$ it goes to the state K'_i given by the right hand side key of rule i.

$$((i,j), \epsilon, (i, j-1), m_{(i,j)}) \in \delta \; for \; all \; 2 \leq j \leq \ell_i.$$

$$((i,j), \epsilon, K'_i, m_{(i,j)}) \in \delta \; when \; j = 1.$$

When the PDS is in state of type K and the top of the stack symbol is either $\square$ *or* $\blacksquare$ then it goes to state $K^{\square}$ or $K^{\blacksquare}$ respectively.

$$(K, \square, K^{\square}, \epsilon) \in \delta, \; (K, \blacksquare, K^{\blacksquare}, \epsilon) \in \delta$$

We can see that in a SPKI system $\mathcal{D}$ a key K_1 gives authorization to another key K_2 if the PDS $\mathcal{P}$ when started in state $K_1^{\square}$ on an empty stack reaches either of the states $K_2^{\square}$ or $K_2^{\blacksquare}$ with the stack empty at the end.

It is easy to see that size of δ is $O(L_{\mathcal{D}} + |\mathcal{D}|)$ and size of Q is $O(|\mathcal{K}| + L_{\mathcal{D}})$.

From the normalized PDS $\mathcal{P}$ we construct an equivalent context free grammar $\mathcal{G}$ according to the rules given in [Sip01]. The variables (i.e., non-terminals) of $\mathcal{G}$ are $\{A_{pq} | p, q, \in Q\}$. The production rules of $\mathcal{G}$ are described below.

For each $p, q, r, s \in Q$ and $A \in \Gamma$, such that δ contains the transitions (p, ϵ, r, A), (s, ϵ, A, q) (i.e., the first transition pushes A onto the stack while the second pops the same symbol from the stack) we have the rule $A_{pq} \rightarrow A_{rs}$ in $\mathcal{G}$.

For each $p, q, r \in Q$ we have the rule $A_{pq} \rightarrow A_{pr} A_{rq}$ in $\mathcal{G}$.

For each $p \in Q$, we have the rule $A_{pp} \rightarrow \epsilon$ in $\mathcal{G}$.

The only terminal symbol of the grammar is the null string ϵ. Let $Q' \subset Q$ be the set $\mathcal{K} \cup \{(i, l_i) | \; 1 \leq i \leq |\mathcal{D}|\}$. Due to the semantics of SPKI, it is easy to see that among the rules of type $A_{pq} \rightarrow A_{pr} A_{rq}$, the ones that are useful are those in which $q, r \in Q'$. It is easy to see that the number of rules of this type is bounded by $O((|\mathcal{K}| + L_{\mathcal{D}})(|\mathcal{K}| + |\mathcal{D}|)|\mathcal{D}|)$. The total number of rules is $O((|\mathcal{K}| + L_{\mathcal{D}})(|\mathcal{K}||\mathcal{D}| + |\mathcal{D}|^2))$. The CFG $\mathcal{G}$ has the following property. The PDS $\mathcal{P}$ can go from state p to q starting and ending in a empty stack iff we can generate the empty string ϵ from CFG symbol A_{pq}. The term reachability problem of the PDS has been converted to a word problem in the CFG.

From the above grammar $\mathcal{G}$ we construct a directed And-Or graph $\mathcal{H} = (V, E)$. The vertex set V is a disjoint union of two sets V_1, V_2. V_1 is the set of all pairs of the form $[p. q]$ where $p \in Q$ and $q \in Q'$. V_2 is the set of all triples of the form $[p, q, r]$ where $p \in Q$ and $q. r \in Q'$. Each vertex in V_1 is an "or" vertex while each vertex in V_2 is an "and" vertex. The set E of edges is defined as follows. From each node of the form $[p, r, q]$ there are exactly two edges going to the vertices $[p, r]$ and $[r, q]$. For each rule of the form $A_{pq} \rightarrow A_{rs}$ in $\mathcal{G}$, we have an edge from ($[p, q]$ to $[r, s]$). For each rule of the form $A_{pq} \rightarrow A_{pr} A_{rq}$ in $\mathcal{G}$, we have an edge from $[p, q]$ to the vertex $[p, r, q]$. Note that there may be cycles in the graph.

Now, we compute a function $F : V \rightarrow \{True, False\}$ which is the least fix point that satisfies the following conditions: for each vertex u of the form $[p, p]$, $F(u) = True$; for each vertex u of the form $[p, q]$ $(p \neq q)$, $F(u)$ is the disjunction of all $F(v)$ such that $(u, v) \in E$; for each vertex $u \in V_2$, $F(u)$ is the conjunction of all $F(v)$ such that $(u, v) \in E$. It is well known that this fix point can be computed in time linear in the size of $\mathcal{H}$ using a simple iterative approach as follows. With each vertex u, we maintain a variable $label(u)$ which is initialized to $True$ for vertices of the form $[p, p]$ and is initialized to $False$ for all other vertices. We also maintain a counter $c(u)$ with each vertex u which is initialized to zero for all vertices. The algorithm also maintains a set X of vertices. Initially, X is the set of nodes of the form $[p, p]$. After this we iterate the following procedure as long as X is non-empty. We delete a node v from X, examine all nodes u from which there is an edge to v; if $u \in V_1$ (i.e.,is an or-node) and $c(u)$ is zero then we increment $c(u)$, set $label(u)$ to $True$ and add u to the set X; if $u \in V_2$ (i.e., is an "and" node) and $c(u) < 2$ then we increment $c(u)$, further if $c(u) = 2$ after this increment, we set $label(u)$ to $True$ and add u to X.

It is easily shown that at the end of the above algorithm, for any vertex u of the form $[p, q]$, $label(u)$ is $True$ iff the contex free grammar $\mathcal{G}$ can generate the empty string ϵ from the non-terminal A_{pq}. Using this and the results of [Sip01], the following theorem is easily proved.

Theorem: At the end of the above algorithm, for any vertex u of the form $[p, q]$, $label(u) = True$ iff the PDS when started in state p with empty stack reaches state q with empty stack.

The size of $\mathcal{H}$ which is $(|V| + |E|)$ can be shown to be $O((|\mathcal{K}| + L_{\mathcal{D}})(|\mathcal{K}||\mathcal{D}| + |\mathcal{D}|^2))$. Thus the complexity of the above fix point computation is $O((|\mathcal{K}| + L_{\mathcal{D}})(|\mathcal{K}||\mathcal{D}| + |\mathcal{D}|^2))$.

Recall that, at the beginning of the section, we described a method for computing relation R_g for each atomic formula g. This procedure used a sequence of sets of certificates $\mathcal{C}_0, \mathcal{C}_1, ..., \mathcal{C}_{l-2}$ where $l \leq 2$,.

R_g is constructed by computing $Eval_{g,i}$ for $i = 0...l-2$. It should be easy to see that, for any fixed i, $Eval_{g,i}$

for all g can be computed by constructing the And-or graph corresponding to the set of certs $\mathcal{C}_i$.

$Eval_{g,i}$ can be computed from the And-or graph $\mathcal{H}_i$ corresponding to the set of certificates $\mathcal{C}_i$ as follows.

Assume g is $authorize(p, q, d, e)$. If both p, q are variables and $d = 1$ then $Eval_{g,i}$ is the set of all pairs (K_1, K_2) such that the vertex $(K_1^{\square}, K_2^{\square})$ is labeled $True$ in graph $\mathcal{H}_i$. If $d = 0$ then we check if either the above vertex or the vertex $(K_1^{\square}, K_2^{\blacksquare})$ are labeled $True$ in $\mathcal{H}_i$. If both p, q are keys say K_1, K_2 and $d = 1$, then $Eval_{g,i}$ is the singleton set containing the empty evaluation if the vertex $(K_1^{\square}, K_2^{\square})$ is labeled $True$; otherwise, it is the empty set. Other cases are similarly handled.

Assume g is $resolve(p, q)$. In this case p is a name and q can be a key or a variable. We can assume that p is the prefix of the right hand side of some rule j in $\mathcal{C}_i$(If this is not the case then we can add a dummy rule whose right hand side is p). Assume that the length of p(number of identifiers) is k.
If q is a key then $Eval_{g,i}$ is the singleton set containing the empty evaluation if the node $[(j, k), q]$ in $\mathcal{H}_i$ is labeled $True$; otherwise, it is the empty set. If q is a variable then $Eval_{g,i}$ is the set of all K_1 such that the node $[(j, k), K_1]$ in $\mathcal{H}_i$ is labeled $True$.

In the above computation if $\mathcal{C}_{i+1} \supseteq \mathcal{C}_i$, then the graph $\mathcal{H}_{i+1}$ is obtained by adding new edges to $\mathcal{H}_i$, corresponding to additional certs in $\mathcal{C}_{i+1}$. The fixpoint computation on $\mathcal{H}_{i+1}$ can start with the labels computed for $\mathcal{H}_i$ and continue with the additional edges till a new fixpoint is reached. Thus we can take advantage of the incremental nature of the fix point computation, with respect to addition of edges to the And-or Structure. It is to be noted that if $\mathcal{C}_{i+1} \subseteq \mathcal{C}_i$, then $\mathcal{H}_{i+1}$ is obtained from $\mathcal{H}_i$ by deleting some edges. However we cannot apply incremental computation while deleting edges in the graph $\mathcal{H}_i$, as the lest fixpoint computation is not incremental with respect to the deletion of edges in the And-or structure.

Since $|\mathcal{C}_i| \leq m$, from the above analysis we can see that R_g, for any atomic g, can be computed in time $O(m(|\mathcal{K}| + |L_{\mathcal{C}}|)(\mathcal{K}m + m^2))$.

