NISTIR 7059
ISBN 1-886843-34-1

1st Annual PKI Research Workshop Proceedings

Sean W. Smith
William T. Polk
Nelson E. Hastings

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

NISTIR 7059
ISBN 1-886843-34-1

1ST Annual PKI Research Workshop Proceedings

Sean W. Smith
Department of Computer Science
Dartmouth College

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

October 2003

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary
TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arden L. Bement, Jr., 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

Organizers

General Chair: Ken Klingenstein, University of Colorado.

Program Chair: Sean Smith, Dartmouth College.

Stipends Chair: Neal McBurnett, Internet2.

WIP Chair: Peter Honeyman, University of Michigan.

Local Arrangements Chair: Nelson Hastings, NIST.

Scribe: Ben Chinowsky, Internet2.

Program Committee:

Peter Alterman, NIH.
Steve Bellovin, AT&T Labs Research.
Stefan Brands, McGill University.
Bill Burr, NIST.
Carl Ellison, Intel.
Stephen Farrell, Baltimore Technologies.
Richard Guida, Johnson and Johnson.
Peter Honeyman, University of Michigan.
Ken Klingenstein, University of Colorado.
Larry Landweber, University of Wisconsin.
Neal McBurnett, Internet2.
Clifford Neuman, USC.
Sean Smith, Dartmouth College.
Steve Tuecke, Argonne National Laboratory.

Additional thanks to: Renee Frost, Tim Polk, Jim Rees, Ellen Vaughan, and Jiaying Zhang.

Date

April 24-25, 2002. Gaithersburg, Maryland USA.

Archival Web Site

`http://www.cs.dartmouth.edu/~pki02/`

Contents

Invited Talks and Experience Reports

Position Papers

Preface

Since its discovery three decades ago, public-key cryptography has excited computer scientists and practitioners alike, because of its potential to enable trusted information services between parties who do not share secrets *a priori*. Public-key infrastructure—the technology to make this cryptography work in practice—would appear to be critical in the emerging information infrastructure, which replete with boundaries—organizational, temporal, and many others—that separate parties and make sharing difficult or impossible.

However, public-key cryptography has not fully achieved this vision. Some would assert that "PKI has not happened yet"; others believe it is happening, but more slowly than anticipated. Researchers also exist with more extreme viewpoints.

We convened this workshop to address the gap between this vision and the current state of PKI, and attracted a critical mass of participants from government, industry, and academia, and representing a full spectrum of approaches and opinions.

This volume contains a written record of the result: the formal refereed papers and experience reports of the conference, as well as summaries of the panels and discussions, and position papers submitted by some attendees. On behalf of the entire organizing team, I thank all the participants. We hope that this is the first in a series of conferences that helps our community achieve the long-term vision of PKI.

Sean Smith, Program Chair
Dartmouth College
Hanover, New Hampshire USA

sws@cs.dartmouth.edu

Foreward

The U.S. Federal Government and academia have a long history of collaborative research and development in the area of Public Key Infrastructure (PKI). This collaboration arose from a shared vision: secure electronic communications based on widely available PKIs. Both government and academia are actively deploying PKIs to link the different members of their own communities. To leverage that investment, each envisions the formation of PKIs that encompass government, academia, and the private sector.

These shared interests have motivated a number of collaborative projects. For example, Georgia Institute of Technology participated in the first demonstration of bridge CA concept at the EMA challenge in 1999. University of Georgetown and University of Washington, among others, have actively participated in the Federal PKI Technical Working Group since 2000. More recently, the National Institutes of Health and EduCause have worked collaboratively on a pilot program for grant application filing.

These collaborative projects are just one piece of the mosaic of PKI research. Research is also underway in a number of independent academic and government projects, as well as numerous private sector efforts. In total, the breadth and quality of PKI-related research is impressive. This research has previously been presented at a variety of general security conferences. A PKI-specific venue is needed to increase opportunities for PKI researchers to share their progress and experiences.

Late in 2001, Internet2 proposed a research and development workshop devoted entirely to the area of PKI. The proposal was enthusiastically received and a core group from government, academia, and private industry emerged to organize and support the workshop. The National Institutes of Health provided funding to support the workshop; NIST volunteered to host the workshop; and Internet2 recruited PKI experts from academia, government, and the private sector for the program committee.

NIST hosted the first annual Public Key Infrastructure (PKI) Research Workshop on April 24-25, 2002. The two-day event brought together PKI experts from academia, industry, and government to explore the remaining challenges in deploying public key authentication and authorization technologies, and to develop a research agenda to address those outstanding issues.

The workshop consisted of the presentation of 14 referred papers, four panel discussions, and a work-in-progress session. About 100 participants from the United States, United Kingdom, Canada, Spain, Sweden, Ireland, Taiwan, and South Korea made the workshop an international event. Based on participant feedback, the workshop provided the most up-to-date information on PKI research and deployment. This proceedings includes the refereed papers, and captures the essence of the panels and interaction at the workshop.

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

Summaries

Workshop Summary

Ben Chinowsky
Internet2

This report summarizes current issues in PKI as discussed at the 1st Annual PKI Research Workshop, held April 24-25, 2002 and sponsored by NIST, NIH and Internet2.

Sense of the Meeting

While reaching consensus was not among the goals of the workshop, there appeared to be something close to general agreement on the following points.

PKI trust relationships must be built on real-world trust relationships. In the workshop's XKMS panel discussion, Phillip Hallam-Baker described PKI as "the interface between the Internet and the Real World," and it was evident throughout the workshop that PKI practitioners are increasingly taking this as a starting point. At the coarsest level of generalization, hierarchical (aka traditional) PKIs are usually more appropriate for hierarchical organizations—such as the military, as discussed by Green, Winnenberg, Henry, and Fink. Non-hierarchical PKIs (aka trust networks, webs of trust, or anarchy) are usually more appropriate for non-hierarchical organizations—such as the collaborative groups discussed by Dohrmann and Ellison. Hallam-Baker illustrated the idea of building PKIs on existing trust relationships in his overview of the Trust Assertion XML Infrastructure (TAXI), the direct ancestor of SAML and XKMS. Even in the "Dueling Theologies" session that opened the workshop, there was little of the my-model-is-better-than-your-model style of argument common to many discussions of PKI. Instead, there is a growing awareness that we have a wide variety of tools and a wide variety of circumstances in which they can be applied, and growing agreement that starting from existing real-world trust relationships, whether those relationships be hierarchies or networks, is the central principle that should guide how we apply these tools.

At the same time, there is also broad agreement that the closer you look at these top-level categorizations—hierarchical vs. non-hierarchical, real-world vs. not—the more questions arise. Does "traditional PKI" refer only to X.509 with a strict X.500-style naming hierarchy, or is it broader than that? When members of a purchasing department operating under instructions to honor any purchase order coming from some specified class of individuals nonetheless insist on making some kind of personal contact before placing an order for someone they're not familiar with, what are the real-world trust relationships that PKI should follow? Clearly the top-level categories, while necessary, are not sufficient for describing either real-world or PKI trust relationships. It was also noted that in some cases—such as the use of PKI to ensure privacy or anonymity—it can be important to make sure that PKI trust relationships *don't* follow real-world trust relationships.

Because the real-world trust relationships of many large organizations are "heterarchical"—consisting of a diverse set of hierarchies, anarchies, and combinations of the two—heterarchical PKIs appear to have a bright future. Such hybrid PKIs are created by means of bridge CAs. Federal PKI Steering Committee Chair Spencer briefly discussed progress on the Federal Bridge Certification Authority (FBCA). Alterman presented a progress report on the NIH-EDUCAUSE PKI Interoperability Project, which centers on communication between the FBCA and the Higher Education Bridge Certification Authority (HEBCA); Alterman summed up by saying that "there are NO show-stoppers." The workshop's work-in-progress session included a discussion by Alterman of possible topologies for a multiple-bridge infrastructure.

Directory functionality is a central concern for both traditionally- and non-traditionally-minded PKI practitioners. For example, Marc Branchaud of RSA noted that "the directory is the main thing that makes X.509 work," and Peter Alterman observed that "solving directory issues is the key to interoperability." On the other hand, in his critique of "conventional PKI wisdom," Carl Ellison puts the problem of naming entities front and center. Ellison sees the gap between the ways computers use names (precisely) and the way humans use names (imprecisely) as a big obstacle to hu-

mans being able to trust that they have chosen the right cert from a directory and are dealing with the person they think they are dealing with. At Intel this has become known as "the John Wilson problem." Ellison advocates using personal directories or naming services that can use "local names" (e.g., "my mom") to retrieve keys.

Users want security, but they're not willing to tolerate much additional system complexity in order to get it. If security adds significant complexity, users will either use it incorrectly—which can provide a false sense of security, leaving the user worse off than before—or not use it at all. Carl Ellison argued that the main successful deployment of certificates so far, SSL, is in effect mostly used to grant this false sense of security. Ellison suggested an experiment comparing the frequency of stolen credit card numbers in encrypted and unencrypted transactions; he was was sufficiently confident in his pessimism about SSL to offer to include his own credit card in the non-encrypted sample.