In our algorithm presented below for computing R_f, we need to maintain lists of intervals of time. A list of intervals is *maximal* if every pair of intervals in the list is non-overlapping and non-adjacent (two time intervals $[s, s'], [t, t']$ are non-overlapping and non-adjacent if $t > s' + 1$ or $s > t' + 1$). We maintain all lists in such a way that they are maximal and the intervals are in sorted order. We call such lists as normalized. We define the size of such a list as the number of intervals in it. All our lists are in normalized form and all operations preserve this property. The union of two normalized lists L_1 and L_2 is another normalized list covering exactly the union of the time points covered by the two lists. The intersection of two normalized lists is a normalized list that covers exactly the time points common to both of the lists. The union and intersection of two normalized lists can be computed in time proportional to the sum of the sizes of the two lists. The complement of a normalized list L_1 is a normalized list that covers exactly all time points in the interval $[0, \infty]$ that are not covered by L_1. The complement of a normalized list can also be computed in time linear in the size of the list.

Now we give the second part of the algorithm based on structural induction for computing relation R_g of a subformula g when g is not atomic. The algorithm computes relation R_g for each subformula g of f in the increasing lengths of the subformula g. On termination of the algorithm we have the required relation R_f. We call the list in each tuple of the relation R_g as the validity list of the tuple. We want to show that the length of the validity list in each tuple of in R_g is of length $O(m|g|)$. To show this, we first define a finite set of time points (i.e., positive integers), called $extreme\_points(g)$, such that the beginning and ending time points of each interval in the validity lists of tuples in R_g belong to $extreme\_points(g)$. The set $extreme\_points(g)$ is defined inductively on the structure of g. If g is an atomic proposition then $extreme\_points(g)$ is the set of points $\{T_i, T_i + 1 : 0 \leq i < l\}$ as given at the beginning of this section. It is to be noted the cardinality of this set is $O(m)$. If $g = h \wedge h'$ then $extreme\_points(g)$ is the union of $extreme\_points(h)$ and $extreme\_points(h')$. If $g = \neg h$ then $extreme\_points(g)$ is same as $extreme\_points(h)$. If $g = \mathbf{X}h$ then $extreme\_points(g)$ is the set $\{0, t - 1 : t \in extreme\_points(h)\}$. If $g = h\mathbf{U}_{[t_1,t_2]}h'$, then $extreme\_points(g)$ is the set $\{t - t_1, t - t_2, t + 1 - t_1, t + 1 - t_2 : t \in extreme\_points(h)$ or $t \in extreme\_points(h')\}$. If $g = \exists\, i(h)$ then $extreme\_points(g)$ is same as $extreme\_points(h)$. By induction on the length of g, it can be shown that the cardinality of $extreme\_points(g)$ is $O(m|g|)$. In the construction of R_g given below, it should be easy to see that the begin and end points of each interval in the validity list of each tuple in R_g is from the set $extreme\_points(g)$. Since the intervals in a validity list are disjoint, each point in $extreme\_points(g)$ can appear as the end point of atmost one interval. Hence, the length of each such validity list is $O(m|g|)$.

Let $|R_g|$ denotes the number of tuples in the relation R_g corresponding to a formula g. As shown above, the length of any validity list of a tuple in R_g is of $O(m|g|)$

where m is the number of certificates. Since the number of free variables appearing in g is at most $|g|$, we see that the size of any single tuple including its validity list is $O(|g| + m|g|)$ which is $O(m|g|)$. Hence, the size of R_g is $O(m|R_g||g|)$.

If the subformula g is of the type $\neg h$, then for every tuple of form $(K_1, K_2, ..., K_r, L) \in R_h$ we include the tuple $(K_1, K_2, ..., K_n, \overline{L})$ in R_g. $\overline{L}$ is the complement of the validity list L as defined earlier. As shown previously we can compute $\overline{L}$ in time linear in the size of the list L. Hence we can compute R_g in time $O(m|R_h||h|)$.

Consider a subformula $g = h \wedge h'$, where $R_h, R_{h'}$ are the relations computed for formulas h and h', having $(i+1)$ and $(j+1)$ attributes respectively. If h, h' have v number of common variables, then R_g has $(i+j-v+1)$ attributes. For a given instantiation ρ if h is satisfied during an interval I (in the validity list) and h' during I', g is satisfied during time $I \cap I'$. For a tuple t_1 in R_h and a tuple t_2 in $R_{h'}$, we include a tuple t_{12} in R_g, if the corresponding values of of the common variables in the two tuples are equal and the intersection of the corresponding validity lists is not empty. We include in the tuple t_{12} all the values related to the variables(without repeating the common variable values) and the validity list in the tuple t_{12} is given by intersection of the validity lists in t_1 and t_2. Given that we can compute the intersection of two validity lists in time linear in the sum of their sizes, we can compute R_g in time $O(m|R_h||R_{h'}|(|h| + |h'|))$.

If the subformula is of the type $g = \mathbf{X}h$, then for every tuple in R_h of the form $(K_1, ...K_r, L)$, we include the tuple $(K_1, ...K_r, L')$ in R_g where L' is as defined below; L' consists of all intervals of the form $[max(t_i - 1, 0), t_j - 1]$ such that $[t_i, t_j] \in L$. We can compute L' in time linear in the size of L. Hence we can compute R_g in time $O(m|R_h||h|)$.

If the subformula is $g = \exists i h(i)$, then R_g is computed as follows. Let $r + 1$ be the number of attributes of the relation R_h. With out loss of generality, assume that the j^{th} attribute of R_h gives the value of the variable i. Note that the values of the last attribute is the validity list of the tuple. We group the tuples of R_h into the smallest collection of groups $G_1, .., G_l$ such that each group G_p satisfies the following property: all the tuples in G_p have the same values for the q^{th} attribute, for each q such that $q \neq j$ and $1 \leq q \leq r$. Corresponding to each group G_p, R_g has a single tuple $(K_1, .., K_{j-1}, K_{j+1}, ..., K_r, L)$ where K_q, for $1 \leq q \leq r$ and $q \neq j$, is the value of the q^{th} attribute in a tuple in G_p and L is the union of all the validity lists in the tuples in G_p; it is to be noted that the list L can be computed in time linear in the sum of the sizes of the validity lists in the tuples of G_p. Hence we can calculate R_g in time $O(m|R_h||h|)$.

Consider a subformula $g = h\mathbf{U}_{[t_1,t_2]}h'$, where $R_h, R_{h'}$ are the relations computed for formulas h and h', having $(i+1)$ and $(j+1)$ attributes respectively. If h, h' have v number of common variables, then R_g has $(i+j-v+1)$ attributes. For a given instantiation ρ of values if h is satisfied during the times given by validity list L_1, h' is satisfied during the time intervals given by the validity list L_2, we give a method to calculate L_{12} the list of time intervals during which g is satisfied. We define two intervals $[p_1, p_2] \in L_1, [q_1, q_2] \in L_2$ as compatible if they overlap or if $[p_1, p_2], [q_1, q_2]$are adjacent i.e. $q_1 = p_2 + 1$. To compute L_{12} take two compatible intervals $I_1 = [p_1, p_2] \in L_1, I_2 = [q_1, q_2] \in L_2$, then by the definition of the bounded until operator we can show that the corresponding interval $I_{12} \in L_{12}$ during which the formula g holds good is $[max(T - t_2, p_1), max(T - t_1, p_1)]$ where $T = min(p_2 + 1, q_2)$. Thus we compute all the intervals in L_{12} from the compatible intervals of L_1 and L_2. Given that we maintain the validity lists in a normalized condition we can calculate L_{12} in time linear in the sum of sizes of validity lists L_1 and L_2. For a tuple t_1 in R_h and a tuple t_2 in $R_{h'}$, we include a tuple t_{12} in R_g, if the values of tuples corresponding to the common variables are equal and the composition of corresponding validity lists as defined above is not empty. We include in the tuple t_{12} all the values related to the variables(without repeating the common variable values) and the validity list in the tuple t_{12} is given by composition of corresponding validity lists in t_1 and t_2 as defined above. We can compute R_g in time $O(|R_h||R_{h'}|m(|h| + |h'|))$.

Thus we calculate the relation R_f for the formula f inductively from the components of the formula. Let n be the total number of principals in the system, L the sum of lengths of right hand side of all certificates and k the number of variables in the formula f. For any formula $g(i_1, ..., i_{l_i})$ with l_i number of variables the size of relation R_g is of order $O(n^{l_i} m|g|)$ because each of the variables can be any of the n principals in the system. The normalized validity list of a tuple in R_g can be maintained in the size of $O(m|g|)$. The relation R_g obtained by composition of relations R_h and $R_{h'}$ can be computed in time $O(|R_h||R_{h'}|m(|h| + |h'|))$. If h contains l_i variables and h' contains l_j variables, R_g can be computed in time $O(n^{l_i} n^{l_j} m(|h| + |h'|)) = O(n^{l_i + l_j} m(|h| + |h'|))$. Thus the overall formula can be computed in time $O(n^{l_i + l_j + \cdots} m(|h| + |h'| + ...)) = O(n^k m|f|)$. Since the size of formula $|f| \approx k$ the complexity of evaluating the formula f given the relations for the atomic formulas $authorize$ and $resolve$ is $O(n^{|f|} m|f|)$. The overall complexity of evaluating the formulas of the logic FTPL is $O(n^{|f|} m|f|) + O(m(n + L)(nm + m^2))$.

VI. Conclusion and Future Work

In this paper, we have proposed a language for analyzing a set of policy statements labeled with validity intervals and gave a list of problems that could be specifiable by the language. We also gave algorithms for computing the formulas of the logic in a incremental fashion. In future we propose to modify the semantics of the modal operators of FTPL to consider a state transition model for the SPKI access control system.

References

[Aba97] Martin Abadi. On SDSI's linked local name spaces. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

[ADT02] A. Arsenault, Diversinet, and S. Turner. *Internet X.509 Public Key Infrastructure: Roadmap*. PKIX working group-Internet Draft, 2002.