Legal issues, in particular certificate policy issues, are very hard. Fink, discussing his work with the Naval Surface Warfare Center, observed that PKI can also stand for "Policy Keeps Interfering." Green laid heavy stress on the DoD's work in this area: "we have a major activity in the certificate policy world...if you're not paying attention to this you're not taking PKI seriously." Klingenstein described a trust continuum running from collaborative trust (handshakes) to legal trust (contracts). While collaborative trust tends to go with the federated models of security (like Shibboleth, which resembles a bridge CA in some respects), and legal trust tends to go with traditional PKI, there are a wide range of intermediate cases, and each user community needs to decide what mix works best for it.

Issues and Approaches

Key management and mobility. Much discussion was devoted to various schemes for ensuring that people can access their keys as needed, both at the time of issuance and thereafter. In the session on key management, Gupta provided a survey of current approaches. She emphasized the wide variety of solutions available and noted three contraindications for attempting to implement mobility: a need for strong non-repudiation, a need to be able to recover encryption keys, and zero tolerance for DoS attacks. Perrin presented a system for sharing a single private key among many users; he noted that his system is intended to interoperate with conventional PKI rather than replace it. Perrin's system uses an online trusted third party; Peter Honeyman pointed out that if you remove the asymmetric cryptography from this system, it looks a lot like Kerberos, and asked why he didn't just use that. Perrin replied that his system makes path validation possible and can be implemented without a central server for the shared private key (though the prototype does indeed use such a server). Also on the theme of incorporating secret-key cryptography, Sandhu pointed out that "it is completely possible to design a sufficiently secure password system...security is always about adequacy." Absolute security doesn't exist anyway, and users don't inherently hate passwords, they just don't want so many of them. With respect to the question of what's holding back physical smartcards; Sandhu observed that "it's the readers, stupid;" he described the principal motivation of his work on virtual smartcards as to provide a "phased migration path" from weak passwords to strong PKI.

Smartcards are a major focus of effort for the military, and the DoD and International Coalitions presentations included two striking cautionary tales drawn from their experience. One speaker noted that smartcard readers present more of a challenge than smartcards themselves, and recounted an episode in which users were issued smartcards and PINs, but then six months elapsed before the card readers were installed and working, so that the PINs were mostly forgotten. Henry noted that the DoD currently combines smartcards with Geneva Convention cards; as the Geneva Convention card is to be surrendered upon capture by the enemy, this clearly needs to be fixed.

Also closely related to key management were the presentations of Boneh and Levy on their respective developments of identity-based encryption (IBE). IBE uses information about the user, such as an email address, to create a public key, making it possible to send someone encrypted mail without them having to first set up a keypair and publish their public key. The recipient then visits a server to obtain the corresponding private key. Boneh emphasized the "viral" deployment properties of this system, seeing its potential to encourage broader use of PKI as its principal advantage. Levy emphasized the control that IBE gives the sender over what information the receiver needs to provide the server in order to get their private key; the sender thus gains precise control over how secure the encryption will be.

Authorization. Underscoring the importance of authorization for PKI as a whole, in addition to the main session on authorization, three of the four presentations in the workshop's "Scale" session were also devoted to authorization. DeTreville set out his thinking on how

to do scalable distributed authorization by building relational algebra into certificates. Canovas discussed his work on delegation of authorization in SPKI, which has been deployed in a production smartcard system at his university. Knight discussed the role-based X.509 privilege management infrastructure he is developing for the Canadian Department of National Defence.

In the authorization session proper, Dam discussed a streamlined version of the SPKI authorization syntax which is adequate for almost all real-world uses but which executes in linear rather than exponential time. Dohrmann outlined a PKI that he and Carl Ellison developed with the overarching goal of improving ease of use, thereby improving the likelihood that the system will be used correctly. One of the ways they do this is by having lines of authority to grant authorizations follow existing lines of authority within an organization; for example, long authorization chains that go up one side of the org chart and down the other are preferred to short ones that cut across from one leaf node to another. Thompson provided an overview of approaches to authorization and an in-depth look at Akenti. Akenti is a Grid-oriented authorization system which has been implemented as an Apache module and which has been used by the Diesel Combustion Collaboratory and the National Fusion Collaboratory.

The workshop's emphasis on ensuring ease of use was especially strong in the discussions of authorization, reflecting a general awareness of the conceptual complexity of relationships in this area.

Validation and revocation. In the validation session, Micali introduced NOVOMODO, a scheme for ultra-lightweight certificate validation via 20-byte tokens. Micali developed an extended analogy between these tokens and the validity stickers affixed to student ID cards at the start of each term. Tero Hasu, presenting work by his colleague Kortesniemi discussed a range of options for validity management of SPKI authorization certificates, and set out a very simple (only two messages) validity management protocol. Branchaud noted that while X.509 was built on the assumption that CAs aren't online, that assumption no longer necessarily holds. He provided an overview of resulting options for distributed and delegated validation, looking beyond OCSP to, "in the limit," possibly getting rid of certificates altogether.

Agenda

In order to work out both the social and the mechanical issues, **we need more deployment experience.** While the deployments discussed at the workshop have provided many useful lessons, the user base of these deployments is tiny in relation to the user base PKI will need to support. In addition to the hundreds of technical details that can only be fully resolved in the course of a full-scale deployment, there are a lot of "Why Johnny Can't Encrypt"-type questions that can't be answered until there is more experience with PKIs supporting thousands rather than dozens of users. In addition to removing obstacles to deployment, we must also ensure that there is sufficient positive motivation for PKI; as Phillip Hallam-Baker noted, "you don't want to deploy PKI starting with problems that have already been solved better."

We need to do a better job of working with social scientists, lawyers, and other "non-technical" experts. It seems clear that these experts are available and willing to help, but the initiative and direction in applying their skills have to come from the technical community.

We need to keep cross-pollinating. There was near-unanimous opinion in favor of immediately beginning planning for a 2nd Annual PKI Research Workshop, and that planning is now underway.

Dueling Theologies

Ben Chinowsky
Internet2

In this panel session, Rich Guida gave his view of what's holding back the traditional X.509 model that he favors, and Carl Ellison summarized his criticisms of this model.

Guida listed several factors holding back wider deployment of PKI, including: too many legacy applications and too few PKI-enabled applications; a widespread desire on the part of decision-makers to be on the leading rather than the bleeding edge; lack of common semantics; organizational politics, including the "not invented here" syndrome; and (least importantly) technical issues. Guida also pointed out that, as with network technologies more generally, it is very hard to calculate ROI for PKI, and suggested that those pushing PKI deployment not get "wrapped around the ROI axle." Guida sees PKI becoming widespread first within enterprises, then between them, and lastly with consumers. Guida also outlined the PKI he's currently working on for Johnson & Johnson.

Ellison sees fundamental problems with conventional PKI. In his view, there are four pieces of PKI "conventional wisdom" which need to be rejected.

- 1. Conventional wisdom: Everyone needs to have an identity cert for digital signatures.

  Objection: Each person has multiple identities (as a driver, as a bank account holder, as an employee...); therefore each person would need many identity certs.

- 2. Conventional wisdom: Certs should come from a CA with strong private-key security.

  Objection: It's too expensive to have more than a few such CAs, making it necessary for users to travel to the CA in order to get a cert. Using RAs can improve this situation; Ellison advocates going this solution one better by putting the CA on the RA's desk.

- 3. Conventional wisdom: Once you've done 1) and 2), you know who you're talking to...

  Objection: "Human beings do not use names the way we computer scientists would like them to." Ellison noted that when he tells stories of the confusion created by the multiple John Wilsons at Intel, people tend to respond along the lines of "that's nothing, listen to this." When using names, people have a strongly ingrained tendency to go with the first apparent match they see, leading to (in the stories Ellison related) misdirected email messages, airline boarding pass mixups, and (almost) unwanted botox injections.

- 4. Conventional wisdom: ...and you also have nonrepudiation.

  Objection: The costs of strong private-key security and the need for tamper-proof cameras to witness digital signing make nonrepudiation impractical. The usefulness of providing nonrepudiation is in any case limited to situations in which a victim can be made whole, thus excluding cases where, for example, secrecy or human life is at stake.

Ellison's solution is to dispense with identity certs and CAs, replacing them with authorization certs issued by whoever has the authority to grant the authorization under existing business practices.

Much of the Q&A was devoted to rebuttals to Ellison's objections to traditional PKI. Several people pointed out that traditional PKI need not lean so heavily on names as Ellison assumes it does: naming is often backed up by established business relationships and larger sets of information about the named entities. While Ellison agreed that the use of these backups can help, his response centered on stressing just how little rigor can be expected from users. He also cited an episode in which he used SSL to make an apparently secure transaction with a vendor, then checked the cert and found that it had been issued to another entity entirely. While presumably the vendor had contracted with this entity for web services, nowhere in the process was there any proof of this.