[BEO97] A. Bouajjani, J. Esparza, and O.Maler. Reachability analysis of pushdown automata: application to model-checking. *Lecture Notes in Computer Science*, 1243, 1997.

[BFIK99] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The role of trust management in distributed systems security. pages 185–210, 1999.

[BFK99] Matt Blaze, Joan Feigenbaum, and Keromytis Keromytis. KeyNote: Trust management for public-key infrastructures. In Bruce Christianson, Bruno Crispo, William S. Harbison, and Michael Roe, editors, *Security Protocols—6th International Workshop*, volume 1550 of *Lecture Notes in Computer Science*, pages 59–66, Cambridge, United Kingdom, 1999. Springer-Verlag, Berlin Germany.

[CEE+01] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4), 2001.

[EFL+99] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. *RFC 2693: SPKI Certificate Theory*. The Internet Society, September 1999. See `ftp://ftp.isi.edu/in-notes/rfc2693.txt`.

[EHRS00] Esparza, Hansel, Rossmanith, and Schwoon. Efficient algorithms for model checking pushdown systems. In *CAV: International Conference on Computer Aided Verification*, 2000.

[Ell99] Carl M. Ellison. *RFC 2692: SPKI requirements*. The Internet Society, September 1999. See `ftp://ftp.isi.edu/in-notes/rfc2692.txt`.

[HK00] Jon Howell and David 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.

[HRU75] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. On protection in operating systems. In *Proceedings of the fifth ACM symposium on Operating systems principles*, pages 14–24. ACM Press, 1975.

[HvdM01] Joseph Y. Halpern and Ron van der Meyden. A logic for sdsi's linked local name spaces. *Journal of Computer Security*, 9:105–142, 2001.

[JR01] Somesh Jha and Thomas Reps. Analysis of spki/sdsi certificates using model checking. In *15th IEEE Computer Security Foundations Workshop*, pages 129–144, Cape Breton, Canada, 24–26 June 2001. IEEE Computer Society Press.

[JR02] S. Jha and T. Reps. Analysis of spki/sdsi certificates using model checking. In *Computer Security Foundations Workshop (CSFW)*, June 2002.

[Li00] Ninghui Li. Local names in SPKI/SDSI. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, pages 2–15. IEEE Computer Society Press, jul 2000.

[LM03] Ninghui Li and John C. Mitchell. Datalog with constraints: A foundation for trust management languages. In *Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages*, January 2003. To appear.

[LWM01] Li, Winsborough, and Mitchell. Distributed credential chain discovery in trust management. In *SIGSAC: 8th ACM Conference on Computer and Communications Security*. ACM SIGSAC, 2001.

[LWM03] Ninghui Li, William H. Winsborough, and Mitchell Mitchell. Beyond Proof-of-Compliance: Safety and availability analysis in trust management. In *Proceedings of the 2003 Symposium on Security and Privacy*, pages 123–139, Los Alamitos, CA, May 11–14 2003. IEEE Computer Society.

[San98] R. S. Sandhu. Role-based access control. In M. Zerkowitz, editor, *Advances in Computers*, volume 48. Academic Press, 1998.

[Sip01] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 20 Park Plaza, Boston, MA 02116, 2001.

Enhancing Security of Security-Mediated PKI by One-time ID

Satoshi Koga[1], Kenji Imamoto[1], and Kouichi Sakurai[2]

[1] Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka, 812-8581, Japan
{satoshi,imamoto}@itslab.csce.kyushu-u.ac.jp
[2] Faculty of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka City, 812-8581, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract. The Security Mediator (SEM) approach was proposed by Boneh et al. at USENIX Security Symposium in 2001. However, their Security-Mediated PKI has a drawback that it is vulnerable to Denial-of-Service (DoS) attacks, since an attacker can send a lot of requests to the SEM server. To prevent DoS attacks, some solutions are proposed. In this paper, we show that the use of Message Authentication Code (MAC), which is one of the solution proposed, cannot avoid DoS attacks because an attacker can reuse the request for replay attacks. In order to prevent DoS attacks efficiently, this paper proposes Security-Mediated PKI using one-time ID. Our proposed system can avoid DoS attacks and protect the privacy of signers, since one-time ID can be used only once.

Keywords: Public Key Infrastructure, Certificate Revocation, Security Mediator, One-time ID

1 Introduction

1.1 Background

A Public Key Infrastructure (PKI) is the basis of security infrastructure whose services are implemented and provided using public key techniques. Most of the protocols for secure e-mail, web service, virtual private networks, and authentication systems make use of the PKI. In the PKI, a certificate is used to bind an entity's identity information with the corresponding public key. When a certificate is issued, its validity is limited by a pre-defined expiration time. Nevertheless, certificates are revoked in case of breaking that binding before its expiration date. Thus, the certificate verifier must check not only the expiration date on the certificate but also the revocation information of it.

A certificate revocation system can be implemented in several ways. The most well-known method is to periodically publish a Certificate Revocation List (CRL)

[6, 8], which is a digitally signed list of revoked certificates and usually issued by the Certification Authority (CA). One of the shortcomings of CRL systems is that the time granularity of revocation is constrained by CRL issuance periods. It is necessary to obtain timely information regarding the revocation status of a certificate. The most popular mechanism that provides real-time status of a certificate is the Online Certificate Status Protocol (OCSP) [13]. The OCSP provides the up-to-date response to certificate status queries. Since the user just requests to return the status of certificate, the communication costs can be reduced in comparison to the CRL.

Recently, Boneh et al. observed that existing revocation techniques, including OCSP, don't provide immediate revocation [3]. Supporting immediate revocation with existing revocation techniques would result in heavy performance cost and very poor scalability. To provide immediate revocation, SEcurity Mediator (SEM) approach was proposed [2, 3]. The basic idea of SEM is as follows. They introduce a new entity, referred to as a SEM: an online semi-trusted server. To sign or decrypt a message, a client must first obtain a message-specific token from its SEM. Without this token, the user cannot accomplish the intended task. To revoke the user's ability to sign or decrypt, the security administrator instructs the SEM to stop issuing tokens for that user's future request. The SEM approach provides several advantages. This approach can eliminate the need for CRLs since private-key operations cannot occur after revocation. That is, this approach enables immediate revocation. Recently, Vanrenen et al. proposed a distributed SEM architecture [14], and Bicakci et al. proposed Privilege Management Infrastructure based on SEM approach [1].

1.2 Motivation

As Boneh et al. pointed out in [3], one of the drawbacks of SEM approach is that it is vulnerable to Denial-of-Service (DoS) attacks. The SEM server must generate the token for every user's request. That is, for every simple request sent by an attacker, the SEM server must generate the token, which is a computationally intensive operation. Consequently, it becomes highly vulnerable to DoS attacks. The goal of this paper is to prevent DoS attacks.

1.3 Related Work

To prevent DoS attacks, there are some solutions as follows [3].

1. Digital signatures
   The simple solution is to authenticate the user by verifying digital signature [3]. For example, a user can generate a partial signature on each request message as follows. (See Section 2.3 for notations.)

   $$\langle EC(m), EC(m)^{d_u} \mod n \rangle$$

   The SEM computes a digital signature $S(m) = (PS_{sem} * PS_u) \mod n$ and verifies it by using public key. Although this method does not prevent DoS

attacks, since a SEM would need to perform two modular exponentiations to authenticate each request [3].

2. Secure Sockets Layer (SSL)
   Another solution is to rely on more general encapsulation techniques, such as SSL, to provide a secure channel for communications between SEMs and users [3]. However, the user must validate the SEM's certificate in order to establish secure channel. Therefore, the burden of a user becomes heavy since the certificate validation processing is intensive operations.
3. Message Authentication Code (MAC)
   More cost-effective approach is to use MAC or keyed hash [3]. This situation requires a shared secret key between a SEM and each user. This paper shows that this approach cannot prevent DoS attacks. An attacker can reuse requests generated by legal users, and send a lot of requests to a SEM. Even if the SEM authenticates requests by checking MAC, it is vulnerable to DoS attacks based on replay attacks, as discussed in Section 3.
4. The DoS resistant protocol
   To avoid DoS attacks from any malicious stranger, it is important to make him have a large burden in sending a lot of requests. One approach to solve the DoS problem is to make the client compute some form of proof of computational effort [5, 12]. This approach is effective to DoS attacks, however computational costs of the legal user are also increasing as well as DoS attackers. In our proposed method, only attackers require exhaustive computations.

1.4 Our Contributions

This paper shows that traditional solutions cannot prevent DoS attacks. An attacker can reuse requests generated by legal users, and send a lot of requests to a SEM. Even if the SEM authenticates requests by checking MAC, it is vulnerable to DoS attacks based on replay attacks.

It is necessary to avoid DoS attacks based on replay attacks efficiently. This paper proposes the SEM approach using One-time ID. One-time ID is a user's extraordinary identity, which has two properties as follows: (1) an attacker cannot specify who is communicating even when she eavesdrops on one-time ID, and (2) One-time ID can be used only once. Our proposed method requires a shared secret key between a SEM and each user. A user sends a request containing a message and one-time ID. One-time ID can be derived by shared secret key and be used only once. By checking one-time ID, a SEM server can confirm that requesting user is legal user. In our proposed method, the user sends a request containing Message Authentication Code (MAC). Therefore, an attacker cannot create the legal request. Moreover, an attacker cannot reuse requests for replay attacks because the One-time ID is used only once. Our proposed system has the following benefits.

1. DoS-resilient
   A SEM can efficiently detect illegal requests, since an attacker cannot generate one-time ID unless she obtains a shared secret key. Moreover, our

proposed system can prevent replay attacks. An attacker cannot reuse the request generated by legal user because one-time ID can be used only once.