There was also a short discussion of nonrepudiation; Ellison argued that online credit card transactions are safe for the purchaser, and therefore widespread, precisely because they *can* be repudiated, and not because SSL protects the transaction from eavesdropping.

XKMS Panel

Ben Chinowsky
Internet2

Phillip Hallam-Baker, one of the architects of XKMS, opened the discussion by describing the central idea of XKMS as to remove complexity—especially the complexity of path discovery—from the client, so that it doesn't have to be concerned with anything more than "I want to talk to Alice." While agreeing that XKMS could prove useful in hiding complexity from the user, and that offloading path validation might be useful in enabling PKI on computationally weak devices such as cell phones, Tim Polk countered that most desktop machines can handle path validation just fine. Polk is also suspicious of "unified field theories" in general, and XKMS's aspiration to be the unified field theory for PKI in particular. He's also skeptical about the claim that XML is superior to ASN.1 as a format for PKI—while ASN.1 is complex, so is XML, and ASN.1 is "the devil we know," as well as being better at describing "bit-for-bit identity." Blair Dillaway, another coauthor of the XKMS technical note, cited Microsoft's interest in using XKMS to develop its digital rights management and delegation system, which is based on XRML, in a more open and flexible direction. MIT's Dan Greenwood argued that XKMS fails to address key business and legal issues. While "public key technologies are best tailored to support and reflect existing business and legal infrastructures—that is where trust is created," XKMS appears to be centrally concerned with "stranger to stranger" authentication. Greenwood cited LegalXML and his own `actuarinet.mit.edu` as exemplifying a better approach.

Eric Norman opened the questions by asking, "what is trust?" Hallam-Baker replied that trust is quantification of risk. Greenwood objected that, while it would be nice if a workable definition of trust could be so simple, the concepts of a trusted third party and nonrepudiation are also necessary. Tim Polk concurred that there is an irreducibly subjective and unquantifiable aspect to trust—we just have to live with it. Hallam-Baker noted that XKMS tools are now available in VeriSign's Trust Services Integration Kit, and stressed the importance of SOAP as the key to interoperability among SAML, GXA and other standards for security information.

Work-in-Progress Session

Ben Chinowsky
Internet2

Peter Honeyman, of the Center for Information Technology Integration at the University of Michigan, hosted a work-in-progress session on the evening of April 24.

Carl Ellison and Peter Alterman focused on details of work presented more fully in the conference proper.

- Ellison discussed Brewer's CAP postulate and its applications to cert validation. The CAP postulate states that a digital system design can achieve any two of {Consistency, Availability, tolerance of network Partitions}, but not all three.
- Alterman discussed the need for a bridge-to-bridge protocol for the emerging multiple-bridge infrastructure. Among other things, such a protocol must be able to cope with naming conflicts and transitive trust. A variety of topologies are possible: peer-to-peer, mesh, a forest of hierarchies, or a single rooted hierarchy.

Workshop chair Sean Smith gave an overview of projects currently underway at the Dartmouth PKI Lab; see `www.cs.dartmouth.edu/~pkilab/` for more information.

Burt Covnot, Mark Earnest, and Allison Mankin presented work not covered in the workshop plenary sessions.

- Covnot, of Bank of America, explored "the care and feeding of identity certificates and attribute certificates." Should customers be issued multiple identity certificates, or should their already-issued identity certificates be extended into broader services? Similarly, do multiple attribute certificates help or hinder deployment of security technologies? When certificates expire, should new keys be generated, or should an existing private key be recertified?
- Earnest recounted Penn State's experiences with DCE, drawing parallels with the challenges of PKI. Penn State plans to make use of both KX.509 and Shibboleth in its PKI migration.
- Mankin gave an update on work on DNS security. The IETF DNSSEC working group has determined that narrowly restricting the use of keys among services minimizes problems with trust transitivity. Other protocol designs being engineered attempt to reduce the complexity of client implementations. After years of technical and political work, the major players appear close to actually deploying DNSSEC.

Refereed Papers

NOVOMODO

Scalable Certificate Validation And Simplified PKI Management

by

Silvio Micali

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

(On Sabbatical Leave)

silviom@rcn.com

Conference's Areas of Inquiry: *Scalability of PKI*; *new approach to attribute certificates*; and *how the required PKI may differ from the PKI traditionally defined.*

Abstract

In [1], a scalable and small-bandwidth certificate validation scheme was presented. We call this system *NOVOMODO*, to emphasize the *new way* in which it approaches the field.

In this paper, we recall the NOVOMODO technology and

- Compare the efficiency and security of NOVOMODO and OCSP; and
- Discuss how NOVOMODO may simplify PKI management in several applications (e.g., attribute certs).

1. Traditional Certificate Validation And NOVOMODO

In essence, a digital certificate C consists of a CA's digital signature securely binding together several quantities: SN, a serial number unique to the certificate, PK, the public key of the user, U, the user's identifier, D_1, the issue date, D_2, the expiration date, and additional fields. In symbols, $C=SIG_{CA}(SN, PK, U, D_1, D_2,...)$.

It is widely recognized that digital certificates provide the best form of Internet authentication. On the other hand, they are also difficult to manage. Certificates may expire after one year (i.e., $D_2 - D_2 =$ 1 year). However, they may be revoked prior to their expiration; for instance, because their holders leave their companies or assume different duties within them. Thus, each transaction enabled by a given digital certificate needs a suitable proof of the current validity of that certificate, and that proof often needs to be archived as protection against future claims.

Unfortunately, the technologies used today for proving the validity of issued certificates do not scale well. At tomorrow's volume of digital certificates, today's validity proofs will be either too hard to obtain in a secure way, or too long and thus too costly to transmit (especially in a wireless setting). *Certificate validation is universally recognized as a crucial problem.* Unless efficiently solved, it will severely limit the growth and the usefulness of our PKIs.

1.1 Traditional Certificate Validation

Today, there are two main approaches to proving certificates' validity: Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP).

CRLs

CRLs are issued periodically. A CRL essentially consists of a CA-signed list containing all the serial numbers of the revoked certificates. The digital certificate presented with an electronic transaction is

then compared to the most recent CRL. If the given certificate is not expired but is on the list, then everyone knows from the CRL that the certificate is not valid and the certificate holder is no longer authorized to conduct the transaction. Else, if the certificate does not appear in the CRL, then the certificate is deduced to be valid (a double negative).

CRLs have not found much favor; for fear that they may become unmanageably long. (A fear that has been only marginally lessened by more recent CRL-partition techniques.) A few years ago, the National Institute of Standards and Technology tasked the MITRE Corporation [3] to study the organization and cost of a Public Key Infrastructure (PKI) for the federal government. This study concluded that CRLs constitute by far the largest entry in the Federal PKI's cost list.

OCSP

In the OCSP, a CA answers a query about a certificate C by returning its own digital signature of C's validity status at the current time. The OCSP is problematic in the following areas.

Bandwidth. Each validity proof generated by the OCSP has a non-trivial length. If RSA or other factoring based signature schemes are used, such a proof in fact requires at a minimum 2,048 bits for the CA's signature.

Computation. A digital signature is a computationally complex operation. In certain large applications, at peak traffic, the OCSP may require computing millions of signatures in a short time, which is computationally very expensive to do.

Communication (if centralized). Assume a single validation server implements the OCSP in a centralized manner. Then, all certificate-validity queries would have, eventually, to be routed to it, and the server will be a major "network bottleneck" causing considerable congestion and delays. If huge numbers of honest users suddenly query the server, a disrupting "denial of service" will probably ensue.

Security (if distributed). In general, distributing the load of a single server across several (e.g., 100) servers, strategically located around the world, alleviates network congestion. In the OCSP case, however, load distribution introduces worse problems than those it solves. In order to sign its responses to the certificate queries it receives, each of the 100 servers should have its own secret signing key. Thus, compromising any of the 100 servers is compromising the entire system. Secure vaults could protect such distributed servers, but at *great* cost.

1.2 NOVOMODO

NOVOMODO works with standard certificate formats (e.g., X.509v3) and enables a CA to prove the validity status of each certificate C at any time interval (e.g., every day, hour, or minute) starting with C's issue date, D_1. C's time granularity may be specified within the certificate itself, unless it is the same for all certificates. To be concrete, below we assume a one-day granularity for all certificates, and that each certificate expires 365 days after issuance.

One-way hashing. NOVOMODO uses a one-way hash function H (such as SHA [4]) enjoying the following properties:

1. H is at least 10,000 times faster to compute than a digital signature;
2. H produces 20-byte outputs, no matter how long its inputs; and
3. H is hard to invert: given Y, finding X such that H(X)=Y is practically impossible.

The Basic NOVOMODO System

Making a certificate C. In addition to traditional quantities such as a serial number SN, a public key PK, a user name U, an issue date D_1, an expiration date D_2 ($=D_1+365$), a certificate C also includes two 20-byte values unique to it. Specifically, before issuing a certificate C, a CA randomly selects two different 20-byte values, Y_0 and X_0, and from them computes two corresponding 20-byte values, $\mathbf{Y_1}$ and $\mathbf{X_{365}}$, as follows. Value $\mathbf{Y_1}$ is computed by hashing Y_0 once: $\mathbf{Y_1}=H(Y_0)$; and $\mathbf{X_{365}}$ by hashing X_0 365 times: $X_1=H(X_0)$, $X_2=H(X_1)$, ..., $\mathbf{X_{365}} = H(X_{364})$. Because H always produces 20-byte outputs, $\mathbf{Y_1}$, $\mathbf{X_{365}}$, and all intermediate values X_j are 20-byte long. The values $Y_0, X_0, X_1, ..., X_{364}$ are kept secret, while $\mathbf{Y_1}$ and $\mathbf{X_{365}}$ are included in the certificate: $C=SIG_{CA}(SN,PK,U,D_1,D_2,...,\mathbf{Y_1},\mathbf{X_{365}})$. We shall call $\mathbf{Y_1}$ the *revocation target* and $\mathbf{X_{365}}$ the *validity target*.

Revoking and validating a not-yet-expired certificate C. On the i-th day after C's issuance (i.e., on day D_1+i), the CA computes and releases a 20-byte proof of status for C as follows. If C is revoked, then, as a proof of C's revocation, the CA releases Y_0, that is, the H-inverse of the revocation target $\mathbf{Y_1}$. Else, as a proof of C's validity on that day, the CA releases X_{365-i}, that is, the i-th H-inverse of the validity target $\mathbf{X_{365}}$. (E.g., the proof that C is valid 100 days after issuance consists of X_{265}.) The CA may release Y_0 or

X_{365-i} by providing the value in response to a query or by posting it on the World Wide Web.

Verifying the status of a not-yet-expired certificate C. On any day, C's revocation proof, Y_0, is verified by hashing Y_0 once and checking that the result equals C's revocation target, $\mathbf{Y_1}$. (I.e., the verifier tests for himself that Y_0 really is the H-inverse of $\mathbf{Y_1}$.) Note that $\mathbf{Y_1}$ is guaranteed to be C's revocation target, because $\mathbf{Y_1}$ is certified within C. On the i-th day after C's issuance, C's validity proof on that day, X_{365-i}, is verified by hashing i times the value X_{365-i} and checking that the result equals C's validity target, $\mathbf{X_{365}}$. (I.e., the verifier tests for himself that X_{365-i} really is the i-th H-inverse of $\mathbf{X_{365}}$.) Note that a verifier knows the current day D as well as C's issuance date D_1 (because D_1 is certified within C), and thus immediately computes $i=D-D_1$.

NOVOMODO Security (Sketch)

- *A proof of revocation cannot be forged.* The proof of revocation of a certificate C consists of the H-inverse of C's revocation target $\mathbf{Y_1}$. Because H is essentially impossible to invert, once a verifier checks that a given 20-byte value Y_0 is indeed C's proof of revocation, it knows that Y_0 must have been released by the CA. In fact, only the CA can compute the H-inverse of $\mathbf{Y_1}$: not because the CA can invert H better than anyone else, but because it computed $\mathbf{Y_1}$ by starting with Y_0 and hashing it! Because the CA never releases C's revocation proof as long as C remains valid, an enemy cannot fake a revocation proof.

- *A proof of validity cannot be forged.* On day i, the proof of validity of a certificate C consists of the i-th H-inverse of C's validity target $\mathbf{X_{365}}$. Because H is essentially impossible to invert, once a verifier checks that a given 20-byte value X_{365-i} is indeed C's proof of validity on day i, it knows that the CA must have released X365-i. In fact, only the CA can compute the i-th H-inverse of $\mathbf{X_{365}}$: not because the CA can invert H better than anyone else, but because it computed $\mathbf{X_{365}}$ by starting with X_0 and hashing it 365 times, thus computing along the way all the first 365 inverses of $\mathbf{X_{365}}$! If certificate C become revoked on day i+1, the CA has already released the values X365-1,...,X_{365-i} in the preceding i days (when C was still valid) but has not released and will never release the value $X_{365-i-1}$ (or any other value X_j for $j < 365-i$) in the future. Consequently, to forge C's validity proof on day i+1, an enemy should compute on his own the i+1st H-inverse of $\mathbf{X_{365}}$ (i.e., the H-inverse of X_{365-i}), which is very hard to do! Similarly, an enemy cannot compute a validity proof for C on any day after i+1. To do so, it should again be able to invert H on input X_{365-i}. For instance, if it could compute C's validity proof on day i+2, $X_{362-i-2}$, then by hashing it once it would easily obtain $X_{365-i-1}$, the H-inverse of X_{365-i}.

NOVOMODO Efficiency

- *A certificate C includes only two additional 20-byte values,* $\mathbf{Y_1}$ and $\mathbf{X_{365}}$. This is a negligible cost. Recall that C already consists of a CA signature (at least 2048-bit long) of data that includes a public key PK (at least 1024-bit long), and that C may include comments and plenty of other data in addition to SN, PK, U, D1 and D_2.

- *Generating* $\mathbf{Y_1}$ *and* $\mathbf{X_{365}}$ *requires only 366 hashings total.* This too is a negligible cost. Recall that issuing a certificate already requires computing a signature.

- *Proofs of revocation and proofs of validity are only 20-bytes long.* Our 20-byte proofs are trivial to transmit and trivial to store, making the 20-byte technology ideal for wireless applications (because here bandwidth is still limited, and so is the storage capacity of many cellular phones and other wireless devices).

  NOVOMODO proofs can be so short because they derive their security from elementary cryptographic components, such as one-way functions, which should exhibit an exponential amount of security. (Quite differently, digital signature schemes have complex security requirements. Their typical number-theoretic implementations offer at best a sub-exponential amount of security, and thus necessitate much longer keys.)

  NOVOMODO proofs remain 20-bytes long whether the total number of certificates is a few hundred or a few billion. In fact there are 2^{160} possible 20-byte strings, and the probability that two certificates may happen to have a common proof of revocation or validity is negligible.

  Note too that the length of our 20-byte proofs does not increase due to encryption or

authentication. Our 20-byte proofs are intended to be public and thus need not be encrypted. Similarly, our 20-byte proofs are self-authenticating: by hashing them the proper number of times they yield either the validity target or the revocation target specified within the certificate. They will not work if faked or altered, and thus need not be signed or authenticated in any manner.

Finally, a 20-byte proof of validity on day i, X_{365-i}, need not additionally include the value i: in a sense, it already includes its own time stamp! Indeed, as discussed before, i is the difference between the current day and the certificate's issue day, and if hashing X_{365-i} i times yields the validity target of certificate C, then this proves that X_{365-i} is C's proof of validity on day i.

- *The 20-byte proofs are computed instantly.* A proof of revocation Y_0 or a proof of validity X_{365-i} is just retrieved from memory. (Alternatively, each X_{365-i} could be recomputed on the fly on day i; for instance by at most 364 hashings, if just X_0 is stored during certificate issuance. Surprisingly more efficient strategies are discussed in the next section.)

NOVOMODO and Wireless

NOVOMODO is ideal for wireless implementations. Its scalability is enormous: it could accommodate billions of certs with great ease. The bandwidth it requires is negligible, essentially a 30-bit serial number for the query and 20-byte for the response. The computation it requires is negligible, because a certificate-status query is answered by a single table look-up and is immediately verified. Of course, great scalability, minimum bandwidth and trivial computation make NOVOMODO the technology of choice in a wireless environment.

But there is another use of NOVOMODO that provides an additional advantage in wireless applications. Namely, every morning --e.g., at midnight-- a wireless user may receive a 20-byte proof of the validity of his certificate for the remainder of the day. (This 20-byte value can be obtained upon request of the user, or pushed to the user's cellular device automatically –e.g., by means of a SMS message or other control message..) Due to its trivial length, this proof can be easily stored in most cellular telephones and PDAs. Then, whenever the user wants to transact on that day, the user simply sends its own certificate *together with* the cert's 20-byte proof of validity for that day. Because the proof of validity is universally verifiable, the verifier of the cert and proof need not call any CA or any responder. The verifier can work totally off-line. In the cellular environment, in which any call translates into money and time costs, this off-line capability is of great value.

2. NOVOMODO vs. OCSP

NOVOMODO and OCSP are both on-demand systems: namely, a user sends a query about the current validity of a certificate and gets back an unforgeable and universally verifiable proof as a response. But there are differences in

1) Time accuracy;
2) Bandwidth;
3) CA efficiency;
4) Security; and
5) Operating costs.

TIME ACCURACY

In principle, an OCSP response may specify time with unbounded accuracy, while a NOVOMODO response specifies time with a predetermined accuracy: one day, one hour, one minute, etc. In low-value applications, one-day validity is plenty acceptable. For most financial applications, Digital Signature Trust considers a 4-hour accuracy sufficient. (Perhaps this is less surprising than it seems: for most financial transactions, orders received in the morning are executed in the afternoon and orders received in the afternoon are executed the next business day.) In any event, time is not specified by a real number with infinitely many digits. In an on-demand validation system, a time accuracy of less than one minute is seldom meaningful, because the clocks of the querying and answering parties may not be that synchronized. Indeed, in such a system, a time accuracy of 15 seconds is *de facto* real time.