2. Efficiency
   One-time ID can be generated by hash computations. Therefore, computational costs of a SEM and a user are more efficient, compared with signature-based solutions. Additionally, communications between a SEM and a user are the same as the existing Security-Mediated PKI.
3. Privacy
   In existing approaches, a user must send a request contained a public key certificate. Thus, an attacker can trace the user's identity by tracking user's request. It is easy to know the user's private information (e.g. user's name, affiliation, address and so on.) from the serial number of his certificate. In our proposed system, an attacker cannot trace user's identity even if she eavesdrops on one-time ID, because one-time ID dynamically changes. That is, our proposed system can protect the privacy of signers, compared with existing approaches.

The rest of this paper is organized as follows. In Section 2, we explain the SEM approach. In Section 3, we show that the traditional approach cannot prevent DoS attacks. In Section 4, we describe our proposed system and Section 5 discusses as to security of our proposed system. Concluding remarks are made in Section 6.

2 Security-Mediated PKI

2.1 Model

In a Security-Mediated PKI, there are three entities, as shown in Fig.1.

1. Certification Authority (CA)
   A Certification Authority (CA) is a trusted third party that issues certificates. Compromise of CA's private key will affect the entire system, so the CA is isolated from the Internet to prevent unauthorized accesses.
2. SEM (SEcurity Mediator)
   A SEM is an online semi-trusted server. A single SEM serves many users.
3. User
   Users trust the CA and SEM. To sign or decrypt a message, they must first obtain a message-specific token from its SEM.

2.2 An overview of a SEM

The basic idea of SEM approach is as follows [2]. We introduce a new entity, referred to as a SEM. A SEM is an online trusted server. To sign or decrypt a message, Alice must first obtain a message-specific token from the SEM. Without this token Alice cannot use her private key. To revoke Alice's ability to sign or decrypt, the security administrator instructs the SEM server to stop issuing

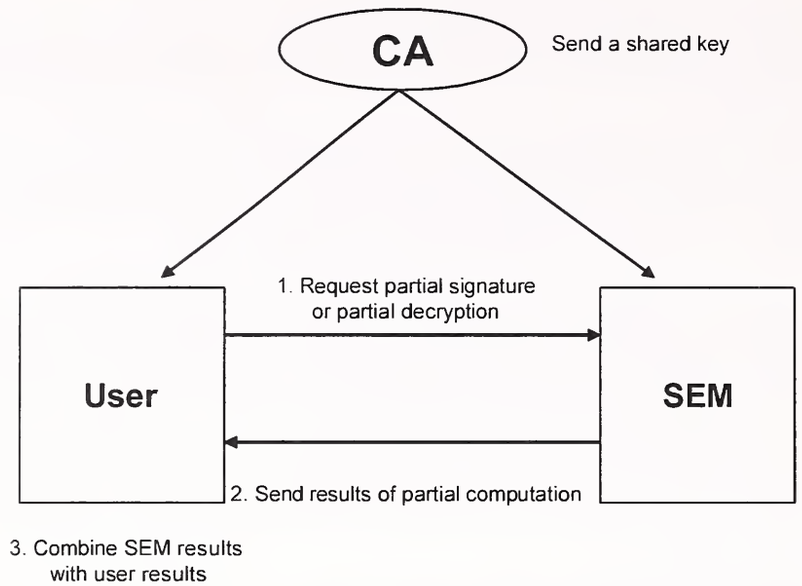


Fig. 1. A general SEM algorithm

tokens for Alice's public key. At that instant, Alice's signature capabilities are revoked. Fig. 1 shows the general SEM algorithm.

A SEM approach is to enable immediate revocation of user's key. This method provides two additional benefits over standard revocation techniques: (1) simplified signature validation, and (2) enabling revocation in legacy systems. The SEM approach naturally provides the following semantics for digital signatures:

Binding Signature Semantics: a digital signature is considered valid if the public key certificate associated with the corresponding private key used to generate the signature was valid at the time the signature was issued.

2.3 Mediated RSA

This section describes in detail how a SEM interacts with users to generate tokens. The SEM architecture is based on a variant of RSA called Mediated RSA (mRSA). The main idea is to split each RSA private key into two parts using simple 2-out-of-2 threshold RSA [4].

Each user U has a unique public key and private key. The public key PK includes n and e, where the former is a product of two large distinct primes (p, q) and e is an integer relatively prime to $\phi(n) = (p-1)(q-1)$. There is also a corresponding RSA private key $SK = (n, d)$ where $d * e = 1 \mod \phi(n)$. However, as mentioned above, no one party has possession of d. Instead, d is effectively split into tow parts: d_u and d_{sem} which are secretly held by the user

and the SEM, respectively. The relationship among them is:

$$d = d_{sem} + d_u \mod \phi(n)$$

The CA generates a distinct set: $\{p, q, e, d, d_{sem}, d_u\}$ for the user. The first four values are generated as in standard RSA. The fifth value, d_{sem}, is a random integer in the interval $[1, n]$, where $n = pq$. The last value is set as: $d_u = d - d_{sem} \mod \phi(n)$. The protocol of key generation algorithm is as follows. Let k be a security parameter. After CA computes, d_{sem} is securely communicated to the SEM and SK is communicated to the user.

(Algorithm: key generation)

(1) Generate random $k/2$-bit primes: p, q
(2) $n \leftarrow pq$
(3) $e \xleftarrow{r} Z^*_{\phi(n)}$
(4) $d \leftarrow 1/e \mod \phi(n)$
(5) $d_{sem} \xleftarrow{r} 1, ..., n-1$
(6) $d_u \leftarrow (d - d_{sem}) \mod \phi(n)$
(7) $SK \leftarrow (n, d_u)$
(8) $PK \leftarrow (n, e)$

2.4 mRSA signatures

According to PKCS #1v2.1 [11], RSA signature generation is composed of two steps: message encoding and cryptographic primitive computation. We denote by $EC()$ the encoding function. This encoding includes hashing the input message m using a collision resistant hash function. $EC()$ is the EMSA-PKCS1-v1_5 encoding function, recommended in [11].

1. The user sends the message m to the SEM.
2. The SEM checks the user's certificate. Only if this certificate is valid, the SEM computes a partial signature $PS_{sem} = EC(m)^{d_{sem}} \mod n$, and replies with it to the user.
3. The user computes $PS_u = EC(m)^{d_u} \mod n$. Then, the user receives PS_{sem} and computes $S(m) = (PS_{sem} * PS_u) \mod n$. It then verifies $S(m)$ as a standard RSA signature. If the signature is valid, the user outputs it.

3 Disadvantages of a SEM approach

One of the drawbacks of SEM approach is that it is vulnerable to Denial-of-Service (DoS) attacks, as pointed in [3]. There are three types of DoS attack:

against SEM's bandwidth, memory, and CPU. The purpose of the first attack is that a SEM cannot receive any more messages. The second one is performed to make a SEM store large quantities of waste states. The last one is the attack which makes a SEM computes a lot of quite inefficient processings. In SEM approach, we focus on DoS attacks against SEM's CPU.

The SEM server must generate the partial signature for every user's request. If an attacker can send a lot of requests, the SEM must compute a lot of partial signatures. Computations of a partial signature require a modular exponentiation. Consequently, it becomes highly vulnerable to DoS attacks.

To prevent DoS attacks, the SEM authenticates incoming requests. Only if a legal user sends a request, the SEM computes a partial signature. The SEM does not respond any request, sent by a party whom the responder cannot specify. Additionally, the SEM should confirm that the request is fresh. If an attacker can eavesdrop on communications between a legal user and a SEM, she can reuse a request created by legal users for replay attacks. That is, an attacker can send a lot of legitimate requests to the SEM. To prevent DoS attacks based on replay attacks, the SEM only responds to new requests.

In [3], several solutions are proposed. However, these solutions cannot prevent DoS attacks efficiently. Most cost-effective approach is to use Message Authentication Code (MAC) or keyed hash [3]. This situation requires a shared secret key k between a SEM and each user. A SEM can authenticate the request by checking MAC. However, it is vulnerable to replay attacks. An attacker can reuse requests generated by legal users, and send a lot of requests to a SEM. Suppose that the legal user sends requests $\langle MAC_k(m_1), m_1\rangle, \langle MAC_k(m_2), m_2\rangle, ..., \langle m_n, MAC_k(m_n)\rangle$, as shown in Fig. 2. $MAC_k(m_i)$ denotes the MAC using shared key k as the input message m_i. An attacker can eavesdrop these requests and send them to the SEM. This attack is called as replay attack. The SEM misunderstands that these request sent by an attacker are legal requests, and computes partial signatures, since the SEM cannot detect replay attacks.

4 Our proposed method

In MAC approach proposed in [3], the SEM cannot detect replay attacks. Suppose that an attacker, who doesn't know secret value, can eavesdrop and modify the data between the SEM server and the legal user. Under this environment, the goal of this paper is to prevent DoS attacks efficiently, and to protect the privacy of signers. This paper proposes Security-Mediated PKI using one-time ID.

4.1 One-time ID

To prevent leakage of user's identity and DoS attacks, Krawczyk proposed "One-time ID" [9]. One-time ID is a user's extraordinary identity, which has two properties as follows: (1) an attacker cannot specify who is communicating even when she eavesdrops on one-time ID, and (2) One-time ID can be used only

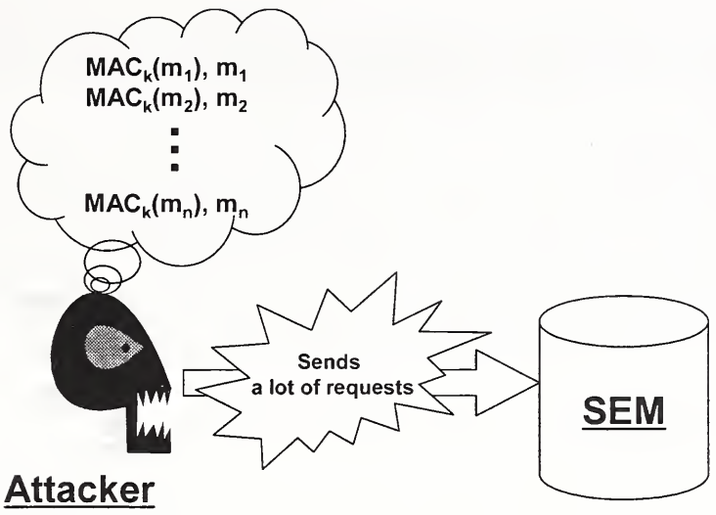


Fig. 2. DoS attack based on replay attack

once. To realize perfect forward secrecy, Imamoto et al. proposed new method of One-time ID calculation [7].

Taking into account the computational cost of users and a SEM, we utilize one-time ID protocol with small computational complexity, like a P-SIGMA. When one-time ID is generated, this method does not require modular exponentiations. The user and a SEM can compute one-time ID using one-way hash function. The SEM can authenticate incoming request by checking one-time ID. Additionally, the SEM can confirm that sending request is fresh because one-time ID is used only once.

Proposed system requires a shared secret key between a SEM and each user. The key-pair of a user is generated by CA, as well as the traditional SEM approach. And the CA generates a shared secret key between a SEM and a user, and sends it to both a SEM and user securely.

We describe the protocol of our system as follows. A user sends a request containing a message and one-time ID. One-time ID can be derived by shared secret key and be used only once. By checking one-time ID, a SEM server can confirm that requesting user is legal user. Only if requesting user is legal, a SEM generates a partial signature.

Our method has the following benefits.

1. DoS-resilient
   A SEM can efficiently detect the attacker's request, since an attacker cannot generate one-time ID unless she obtains a shared secret key. Moreover, proposed system can prevent replay attacks. An attacker cannot reuse the request generated by legal user because one-time ID can be used only once.
2. Efficiency
   One-time ID can be generated by hash computations. Therefore, computa-

tion costs of a SEM and a user are more efficient, compared with traditional methods. Additionally, communications between a SEM and a user are the same as the existing Security-Mediated PKI.

3. Privacy
   In existing approaches, a user must send a request contained a public key certificate. Thus, an attacker can trace the user's identity (e.g. public key certificate) by tracking user's request. In our proposed system, an attacker cannot trace user's identity even if she eavesdrops on one-time ID, because one-time ID is used only once. That is, our proposed system can protect the privacy of signers, compared with existing approaches.

4.2 Preliminaries

(Notations)

- U: user.
- PK: a public key of U.
- SK: a partial private key of U.
- d_{sem}: a partial private key of a SEM.
- k: a shared secret key between U and a SEM.
- $MAC_k()$: Message Authentication Codes using shared key k, such as HMAC [10].
- $H()$: collision-resistant hash function.
- OID_i: U's one-time ID with i-th session.
- m: message.

(Assumptions)

1. There is a shared secret key between the user and the SEM.
2. The secure channels are established between the CA and users, the CA and the SEM. This is the same assumption as the existing SEM approach [2, 3].
3. Communication between the SEM and users does not have to be protected. An attacker can eavesdrop and modify the contents of request message and response message.

4.3 SEM using one-time ID

(Setup)

1. As shown in Section 2.3, the CA generates PK, SK, d_{sem}, respectively. Then the CA generates k and issues the public key certificate of U. $\langle k, d_{sem} \rangle$ are securely communicated to the SEM and $\langle k, SK \rangle$ is communicated to U.

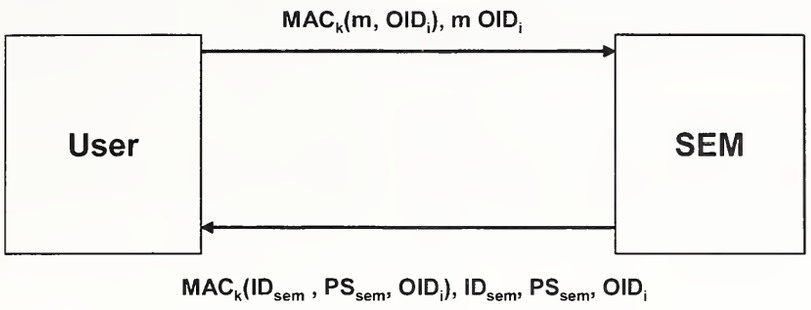


Fig. 3. Our proposed method

2. At first session, a SEM generates the list of OID_1 and U's certificate. OID_1 is derived as follow.

$$OID_1 = H(1, k)$$

(Signature)

1. U makes a request using one-time ID. First, U generates OID_1 and sends the following request.

$$\langle MAC_k(m, OID_1), m, OID_1 \rangle$$

2. First, the SEM confirms that the contents of a request is fresh by using k. That is, the SEM check the MAC value $MAC_k(m, OID_1)$. Then, the SEM checks who U is by using OID_1. If U is a legal user, the SEM validates U's certificate.
3. If U's certificate is valid, the SEM generates a partial signature $PS_{sem} = EC(m)^{d_{sem}} \mod n$ and sends the following response. Let ID_{sem} denote SEM's ID.

$$\langle MAC_k(ID_{sen}, PS_{SEM}, OID_1), ID_{sem}, PS_{sem}, OID_1 \rangle$$

4. U computes $PS_u = EC(m)^{d_u} \mod n$. Then U receives above response and confirms that contents of a response is fresh by checking $MAC_k(PS_{SEM}.OID_1)$. And U computes $S(m) = PS_{sem}^m * PS_u \mod n$. It then verifies $S(m)$ as a standard RSA signature. If the signature is valid, outputs it.

(Update of one-time ID)

1. SEM's update
   In session i, the SEM stores two one-time IDs $\langle OID_{i-1}, OID_i \rangle$ for U. If

one-time ID sent by U is correct, the SEM computes the partial signature and updates one-time ID as follows.

$$OID_{i+1} = H(k, OID_i)$$

And the SEM OID_{i-1} is deleted. If connection error is occurred, U does not receive the response. In that case, U does not update one-time ID. So, the SEM stores OID_i for authenticating U.

2. U's update
   If $S(m)$ is valid signature, U updates one-time ID as follows.

$$OID_{i+1} = H(k, OID_i)$$

   If some error is occurred, U sends the same request once again. The SEM sends the same response. It is notice that the SEM doesn't have to compute a partial signature again. The SEM only stores the same response.

5 Discussions

1. Signature forgery
   As mentioned in [2, 3], the user cannot generate signatures after being revoked. And the SEM cannot generate signatures on behalf of the user.
   However, an attacker can collude with the malicious user. If an attacker compromises a SEM and learns d_{sem}, he can create a signature by colluding with user. The existing SEM approach has the same problem.
2. DoS attacks
   After authenticating requests, a SEM validates a user's certificate and generates a partial signature. Suppose that an attacker can send a lot of requests to a SEM. An attacker cannot generate a legal request unless he can obtain a k. Our proposed system can prevent DoS attacks, since the SEM can detect illegal requests.
3. Replay attacks
   An attacker can reuse requests generated by legal users. Since one-time ID is changed for each session, an attacker cannot reuse a request for replay attacks.
4. Man-in-the-middle attack
   An attacker can modify the contents of request message. Suppose that U sends $\langle OID_i, m\rangle$. An attacker modifies $\langle OID_i, m'\rangle (m \neq m')$. To prevent this attack, U sends $\langle MAC_k(m, OID_i), m, OID_i\rangle$. Unless an attacker obtains k, an attacker cannot generate the legal request. Thus, our proposed system can prevent man-in-the-middle attacks.
5. Privacy
   In existing approaches, a user must send a request contained a public key certificate. Thus, an attacker can trace the user's identity (e.g. public key certificate) by tracking user's request. Even if the contents of the request are encrypted with k, an attacker can trace the user's identity. This fact will be

leading the privacy concerns. The privacy of signer can be protected by using SSL, but the burden of both users and the SEM may be heavy to establish the secure channel.
In our proposed system, the user doesn't have to send a request containing user's certificate, since the SEM can specify the user by checking one-time ID. Additionally, an attacker cannot trace user's identity even if she eavesdrops on one-time ID, because one-time ID is used only once. That is, our proposed system can protect the privacy of signers from any eavesdropper, compared with existing approaches.

5.1 Analysis of SEM's computational costs

In this section, we analyze the computational costs of the SEM. N_{DoS} denotes the number of illegal requests sent by an attacker. N_u denotes the number of legal requests sent by legal users. P and H mean the modular exponentiation and hash computation, respectively. In the traditional SEM approach [2], the computational cost of the SEM is $P(N_{DoS} + N_u)$. On the other hand, in our proposed system, the computational cost of the SEM is $H(N_{DoS} + N_u) + PN_u$.

We evaluate the number of hash computations. It is assumed that $P = 1000H$, since H is at least 1,000 times faster to compute than P. In the traditional SEM, the number of hash computations is $1000N_{DoS} + 1000N_u$. In our proposed SEM, the number of hash computations is $N_{DoS} + 1001N_u$. In our proposed system, the computational costs of the SEM are more efficient than those of traditional SEM.

5.2 Other solutions

There are some solutions to avoid DoS attacks based on replay attacks. This paper describes these methods in detail. However, these approaches cannot protect the privacy of signer.

(Challenge and response)

First, U sends a request message req and identifier of the user ID_u to the SEM. Then the SEM responds a random number referred to as a $Nonce$. After receiving a $Nonce$, U generates a MAC using shared key k and the $Nonce$ sent by the SEM.

1. $U \rightarrow SEM$: req, ID_u
2. $SEM \rightarrow U$: $Nonce$
3. $U \rightarrow SEM$: $MAC_k(m, Nonce), m, Nonce$
4. $SEM \rightarrow U$: $MAC_k(PS_{sem}, Nonce), PS_{sem}, Nonce$

- Security
  The SEM can confirm that the request is fresh by checking $Nonce$. The SEM must store the $Nonce$. It is possible to DoS attack against SEM's memory.
- Efficiency
  The number of rounds is four. It is inefficient of communications, compared with those of traditional method.