To handle such an extreme accuracy, NOVOMODO needs to compute hash chains that are roughly 1M long (i.e., needs to compute validity fields of the type $\mathbf{X_{1M}}$), because there are at most 527,040 minutes in a year. If chains so long could be handled efficiently, NOVOMODO would de facto be real time. Computing 1M hashings is not problematic at certificate issuance: 1M hashings can be performed in less than 1 second even using very reasonable platforms, and a certificate is typically issued only once a year, and not under tremendous time pressure. Similarly, 1 second of computation is not problematic for the verifier of a cert validity proof (e.g., a merchant relying on the certificate) considering that he generally focuses just on an individual transaction,

and has more time at hand. Computing 1M hashings per certificate-status request would, however, affect the performance of the server producing validity proofs, because it typically handles many transactions at a time. Fortunately, this server needs not to compute all these hashings on-line starting with $\mathbf{X_0}$, but by table look up –capitalizing on having in storage the full hash-chain of every certificate. Nonetheless, storing 1M-long hash-chains may be a problem in applications with huge numbers of certificates. But, fortunately, as we shall mention later on, even ordinary servers can, using better algorithms, re-compute 1M-long hash chains with surprising efficiency.

BANDWIDTH
NOVOMODO has an obvious bandwidth advantage over OCSP. The former uses 20-byte answers, while the latter typically uses 256 bytes.

CA EFFICIENCY
A validity query is answered by a (complex) digital signature in the OCSP case, and by a (trivial) table look-up in the NOVOMODO case, as long as the CA stores the entire X-chain for each certificate.

Note that, with a population of 1 million certificates, the CA can afford to store the entire X-chain for each certificate when the time accuracy is one day or one hour. (In the first case, the CA would have to store 365 20-bytes values; that is, 7.3K bytes per cert, and thus 7.3B bytes overall. In the second case, 175.2B bytes overall.) If the time accuracy were 15 seconds, then each hash chain would consist of 1M 20-byte values, and for the entire system the overall storage requirement would be around 10.5 tera-bytes: a sizable storage.

To dramatically decrease this storage requirement, the CA may store just a single 20-byte value (i.e., $\mathbf{X_0}$) for each cert, and re-compute from it each $\mathbf{X_i}$ value by at most 1M hashings. Alternatively, Jacobsson [5] has found a surprising time/storage tradeoff. Namely, the CA may re-compute all n $\mathbf{X_i}$ values, in the right order, by storing log (n) hash values and performing log(n) hashings each time. If n were 1M, this implies just storing 20 hash values per cert and performing only 20 hashings each time the cert needs validation. Other non-trivial tradeoffs are possible. In particular, for our 1M-chain case, Reyzin [R] has shown that a CA can compute all $\mathbf{X_i}$ values (i=1M down to 1) by storing only 3 hash values and performing at most 100 hashings each time.

In sum, even in a de facto real-time application (i.e., using a 15-second time accuracy) NOVOMODO can, by just storing 60 bytes per cert, replace a complex digital signature operation with a trivial 100-hash operation.

SECURITY AND OPERATING COSTS
The last two differences are better discussed after specifying the type of implementation of NOVOMODO and OCSP under consideration.

Centralized NOVOMODO vs. Centralized OCSP: Security Analysis

Whenever proving certificate validity relies on the secrecy of a given key, a secure *vault* ought to protect that key, so as to guarantee the integrity of the entire system. By a centralized implementation of NOVOMODO or OCSP, we mean one in which a single vault answers all validity queries. Centralized implementations are preferable if the number of deployed certificates is small (e.g., no more than 100K), so that the vault could handle the query volumes generated even if almost all certificates are used in a small time interval, triggering almost simultaneous validity queries. In such implementations, NOVOMODO is preferable to OCSP in the following respects.

CENTRALIZED NOVOMODO OFFERS BETTER DOOMSDAY PROTECTION
In the traditional OCSP, if (despite vaults and armored guards) an enemy succeeds in penetrating the vault and compromises the secret signing key, then he can both "resurrect" a previously revoked certificate and "revoke" a still valid one. (Similarly, if the CRL signing key is compromised in a CRL system.) By contrast, in NOVOMODO penetrating the secure vault *does not help an adversary to forge the validity of any previously revoked certificate*. In fact, when a certificate becomes revoked at day i, not only is its revocation proof Y_0 made public, but, simultaneously, all its X_i values (or at least the values X_0 through X_{365-i}) are deleted. Therefore, after a successful compromise, an enemy finds nothing that enables him to "extend the validity" of a revoked certificate. To do so, he should succeed in inverting the one-way hash H on X_{365-i} without any help, which he is welcome to try (and can indeed try without entering any secure vault). The worst an enemy can do in a NOVOMODO system after a successful compromise is to fake the revocation of valid certificates, thus preventing honest users from authenticating legitimate transactions. Of course, this would be bad, but *not as bad as enabling dishonest users to authenticate illegitimate transactions.*

Distributed NOVOMODO vs. Distributed OCSP: Security and Operating-Cost Analysis

Centralized implementations of NOVOMODO and OCSP require all queries about certificate validity to be routed to the same vault. This easily results in long delays and denial of service in applications with millions of active certificates. To protect against such congestion, delays, and denial of service, one might spread the load of answering validity queries across several, geographically dispersed, responder servers. However, in the case of the OCSP each additional responder needs to have a secret signing key, and thus needs to be hosted in a vault, making the cost of ownership of an OCSP system very onerous. A high-grade vault meeting the requirements of financial institutions costs at least $1M to build and $1M to run. (A good vault would involve armored concrete, steel doors, back-up power generators, protected fuel depot to run the generator for potentially a long time, etc. Operating it would involve a minimum of 4 different teams for 24X7X365 operations, plus managerial supervision, etc.) In an application requiring 10 such vaults to guarantee reasonably fast response at peak traffic, the cost of ownership of the OCSP system would be $10M of initial investment and an ongoing budget of $10M/year. Even if less secure vaults and operations were used, millions of dollars in initial and ongoing costs would still be necessary.

In the NOVOMODO case, however, a distributed implementation can be achieved with a single vault (which a CA would have anyway) and an arbitrary number of "un-trusted responders" (i.e., ordinary servers). Let us see the exact details of a distributed NOVOMODO system assuming, to be concrete, that (a) there are 10M certs; (b) there are 1,000 servers, strategically located around the globe so as to minimize response time; and (3) the time granularity is one-day.

Distributed NOVOMODO: CA Operations *(Initialization Cost)*
Every morning: Starting with the smallest serial number, compile a 10M-entry array F as follows: For each certificate C having serial number j, store C's 20-byte validity/revocation proof in location j. Then, date and sign F and send it to each of the 1,000 servers.

Distributed NOVOMODO: User Operations *(Query Cost)*
To learn the status of a certificate C, send C's serial number, j, (and CA ID if necessary) to a server S.

Distributed NOVOMODO: Server Operations *(Answer Cost)*
Every morning: If a properly dated and signed array F is received, replace the old array with the new one.
At any time: answer a query about serial number j by returning the 20-byte value in location j of the current F.

Distributed NOVOMODO Works:

1. *Preparing Array F is instantaneous.*
   If the whole hash chain is stored for each cert, then each entry is computed by a mere table look-up operation. (Else, it can be computed on the spot by using Reyzin's method.)
2. *F contains no secrets.*
   It consists of the accurate and full account of which certificates are still valid and which revoked. (The CA's goal is indeed making this non-secret information as public as possible in the most efficient manner)
3. *Transferring F to the servers is straightforward.*
   This is so because F contains no secrets, requires no encryption, and poses no security risks. Though 10M certs are a lot, sending a 200M-byte file to 1000 servers at regular intervals is very doable.
4. *Each server answer is 20-byte long.*
   Again, each answer requires no encryption, signature or time stamp.
5. *No honest denial of service.*
   Because each value sent is just 20-byte long, because each such a value is immediately computed (by a table look up), and because the traffic can be spread across 1000 servers, no denial of service should occur, at least during legitimate use of the system.
6. *Servers need not be trusted.*
   They only forward 20-byte proofs received by the CA. Being self-authenticating, these proofs cannot be altered and still hash to the relevant targets.

DISTRIBUTED NOVOMODO OFFERS BETTER CA SECURITY
Distributed NOVOMODO continues to enjoy the same doomsday protection of its centralized counterpart: namely, an enemy successfully entering the vault cannot revive a revoked certificate. Sophisticated adversaries, however, refrain from drilling holes in a vault, and prefer software attacks whenever possible. Fortunately, software attacks, though possible against the distributed/centralized OCSP, cannot be mounted against Distributed NOVOMODO.

In the OCSP, in fact, the CA is required to receive outside queries from untrusted parties, and to answer them by a digital signature, and thus by means of its precious secret key. Therefore, the possibility exists that OCSP's required "window on the outside world" may be maliciously exploited for exposing the secret signing key.

By contrast, in distributed NOVOMODO there are no such "windows:" the CA is in the vault and never receives or answers any queries from the outside; it only outputs non-secret data at periodic intervals. Indeed, every day (or hour) it outputs a file F consisting of public information. (The CA may receive revocations requests from its RAs, but these come from fewer trusted entities via authenticated channels ---e.g., using secure smart cards.) The untrusted responders do receive queries from untrusted parties, but they answer those queries by means of their file F, and thus by public data. Therefore, a software attack against NOVOMODO ordinary responders may only "expose" public information.

3. NOVOMODO and Simplified PKI Management

PKI management (e.g., [7] [8]) is not trivial. NOVOMODO may improve PKI management in many applications by

- Reducing the number of issued certs;
- Enabling privilege management on the cert; and
- Sharing the registration function with multiple independent CAs.

Let us informally explain these improvements in PKI management in a series of specific examples. (Note that features and techniques used in one example can be easily embedded in another. We do not explicitly do this to avoid discussing an endless number of possible variations.)

3.1 Turning a Certificate ON/OFF (and Suspending It)

EXAMPLE 1: MUSIC DOWNLOADING
Assume an Internet music vendor wishes to let users download any songs they want, from any of its 1000 servers, for a $1/day fee. This can be effectively accomplished with digital certificates. However, in this example, U may be quite sure that he will download music a few days of the year, yet he cannot predict *which* or *how many* these days will be. Thus the Music Center will need to issue for U a different one-day certificate whenever U so requests: U requests such a certificate and, after payment or promise of payment, he receives it and then uses with any of the 1000 music servers on that day. Issuing a one-day cert, however, has non-trivial management costs both for the vendor and the user. And these costs must be duplicated each time the user wishes to enjoy another "music day."

NOVOMODO technology can alleviate these costs as follows. The first time that U contacts the vendor, he may be issued a certificate C with issue date $D_1=0$, expiration date $D_2= 365$, and a validity field $\mathbf{X_{365}}$, a revocation target $\mathbf{Y_1}$, and a suspension field $\mathbf{Z_{365}}$. (The vendor's CA builds the suspension field very much as a validity field: by starting with a random 20-byte value Z_0 and then hashing it 365 times, in case of one-day granularity. It then stores the entire hash chain, or just Z_0, or uses a proper time/storage method to be able to generate any desired Z_i.) At day i=1,...,365, if U requests "a day of music" for that day, then the vendor simply releases the 20-byte value X_{365-i} to indicate that the certificate is valid. Else, it releases Z_{365-i} to indicate that the certificate is "suspended." Else, it releases Y_0 to indicate that the certificate is revoked.[1]

That is, rather than giving U a new single-day certificate whenever U wishes to download music, the vendor gives U a *single, yearly* certificate. At any time, this single certificate can be turned ON for a day, by just releasing the proper 20-byte value. Thus, for instance, NOVOMODO replaces issuing (and embedding in the user's browser) 10 single-day certificates by issuing a single yearly cert that, as it may happen, will be turned ON for 10 out of the 365 days of the year.[2]

3.2 Turning ON/OFF Many Certificates For The Same User

[1] Optionally, if U and the music vendor agree to --say-- a "week of music starting at day i," then either the 20-byte values for those 7 days are released at the proper time, or the single 20-byte value $X_{365-i-7}$ is released at day i.

[2] The vendor could also use the method above to issue a cert that specifies a priori the number of days for which it can be turned ON (e.g., a 10-day-out-of 365 cert). Because it has a more predictable cost, such certs are more suitable for a gift.

EXAMPLE 2: SECURITY-CLEARANCE MANAGEMENT

Digital certificates work really well in guaranteeing that only proper users access certain resources. In principle, privileges could be specified on the cert itself. For instance, the State Department may have 10 different security-clearance levels, L1,...L10, and signify that it has granted security level 5 to a user U by issuing a certificate C like

$$C=SIG_{SD}(SN,PK,U, L5, D_1,D_2,...)$$

Where again $D_1$and D_2, represent the issue and expiration dates.

However, specifying privileges on the cert itself may cause a certificate-management nightmare: whenever its privileges change, the cert needs to be revoked. Indeed, the security level of an employee may vary with his/her assignment, which often changes within the same year. For instance, should U's security-clearance level be temporarily upgraded to 3, then the State Department should revoke the original C and issue a new cert C'. This task could be simplified somewhat by having U and thus C' retain the same public key (and expiration date) as before; for instance, by having

$$C'=SIG_{SD}(SN',PK,U, L3, D_1',D_2,...).$$

However, U still faces the task of "inserting" the new C' into his browser in a variety of places: his desk-top PC, his lat-top, his cell phone, his PDA, etc. Now, having the CA take an action to re-issue a certificate in a slightly different form is one thing, but counting on users to take action is a totally different thing!

This management problem is only exacerbated if short-lived certificates (e.g. certificates expiring one day after issuance) are used. In the context of the present example, single-day certs may enable a State Department employee or user U to attend a meeting where a higher security level is needed. (If U had such a cert in a proper cellular device, smart card or even mag stripe card, he could, for instance, use it to open the door leading to the meeting that day.) The use of short-lived certificates is much broader, and has been advocated because it dispenses with the difficulty of revocation to a large extent (no point revoking a cert that will expire in 24hours, at least in most applications). However, issuing short-lived certs so that they reside in all pertinent users' browsers still is a management cost.

These management costs can be alleviated with use of NOVOMODO as follows. Assuming that one-day time accuracy is enough, the State Department issues to a user U a certificate containing 10 validity fields and 1 revocation field: e.g.,

$$C=SIG_{SD}(SN,PK,U,D_1,D_2, \mathbf{A_{365}}, \mathbf{B_{365}}, \mathbf{C_{365}}, \mathbf{D_{365}}, \mathbf{E_{365}}, \mathbf{F_{365}}, \mathbf{G_{365}}, \mathbf{H_{365}}, \mathbf{I_{365}}, \mathbf{J_{365}}, \mathbf{Y_1},)$$

where the first validity field, $\mathbf{A_{365}}$, corresponds to security-clearance level 1 ... and the 10th validity field, $\mathbf{J_{365}}$, corresponds to security-clearance level 10, while, as usual, $\mathbf{Y_1}$ is C's revocation field. Cert C is used as follows. If, on day n, U is in good standing (i.e., cert C is still valid), and U's security-clearance level is 5, then the State Department publicizes (e.g., sends to all its responders in a distributed NOVOMODO implementation) the 20-byte validity proof $\mathbf{E_{365-n}}$. If, on day m, U's security-clearance level becomes 2, then the State Department publicizes $\mathbf{B_{365-m}}$. And so on. As soon as C becomes invalid (e.g., because U is terminated as an employee or because U's secret key is compromised), then the State Department publicizes $\mathbf{Y_0}$ (and erases "future" A, B, C, D, E, F, G, H, I, and J values from its storage).

This way, cert C, though internally specifying its own privileges, needs *not* be revoked when these privileges change in a normal way, and users need *not* load new certs in their browsers. In essence, NOVOMODO has such minimal footprint, that a CA (rather than issuing, revoking, and re-issuing many related certs) can issue with great simplicity a single cert, having a much higher probability of not being revoked (because changes of security-clearance level do not translate into revocation). As a result, fewer certs will end up been issued or revoked in this application, resulting in simpler PKI management.

In sum,

> *NOVOMODO replaces the complex certificate management relative to a set of dynamically changing properties or attributes by a single certificate (with minimum extra length) and a single 20-byte value for attribute.*

Telecom companies may use a method similar to that of Example 2 to switch a given wireless device from one rate plan to another, or for roaming purposes.

3.3 Landlord CAs and Tenant CAs

A main PKI cost is associated to the RA function. Indeed, identifying a user U may require an expensive personal interview and verifying that indeed U knows the right secret key (corresponding to the to-be-certified public key PK). It would be nice if this RA function could be shared across many CAs,

while enabling them to retain total independent control over their own certs.

EXAMPLE 3: ORGANIZATION CERTIFICATES

The Government and big organizations consist of both parallel and hierarchical sub-organizations: departments, business units, etc. An employee may be affiliated with two or more sub-organizations. For instance, in the U.S. Government, he may work for NIST and the Department of Commerce. Issuing a digital certificate for each such affiliation results in a high total number of certificates and a complex PKI management: every time an employee drops/adds one of his/her affiliations, it is best to revoke the corresponding cert/issue a new one. Ideally, two opposites should be reconciled: (1) The Organization issues only one cert per employee, and (2) Each Sub-Organization issues and controls a separate cert for each of its affiliates.