(Timestamp)

U generates a MAC using timestamp. TS_1 denotes the time of generating a request. The SEM check that TS_1 is fresh. Instead of timestamp, the user can send a request containing sequence number.

1. $U \rightarrow SEM: \quad MAC_k(m, TS_1), m, TS_1, ID_u$
2. $SEM \rightarrow U: \quad MAC_k(PS_{sem}, TS_2), PS_{sem}, TS_2$

- Security
  The SEM sets a time α. Only if $T \leq TS_1 + \alpha$, where T is the time of receiving request, the SEM can accept this request.
- Efficiency
  The computational and communicational costs are the same as those of traditional method.

6 Conclusions

The traditional SEM approach has the disadvantage that it is vulnerable to DoS attacks, since an attacker can send a lot of requests to the SEM server. To prevent DoS attacks, some solutions are proposed. However, these approaches cannot prevent DoS attacks. This paper proposes Security-Mediated PKI using one-time ID. Our proposed system can avoid DoS attacks with small computational complexity. Additionally, our proposed system can protect the privacy of signers, since one-time ID can be used only once.

Suppose that the SEM's partial private key d_{sem} is compromised. If a revoked user colludes with an attacker who obtains d_{sem}, a revoked user can generate a signature. This is the security issue in traditional SEM. Our future work is to improve this issue.

References

1. K. Bicakci, and N. Baykal, "A New Design of Privilege Management Infrastructure with Binding Signature Semantics," 1st European PKI Workshop (EuroPKI 2004), LNCS 3093, pp. 306–313, Springer-Verlag, 2004.
2. D. Boneh, X. Ding, G. Tsudik, and C. M. Wong, "A Method for Fast Revocation of Public Key Certificates and Security Capabilities," In proceedings of the 10th USENIX Security Symposium, pp. 297–308, 2001.
3. D. Boneh, X. Ding, and G. Tsudik, "Fine-grained Control of Security Capabilities," ACM Transactions on Internet Technology (TOIT), Volume 4, pp.60–82 , 2004

4. C. Boyd, "Digital multisignatures," Cryptography and Coding, Oxford University Press, pp. 241–246, 1989.
5. E. Bresson, O. Chevassut, and D. Pointcheval, "New Security Results on Encrypted Key Exchange," International Workshop on Practice and Theory in Public Key Cryptography (PKC 2004), LNCS 2947, pp.145-158, 2004.
6. R. Housley, W. Polk, W. Ford, and D. Solo, "Certificate and Certificate Revocation List (CRL) Profile," IETF RFC3280, 2002.
http://www.ietf.org/rfc/rfc3280.txt
7. K. Imamoto, and K. Sakurai, "A Design of Diffie-Hellman Based Key Exchange Using One-time ID in Pre-shared Key Model," The 18th International Conference on Advanced Information Networking and Applications (AINA 2004), pp. 327–332, 2004.
8. ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection - The Directory: Authentication Frameworks, 1997.
9. H. Krawczyk, "The IKE-SIGMA Protocol," Internet Draft, 2001.
http://www.ee.technion.ac.il/ hugo/draft-krawczyk-ipsec-ike-sigma-00.txt
10. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," IETF RFC2104, 1997. http://www.faqs.org/rfcs/rfc2104.html
11. R. Labs, "PKCS #1v2.1: RSA cryptography standard. Tech. rep.," RSA Laboratories, 2002.
12. K. Matsuura, and H. Imai, "Modified Aggressive Mode of Internet Key Exchange Resistant against Denial-of-Service Attacks," IEICE TRANS. INF.&SYST., Vol.E83-D, No.5, pp.972-979, 2000.
13. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSP," IETF RFC2560, 1999. http://www.ietf.org/rfc/rfc2560.txt
14. G. Vanrenen, and S. Smith, "Distributing Security-Mediated PKI," 1st European PKI Workshop (EuroPKI 2004), LNCS 3093, pp. 218–231, Springer-Verlag, 2004.

On Securing the Public Health Information Network Messaging System

Barry Rhodes
Associate Director For Public Health System Development
Information Resources Management Office
Centers for Disease Control and Prevention
www.cdc.gov

Rajashekar Kailar
Chief Technology Officer
Business Networks International, Inc.
www.bnetal.com

Abstract

The Public Health Information Network Messaging System (henceforth, PHINMS) is the Centers for Disease Control and Prevention's (CDC) implementation of the ebXML 2.0 messaging standards [EbXML]. This system was developed for the purpose of secure and reliable messaging over the Internet. This software has been widely deployed by CDC and its public health partners, including state and local health departments, and healthcare providers. PHINMS is designed to leverage X.509 digital certificates issued by public key infrastructures, but does not require a single, universal PKI. In this paper we discuss some of the security aspects of PHINMS.

Introduction

The Public Health Information Network Messaging System (PHINMS) is a CDC developed implementation of existing standards for the secure and reliable transmittal of messages across the Internet.

The PHINMS relies on ebXML, XML encryption [XMLENC], XML Digital Signature [XMLDSIG], SOAP [SOAP] and other standards. PHINMS is the primary message transport system for the National Electronic Disease Surveillance System [NEDSS], the Laboratory Response Network [LRN], National Health Safety Network [NHSN] and various other public health preparedness programs within CDC.

By design, PHINMS is message data (payload) independent; hence it can be used to transport any type of data (e.g., text, binary).

PHINMS is operating system neutral since it is implemented using Java and J2EE standards.

Further, it provides language neutral, queue based interfaces for sending and receiving messages. The preferred queue implementation is an ODBC/JDBC compliant database table, but support for queues based on XML file descriptors also exists. PHINMS supports peer-to-peer messaging, as well as messaging via a third party using a send and poll model.

Message data security is accomplished using a combination of encryption, end-point authentication, and access control techniques. Transport reliability is accomplished using message integrity verification, transmission retries and duplicate detection on message receipt.

Since PHINMS is used to transport sensitive data over public un-trusted networks (e.g., Internet), it is important to make sure that end-points trust each other, are able to identify and authenticate each other, and that communication channels preserve data confidentiality and integrity. Further, access to data sent and received should be controlled.

The balance of this paper will focus on some of the security considerations that went into the design and implementation of PHINMS.

Security Considerations

Several security considerations went into the design, implementation and deployment of PHINMS. The following is a brief description:

Trust[1]

Secure messaging over publie un-trusted networks requires messaging parties to be able to identify, authenticate and trust each other. For this, firstly, real world trust relationships need to be established between messaging organizations. This may include establishing written agreements on service levels, liabilities, etc., pursuant to OMB guidance on the Government Paperwork Elimination Act (GPEA) as well as the Electronic Signatures in Global and National Commerce Act (E-SIGN). Further, business processes for creating and handling messages at each end of the messaging pipe need to be put in place. Once trust and business relationships are established in real world terms, electronic collaboration agreements can be setup for message transport and processing. This includes setting up relationships to trust certification authorities and the identity of the sending and receiving components (e.g., using access control lists).

In a centralized trust model, a central node performs identity binding and security credentialing, and all nodes establish trust relationships with a central node. In this case, assuming n nodes, only $O(n)$ trust relations are needed. However, in a heterogeneous environment where trust is de-centralized, with n nodes, each node may need to establish a trust relationship and security credentials with every other node, and in the worst case scenario $O(n^2)$ trust relationships may be needed. Since messaging nodes typically belong to autonomous organizations and realms, establishing a globally accepted central identity and trust authority may not be politically acceptable. In a purely PKI based authentication framework, a trust bridge such as the Federal Bridge CA could be used to address this problem. However, while PHINMS supports PKI based authentication, it also supports other modes of authentication, such as basic or custom authentication.

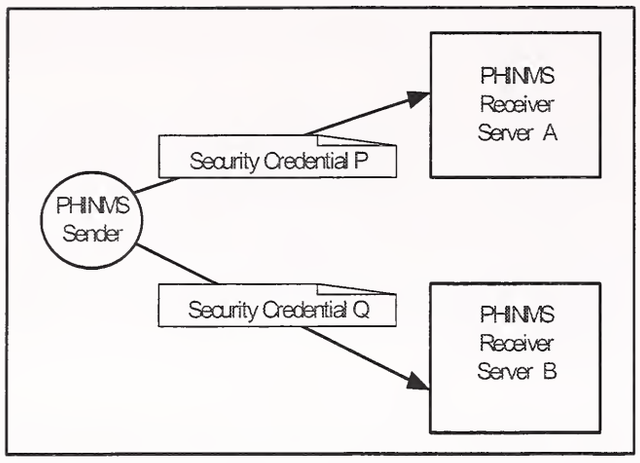
PHINMS is designed to support both centralized and de-centralized trust models. Decisions on identity binding and security credentialing are made by the deploying organizations. Decisions on trusting the identity and security credentials are made mutually between messaging parties at the time when electronic collaborations are created.

Identification, Authentication and Authorization

Identification and authentication in messaging is a difficult topic and is one that is far from mature. Since the message is typically sent by a process and not necessarily triggered by an individual, the authentication dialog must be scriptable. That is, the sending application must be able to negotiate the exchange of credentials without human intervention. This is only possible for certain security tokens (e.g., hardware based one time passwords and biometric identities don't lend themselves to this kind of scripted authentication exchange).