These two opposites can be reconciled by NOVOMODO as follows. To begin with, notice that NOVOMODO is compatible with de-coupling the process of certification from that of validation, the first process being controlled by a CA and the second by a validation authority (VA). For instance, assuming a one-day time accuracy, once a CA is ready to issue a certificate C with serial number SN, it sends SN to a VA, who selects Y_0 and X_0, secretly stores the triplet (SN, Y0,X0), computes as usual $\mathbf{Y}_1$ and $\mathbf{X}_{365}$, and then returns $\mathbf{Y}_1$ and $\mathbf{X}_{365}$ to the CA, who includes them within C. This way, the CA need not bother validating C: the CA is solely responsible for identifying the user and properly issuing C, while the VA is the only one who can prove C valid or revoked. This de-coupling may be exploited in a variety of ways in order to have *organization certificates* that flexibly reflect internal sub-organization dynamics. The following is just one of these ways, and uses Government and Departments as running examples. The Government as a whole will have its own CA, and so will each Department.

Envisaging k different Departments with corresponding CAs, $CA^1 \ldots CA^k$, and one-day time accuracy, a Government certificate C has the following form:

$$C=SIG_{GOV}(SN,PK,U, D_1, D_2, \mathbf{X}_{365}, \mathbf{Y}_1, [\mathbf{X}_{365}^1, \mathbf{Z}_{365}^1],\ldots, [\mathbf{X}_{365}^k, \mathbf{Z}_{365}^k])$$

where, as usual, SN is the cert's serial number, PK the public key of the user, U the user's identity, D_1 the issue date, D_2 the expiration date, $\mathbf{X}_{365}$ the validity field, $\mathbf{Y}_1$ the revocation field, and where

$\mathbf{X}_{365}^j$ is the validation field of CA^j; and
$\mathbf{Z}_{365}^j$ is the suspension field of CA^j.

Such a certificate is generated by the Government CA with input from the Department CAs. After identifying the user U and choosing a unique serial number SN, the issue date D_1, and the expiration date D_2, the Government CA sends SN, PK, U, D_1, D_2 (preferably in authenticated form) to each of the Department CAs. The jth such CA then

- chooses two secret 20-byte values $\mathbf{X}_0^j$ and $\mathbf{Z}_0^j$,
- locally stores (SN,PK,U, D_1, D_2, $\mathbf{X}_0^j$, $\mathbf{Z}_0^j$) or, more simply, (SN, $\mathbf{X}_0^j$, $\mathbf{Z}_0^j$); and
- returns [$\mathbf{X}_{365}^j$, $\mathbf{Z}_{365}^j$] for incorporation in the Government certificate in position j (or with "label" j).

This certificate C is managed with Distributed NOVOMODO as follows, so as to work as a 1-cert, a 2-cert,..., a k-cert; that is, as k independent certs, one per Department. On day n, envisaging 100 responders,

- the Government CA sends all 100 responders the 20-byte value $\mathbf{X}_{365-n}$ if C is still valid, and Y_0 otherwise.
- the jth Department CA sends all 100 responders the 20-byte value $\mathbf{X}_{365-n}^j$ to signify that C can be relied upon as a j-cert and $\mathbf{Z}_{365-n}^j$ otherwise.

Therefore, the Government CA is solely responsible for identifying the user and issuing the certificate, but each of the Department CAs can *independently* manage what de facto is *its own* certificate. (This is absolutely crucial. If CA^1 were the Justice Department and CA^2 the DOD, then, despite some overlapping interests, it is best that each acts independently of the other.) The resulting certificate system is very economical to run. First, the number of certs is greatly reduced (*in principle*, there may be just one cert for employee). Second, a given employee can leave and join different Departments without the need of revoking old certs or issuing new ones. Third, different Department CAs may share the same responders. (In fact, whenever the mere fact that a given user is affiliated with a given Department is not a secret –something that will be true for most departments-- the servers essentially contain only "publishable information".) Thus a query about the status of C as a j-certificate is answered with two 20-byte values: one as a Government cert and one as a j-cert. This enables one to more nimbly revoke C at a "central level" (e.g., should U lose the secret key corresponding to PK).

POSSIBLE ALTERNATIVES

In the above example, certificate C was only revocable in a central way, but it could easily be arranged that the responsibility of revocation is push down to individual Departments. For instance, to enable the jth Department CA, in full autonomy, to revoke as well as suspend C as a j-certificate, C may take the following form:

$$C=SIG_{GOV}(SN,PK,U, D_1, D_2, [\mathbf{X}_{N1}^{1}, \mathbf{Y}_{1}^{1}, \mathbf{Z}_{N1}^{1}],\ldots, [\mathbf{X}_{Nk}^{k}, \mathbf{Y}_{1}^{1}, \mathbf{Z}_{Nk}^{k}]).$$

Also, different departments may have different time accuracies for their own certs. This too can be easily accomplished by having C of the following format,

$$C=SIG_{GOV}(SN,PK,U, D_1, D_2, [TA^{1}, \mathbf{X}_{N1}^{1}, \mathbf{Y}_{1}^{1}, \mathbf{Z}_{N1}^{1}],\ldots, [TA^{k}, \mathbf{X}_{Nk}^{k}, \mathbf{Y}_{1}^{1}, \mathbf{Z}_{Nk}^{k}])$$

where

TA^j is the time accuracy of the jth CA; and
Nj is the number of time units between D_1 and D_2. (E.g., if TA^j is one day and $D_1 - D_2 = 1$ year, then $\mathbf{X}_{Nj}^{j} = \mathbf{X}_{365}^{j}$.)

LANDLORD CAs, TENANT CAs, AND LEASED CERTS

Within a single organization, one major advantage of issuing certs structured and managed as above consists in enabling the cert to stay alive though the user moves from one sub-organization to another. It should be realized, however, that the above NOVOMODO techniques are also applicable outside a single-organization domain. Indeed, the Government CA can be viewed as a *landlord CA*, the k Department CAs as *tenant CAs* servicing *unrelated* organizations (rather than sub-organizations), and the certificate can be viewed as a *leased cert*. This terminology is borrowed from a more familiar example where the advantages of "joint construction and independent control" apply. Leased certs are in fact analogous to spec buildings having the identical floor footprints. Rather than building just his own apartment, a builder is better off constructing a 20-floor building, setting himself up in the penthouse apartment and renting or selling out right the other floors. Each of the 20 tenants then acts as a single owner. He decides in full autonomy and with no liability to the builder whom to let into his flat, and whom to give the keys. A 20-story building is of course less expensive than 20 times a single-story one: it may very well cost 10 times that. This economy of scale is even more pronounced in a leased cert. Indeed, the cost of issuing a regular cert and that of issuing a leased one is pretty much the same. Thus issuing leased certs could be very profitable to a landlord CA, or at least repay it completely of the costs incurred for its own certs. On the other hand, tenant CAs have their advantage too, in fact

1. *they save on issuance costs*: they share the cost of issuing a cert k ways; and
2. *they save on infrastructure costs*: they share the same responders (since they contain only public data).

Natural candidates to act as landlord CAs for external tenant CAs are:

- credit card companies;
- large financial institutions, and again
- the Government (e.g., via the USPS or the IRS).

In many cases, in fact, they have long and close relationships with millions of "users" and may more easily issue them a digital cert without investing too many resources for user identification (e.g., a credit card company has been sending bills for years to its customers, and can leverage this knowledge). A credit card company may like the idea of issuing certificates as a landlord CA in order to run more effectively its own affinity program (having hotel chains, airlines etc. as their tenants). The IRS may have already decided to use digital certificates, and leased certs may later on provide them with a revenue stream that will repay of the costs incurred for setting up a faster and better service.

FURTHER ALTERNATIVES

So far, the way we have described landlord and tenant CAs requires that the landlord CA cooperates with its own tenant CAs during the issuance process, and thus that it has already identified its tenant CAs beforehand. It is actually possible, however, for a landlord CA to issue rental certs envisioning –say— 20 tenant CAs, without having identified all or any of these tenants. Rather, future tenant CAs will be able to rent space in already issued certs. This capability is ideal for new cert-enabled applications. Rather than undergoing the expenses necessary to issue certs to millions of customers, a company offering a new certificate-enabled product may approach a landlord CA having issued millions of certs, rent space in them *after the facs*, and then sign on as customers a large portion of the landlord-CA users by *turning ON* all their corresponding certs *overnight* (without any customer identification and other issuing costs) and then starting managing them according to its own

criteria. We shall describe various techniques for enabling this functionality in a forthcoming paper.

References

[1] *Efficient Certificate Revocation*; by Silvio Micali; Proceedings 1997 RSA Data Security Conference.

[2] *Online Certificate Status Protocol*, version 2. Working document of the Internet Engineering Task Force (IETF) RFC 2560.

[3] Public Key Infrastructure, Final Report; MITRE Corporation; National Institute of Standard and Technology, 1994.

[4] *Secure Hash Standard*; FIPS PUB 180, Revised July 11, 94 (Federal Register, Vol. 59, No. 131, pp. 35211-34460); revised August 5, 1994 (Federal Register Vol. 59, No. 150, pp. 39937-40204).

[5] *Low-Cost Hash Sequence Traversal*; by Markus Jakobsson; To appear in Financial Cryptography 2002.

[6] *General Time/Storage Tradeoffs for Hash-Chain Re-computation*; by Leo Reyzin; unpublished manuscript.