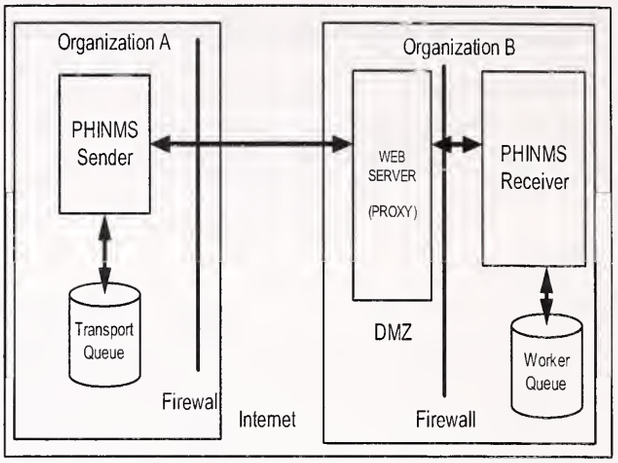
PHINMS supports automated authentication dialogs for client-certificate based authentication over SSL, basic authentication, and form based authentication. The method used for mutual and automated identification and authentication between messaging parties is part of the electronic agreement between them, and should be established upfront, after the real world trust relationship has been established.

[1] "Trust" in this context is more generic than what is involved in PKI based certificate chain validation. In particular, it may involve other (non-PKI) authentication mechanisms (e.g., basic or form based authentication).



Each messaging node in the Public Health Information Network (PHIN) is identified by a globally unique identifier. As shown in the diagram above, a messaging node (i.e., PHINMS sender) may contain one or more security credentials that allow it to conduct automated authentication dialogs with other messaging nodes. In the absence of a universally trusted authority to issue security identities and credentials, potentially, a different security identity and set of credentials may be needed for the purpose of authenticating to each message destination. The security credentials may include client certificates (key-stores), passwords, etc. Managing these security credentials can be a daunting task for the messaging administrator in the face of expiring certificates, password renewals etc. While certificates can be issued with an expiration period of years, passwords typically must be changed every 90 days, so the problem with the latter is far more daunting.

The recommended architecture for PHINMS messaging is one where the PHINMS receiver components are protected from direct access from the Internet, by web-server proxies as shown in the diagram at the top of the next column:



The web-server proxies typically reside in the organization's DMZ, and mediate all inbound traffic for the PHINMS receiver server, authenticating the sending process. SSL with client-certificate based authentication is the preferred method of authentication for PHINMS, since it is a well established standard and is widely implemented by web-server proxies.

Once the message sender is authenticated, it is the responsibility of the receiving organization's web-server proxy to ensure that an authenticated sender only gains access to the receiver URL. At this time, PHINMS does not provide support for attribute certificates which can be used for authorization decisions. Authorization information is stored on the receiver server, and enforced by the web-server proxy based on the authenticated identity of the PHINMS senders.

Authentication Factors

For interactive authentication dialogs over the Internet, generally, two factor authentication[2] is considered stronger and more secure than single factor authentication.

[2] Authentication mechanisms typically use secrets such as what a user knows (e.g., password), what a user has (e.g., hardware token) and what a user is (e.g., thumbprint). These are called authentication factors. For strong authentication, a combination of two of these three factors is used.

However, in the case of B2B automated security dialog, the security value of two-factor authentication is significantly diminished, since there is no real user behind the authentication dialog. All user factors required for the authentication dialogs would need to be pre-configured into the software that initiates the authentication handshake. Further, at the time of this writing, there are no published and accepted Internet standards for two factor authentication in B2B transactions. While it is possible to use hardware based security modules (sometimes called HSM) to emulate additional authentication factors for B2B exchanges, such mechanisms require additional hardware and management complexity.

Confidentiality

Since communication is over un-trusted public networks, protecting its confidentiality is important. PHINMS uses payload level asymmetric encryption for end-to-end persistent confidentiality. The XML encryption standard The XML encryption standard [XMLENC] is used for encrypting the payload.

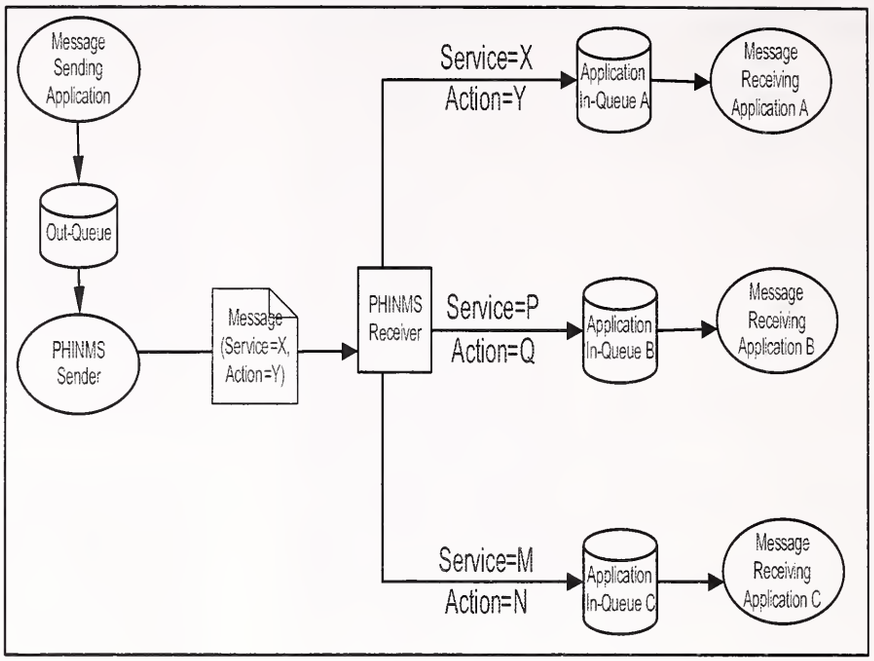
In the case of store and forward messaging, data is protected from being read by intermediaries by using asymmetric encryption using the public key of the message recipient to encrypt a random symmetric key, which in turn encrypts the data. Additionally, communication is typically conducted over a Secure Sockets Layer (SSL) channel, ensuring that the message meta-data is also protected. To ensure end-to-end confidentiality, the channel between the web-server proxy and the application server is also over SSL.

Integrity and Non-repudiation

PHINMS supports the use of XML digital signatures [XMLDSIG] for message integrity and non-repudiation of message data. Signing certificates can be sent as part of the signature meta-data facilitating verification of the signature, alternatively, signing certificates can be statically pre-configured at the receiving node. Additionally, communication is typically conducted over SSL with client-certificate based authentication, which provides further message integrity and non-repudiation assurances.

Access Control

The ebXML messaging standard supports message labels called "Service" and "Action". These XML tags are part of the message envelope, and can be mapped to a service on the receiving node.



In the PHINMS implementation, messages that are received using the receiver server are stored in database tables (queues) based on their Service and Action tags. These queues are the equivalent of an application "inbox", and each application can only access its own inbox.

Public Key Infrastructure (PKI)

PHINMS is designed to leverage a PKI, but it does not require a universal PKI. For instance, a PHINMS sending client can use a client certificate issued by one certification authority (CA) to authenticate itself to a PHINMS receiver server, and use a client certificate issued by a different CA to authenticate itself to a different PHINMS receiver server. Currently, PKI trust relations are statically defined at the time when collaboration is established and configured between messaging entities. This is sometimes called the "Certificate Trust List" model.

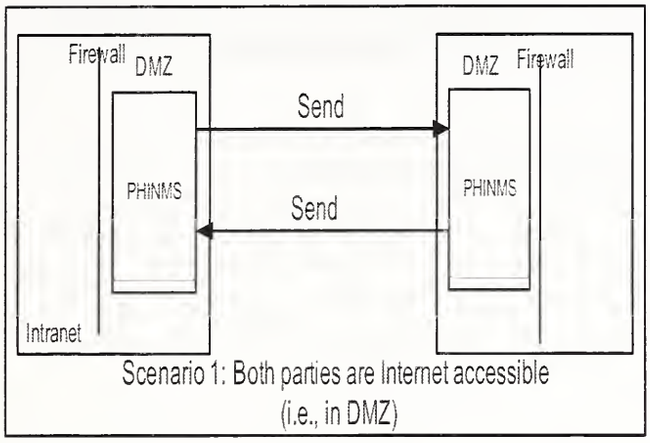
Ideally, public key certificates are published in an LDAP directory (need not be centralized), but PHINMS also supports a web-service interface to publish and retrieve certificates. As a third alternative, encryption public key certificates can be distributed out of band and pre-configured at the message sending nodes. Public key certificates can be published in de-centralized LDAP directories as well.

Firewalls

Though firewalls are necessary for the protection of resources within an enterprise, they complicate matters for a messaging system trying to send messages across enterprise boundaries. PHINMS uses two independent pieces of code, a client capable of sending messages and receiving real time (synchronous) responses, and a server receiver that can receive messages at any time. These two components may be used in three possible scenarios. These examples assume that the parties are in different organizations with separate firewalls.

1. Both parties are located outside their respective firewalls (i.e., in their DMZ)
2. One party is outside the firewall and the other is inside a firewall.
3. Both parties are inside their respective firewalls.

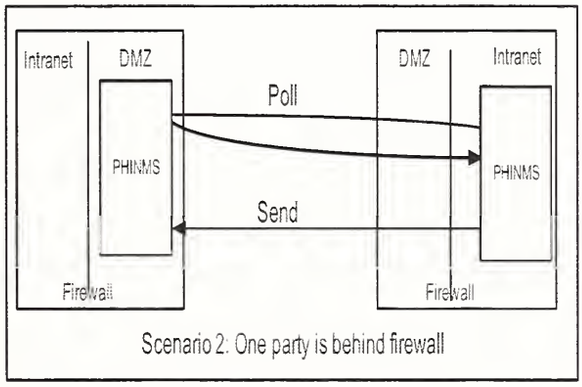
In the case where both parties are located outside their respective firewalls, messages may be sent and received at any time and acknowledgements send either synchronously or asynchronously. This requires that both parties have sending and receiving components installed.



Scenario 1: Both parties are Internet accessible (i.e., in DMZ)

For the situation where one party is behind a firewall and the other party has a server receiver located in the public Internet space, message sending options are slightly reduced. The client piece behind the firewall can send data much like a typical browser to a receiver and receive synchronous acknowledgements back.

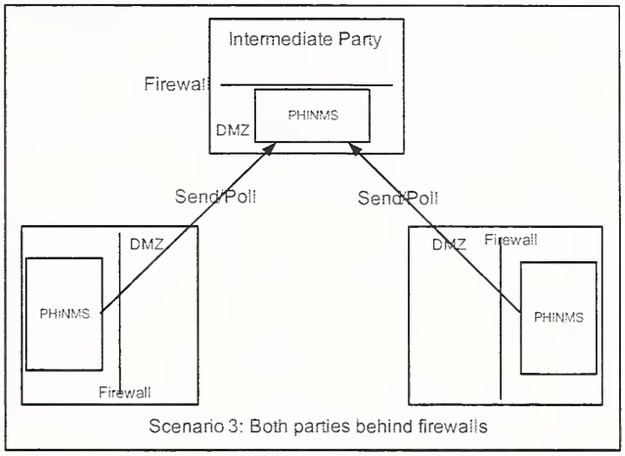
Because it sits behind a firewall, the client cannot receive messages as firewalls typically block this type of "push" of information. What it can do is poll for messages.



Scenario 2: One party is behind firewall

Basically a poll is where a client sends a message to a server with some meta-data which maps to a piece of functionality that looks to see if the server has something for the client. If so, the server can return the file as a response to the send. Because the client is not really receiving messages, the complexity of the software is reduced and therefore the platform requirements are reduced as well.

Typically the client can reside on a workstation capable of hosting a Java application.

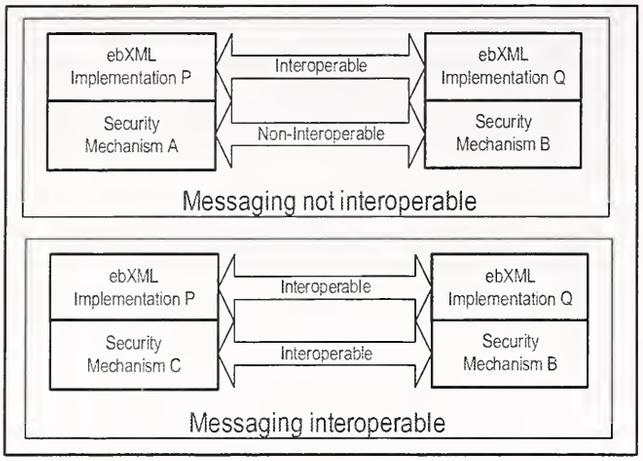


Scenario 3: Both parties behind firewalls

In the case where both parties are behind firewalls, a third party server with Internet presence is required to broker the exchange. For example let's say party A is located behind a firewall in enterprise 1 and A wants to send a message to party B in enterprise 2, where B is also behind a firewall. Then A must send a message to an intermediary server on the Internet with a service action that states that the server should hold the message in a queue for B. Then when B polls the server, it will find the message from A in its queue and request it.

Authentication Interoperability

The ebXML messaging standard specifies the structure and semantics of message meta-data and addressing information, but for the most part, leaves the messaging security (identification and authentication) aspects to the implementers.



As shown in the above diagram, for interoperability, in addition to the message structure and semantics, the security mechanisms also need to interoperate. XML digital signatures can be used to support message non-repudiation (the strength of which is dependent upon legal elements that transcend the technology), but using them may not be sufficient for the purpose of authenticating clients to sensitive applications unless its freshness is established. Without adequate freshness assurances use of DSIG in authentication may not be adequate for some applications.

When used, XML digital signatures should be combined with a handshaking protocol such as SSL, which mitigates the threat of replay attacks and provides freshness assurances. The alternative is to use SSL with client certificate based authentication. This provides per-link assurance of identity and authentication, as well as confidentiality. Since SSL is the most widely accepted standard, this is the recommended mode of authentication for PHINMS.

Acknowledgements

The authors would like to thank the Public Health Information Network Messaging System team: Michele Bowman, Thomas Brinks, Dongtao Jiang, Vaughn McMullin, Thomas Russell, and John Thomas.

Summary

The security design, implementation and deployment considerations of CDC's Public Health Information Network Messaging System (PHINMS) were discussed herein.

References

[EbXML] Message Service Specification

Version 2.0, OASIS ebXML Messaging Services Technical Committee (http://www.ebxml.org/specs/ebMS2.pdf)

[LRN] The Laboratory Response Network Partners in Preparedness http://www.bt.cdc.gov/lrn/

[NEDSS] National Electronic Disease Surveillance System, The Surveillance and Monitoring Component for the Public Health Information Network. (www.cdc.gov/nedss/)

[NHSN] National Healthcare Safety Network (NHSN)(http://www.cdc.gov/ncidod/hip/NNIS/members/nhsn.htm)

[SOAP] SOAP Version 1.2 Part 0: Primer

(http://www.w3.org/TR/2003/REC-soap12-part0-20030624/)

[XMLENC] XML Encryption Requirements

(www.w3.org/TR/xml-encryption-req)

[XMLDSIG] XML-Signature Syntax Processing (www.w3g.org/TR/xmldsig-code

List of Acronyms

AA	Attribute Authority
AC	Attribute Certificate
ACL	Access Control List
ACRL	Attribute Certificate Revocation List
AIA	Authority Information Access
AKI	Authority Key Identifier Extension
API	Application Programming Interface
AS	Autonomous Systems
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BCA	Bridge Certification Authorities
BER	Basic Encoding Rules for ASN.1
CA	Certification Authority
BGP	Border Gateway Protocol
CAS	Community Authorization Service
CAC	Common Access Cards
C & A	Certification and Accreditation
CBC	Cipher Block Chaining
CBC-MAC	Cipher Block Chaining Message Authentication Code
CBEFF	Common Biometric Exchange Format Framework
CDC	Centers for Disease Control
CHUID	Card Holder Unique Identifier
CIA	Community Informative Authority
CMP	Certificate Management Protocol
CP	Certificate Ppolicy
CPFCA	Common Policy Framework Certificate Authority
CRC	Certificate Result Certificates
CPS	Certification Practices Statement
CRL	Certificate Revocation List
CRT	Certificate Revocation Tree
CUID	Card Unique Identifier
CV	Control Value
CVCA	Commercial Vendor's Certification Authority
DER	Distinguished Encoding Rules for ASN.1

DIS	Delegation Issuing Service
DIT	Directory Information Tree
DN	Distinguished Name
DoS	Denial of Service
DSA	Digital Signature Algorithm
DSL	Digital Subscriber Line
EKU	Extended Key Usage
E-SIGN	Electronic Signatures in Global and National Commerce Act
FBCA	Federal PKI Bridge Certificate Authority
FICC	Federal Identity Credentialing Committee
FIPS	Federal Information Processing Standard
FPKI-PA	Federal PKI Policy Authority
FTPL	First order Temporal Policy analysis Logic
GDS	Global Directory Services
GOL	Government On-Line
GPEA	Government Paperwork Elimination Act
GSER	Generic String Encoding Rules
HIBE	Hierarchical Identity-Based Encryption
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
IANA	Internet Assigned Numbers Authority
IBE	Identity-Based Encryption
IE	Internet Explorer
IEEE	Institute for Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IRV	Inter-domain Routing Validation
KED	Key Escrow Database
LDAP	Lightweight Directory Access Protocol
LRA	Local Registration Authorities
MAC	Message Authentication Code
MIME	Multipurpose Internet Mail Extensions
MOA	Memorandum of Agreement
MRAI	Minimum Route Advertisement Interval
MTA	Mail Transfer Agent

NIST	National Institute of Standards and Technology
NMI	NSF Middleware Initiative
OASIS	Organization for the Advancement of Structured Information Standards
OAT	Origin Authentication Tags
OCSP	Online Certificate Status Protocol
OGSA	Open Grid Services Architecture
OGSI	Open Grid Services Infrastructure
OMB	Office of Management and Budget
PAC	Privilege Attribute Certificate
PDA	Personal Digital Assistant
PDP	Policy Decision Point
PGP	Pretty Good Privacy
PHIN	Public Health Information Network
PHINMS	Public Health Information Network Messaging System
PIP	Policy Information Point
PKC	Public Key Certificate
PKCS-12	Public-Key Cryptography Standard Number 12
PKG	Private Key Generator
PKI	Public Key Infrastructure
PMI	Privilege Management Infrastructure
POLA	Principle of Least Authority
POP3	Post Office Protocol Version 3
PV	Protection Value
PWGSC	Public Works and Government Services Canada
RA	Route Attestations
RAP	Roll Assignment Policy
RBAC	Roll Based Access Control
RP	Relying Party
RSA	Rivest Shamir Adelman cryptographic algorithm
SAML	Security Assertion Markup Language
SAS	Sequential Aggregate Signature
S-BGP	Secure Border Gateway Protocol
SCVP	Simple Certificate Validation Protocol
SDSI	Simple Distributed Security Infrastructure
SEM	SEcurity Mediator

SHA-1	Secure Hash Algorithm, as specified in FIPS 186-1 (also denoted SHA1)
SKI	Subject Key Identifier extension
S/MIME	Secure/Multipurpose Internet Mail Extensions
SMTP	Simple Mail Transfer Protocol
SOA	Source of Authority
SOAP	Simple Object Access Protocol (XML protocol)
SPKI	Simple Public Key Infrastructure
SSH	Secure Shell
SSL	Secure Sockets Layer protocol
TAP	Target Access sub-Policy
TCP	Transmission Control Protocol
UPN	User Principal Name
URL	Universal Resource Locator
VO	Virtual Organization
VOMS	Virtual Organization Management Service
W3C	World Wide Web Consortium
WAYF	The “Where are you from?” problem
WDSL	Web Services Definition Language
X.509	The ISO/ITU X.509 standard
XACML	Extensible Access Control Markup Language
XER	XML Encoding Rules for ASN.1
XKMS	XML Key Management System
XML	Extensible Markup Language