[7] *Internet Public Key Infrastructure, Part III: Certificate Management Protocols*; by S.Farrell, A. Adams, and W. Ford; Internet Draft, 1996

[8] *Privacy Enhancement for Internet Electronic Mail – Part II: Certificate-Based Key Management*; by S. Kent and J. Linn; 1989.

Validity Management in SPKI

Yki Kortesniemi
Helsinki University of Technology
Yki.Kortesniemi@hut.fi

ABSTRACT

In a distributed system, using authorisation certificate based access control tends to facilitate the granting of rights. On the other hand, the problems of limiting usage or revoking the rights become more difficult, as the issuer of the right is no longer in control of the issued certificate.

In this paper we take a look at the role of certificates in access control, evaluate the technical merits of different validity management mechanisms an SPKI authorisation certificate supports, discuss the problems related to managing the validity and finally introduce a protocol for validity management.

1. Introduction

Access control becomes an interesting question whenever an entity controls some resource that others would like to use. In the absence of control, a resource likely ends up being exploited without any benefit to the owner. A computer related example is the protection of a database system. Traditionally, this has been implemented using an ACL (Access Control List), which lists the authorised usernames and their associated rights. This solution has many good qualities in mainframe-type systems, but in a distributed environment with multiple instances of the database, problems arise because we are relying on a central list. Solutions like replication can be used to lessen the impact, but essentially an ACL is a centralised solution.

Authorisation certificates, on the other hand, yield themselves quite naturally to a distributed environment. SPKI certificates, for instance, can successfully be used to implement systems that support anonymity, delegation and dynamic distributed policy management – all qualities not traditionally associated with ACLs. The key idea in authorisation certificates is to give the user an unforgeable ticket, which states the user's rights, thus making ACLs unnecessary. The verifier monitoring the resource simply has to make sure that the certificate is valid, originates from the verifier and has been granted to the user in question, before giving the user access to the resource. It is interesting to note that Kerberos combines elements from both ends: it maintains the long term access information in the server's database (ACL), but the actual access control decisions are based on short-lived tickets not unlike certificates. However, actual authorisation certificates tend to be much longer-lived and do not normally rely on a backing ACL.

The self-containment is a strong point of authorisation certificates, but also the source of one of their weaknesses: the difficulty of revoking them. With ACLs, revocation is easy: just erase the unwanted entries. With certificates, the problem is more complicated, because instead of the issuer, the user is in control of the certificate. Therefore, all the revocation solutions for SPKI certificates rely on additional online checks. By using online servers, we lose the self-containment, but this loss is often acceptable. Nevertheless, using these revocation mechanisms always has a performance impact on the system, and they should therefore be used with consideration.

The immutability of certificates, unfortunately, also makes it difficult to keep track of the amount of usage – we cannot just cross out a part of the certificate as a sign of usage, we need other methods. One solution proposed in [10] is to use online servers to keep track of usage, thus enabling the use of tickets that are valid 10 times or credit cards that have monthly limits. However, managing this limit presents some problems.

In this paper, we take a look at the validity management options of one particular authorisation certificate, namely Simple Public Key Infrastructure (SPKI) certificate[7][8], study the problems of managing them and finally offer a solution in the form of a revocation management protocol.

The intended application domains could include things like organisations, which want to control their internal access rights – in these cases the users identity is usually known by the administrators granting the rights and the user might have several rights assigned to the same public key. At the other end we have global applications, where consumers buy some access rights with cash (e.g. the right to read the current issue of a particular magazine) and want to stay anonymous. In this case, the user might create a new public key for every right bought just to enhance privacy.

The rest of the paper is organised as follows: we first look at access control and how certificates can be used for it. Then, we look at SPKI certificates and their validity management methods, discuss their suitability for different situations and finally present a protocol for managing the online servers.

2. Access Control and certificates

The goal of access control is to make sure that only authorised users (be they humans or computers) get access to the protected resources. The access control process therefore can be said to consist of the following phase (depicted in Figure 1):

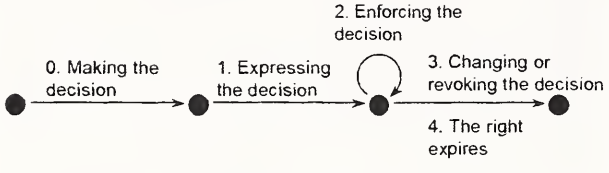


Figure 1. Phases of access control.

0. Making the decision
   In this phase, the issuer (someone either owning the resource or having the right to control access to it) makes the decision to grant a subject the right to access the resource. This decision could be based on things like the issuer knowing the subject (a friend), the subject holding some position in issuer's organisation or the subject being a paying customer to issuer's service.

1. Expressing the decision
   For the decision to be automatically enforced, it has to be expressed in a precise format. This could be e.g. an ACL entry or an authorisation certificate.

2. Enforcing the decision
   Whenever the subject tries to use the resources, the validator makes sure that the right still exists. This could entail checking the subject's credentials or the ACL and verifying that the subject is indeed the same as mentioned in the credentials or in the ACL.

3. Changing or revoking the decision
   Should the access right become insufficient, unnecessary or should there be risk of misuse, the right can be changed or even revoked.

4. The right expires
   Eventually, the right expires, either intentionally, or implicitly.

2.1. Different types of certificates

There exist three major types of certificates: identity certificates (e.g. X.509 and PGP), authorisation certificates (e.g. SPKI) and attribute certificates as shown in Figure 2.

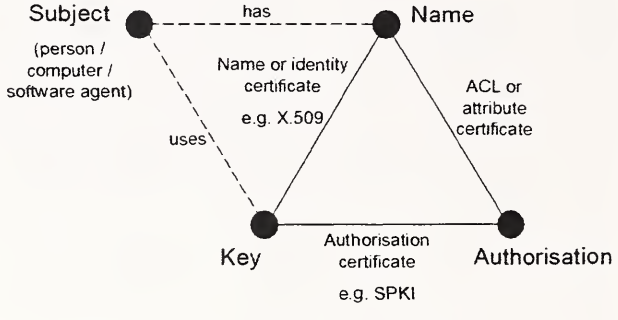


Figure 2. Three major types of certificates.

An identity certificate binds a public key to a name so that outside parties can be convinced that a particular person uses a particular key. Of course, this entails that the issuer (typically an organization called Certification Authority, CA) actually makes sure that the key is controlled by said person. Hence, CAs must be trusted by all users and they tend to be large organisations.

An authorisation certificate, on the other hand, binds a right to a public key. Authorisation certificates can be issued by anyone owning a resource or having the right to grant access to someone else's resource. This means that potentially every human, computer, or even a software agent could be issuing certificates. This difference in the number and resources of issuers between the two certificate types has significant implications on the revocation systems used, as we'll later discuss.

The third, a less common type, an attribute certificate, is used to bind an authorisation to a name. Essentially, it is a distributed version of an ACL.

To better appreciate the differences between identity and authorisation certificates, let us briefly look at how they are utilised in phases 1 and 2 of the access control process. In phase 0, certificates play no role, and the role of authorisation certificates in phases 3 and 4 is the subject for the rest of the paper. In this discussion, we assume the usage of public key based authentication.

2.2. Using certificates in phase 2: Enforcing the decision

To fulfil phase 2 in the access control process, we have to prove the binding between the subject requesting access and the required right. As we can see from Figure 2, there are several ways of doing this. In all of these, the binding between the subject and the key is assumed much tighter than the binding with password. This assumption however does not always hold, as the subject can either lose the control or just give the required private key away. In both these situation, revocation of that key and the associated rights is normally required.

The most common way of using certificates is to use identity certificates to establish a mapping from the key to a name and then use ACLs or attribute certificates to map the name to an authorisation. This approach nicely extends existing solutions, but it also has many problems:

- By design, it makes anonymous usage impossible. In some system, it is a requirement to prevent anonymous usage, but in other cases knowing the user's identity is not a necessity; it merely promotes unnecessary monitoring of users.

- Making a tight binding through the name is not easy, as it requires names that unique within the application domain – otherwise namesakes can share their rights. If we have a small organisation, this might be quite feasible, but even in a moderately sized organisation there can be more than one John Smith and we have to be very careful never to mix them up. And if we make global consumer applications, we need globally unique names, which are difficult for humans and impractical for computers. The local names can be complemented with additional information to make them global, but for global applications it is more straightforward to use global identifiers like public keys from the beginning.

- The binding from a key to an authorisation is unnecessarily long – it consists of two steps: key to name and name to authorisation. This is an important aspect, as the verification of this binding will be performed many times – in fact, every time the subject uses the resource.

However, this two step binding does present an advantage: by revoking the identity certificate we can automatically revoke all the associated rights (naturally, this is an advantage only if there are several rights associated to a single certificate). Further, we can create a similar construct with authorisation certificates, if necessary, so this is not a unique advantage of identity certificates.

An authorisation certificate, on the other hand, makes a direct binding from the key to the authorisation. This makes the binding simpler, but also practically anonymous. In reality, the key is a pseudonym, but since these pseudonyms do not have to be registered anywhere, it can be very difficult to trace them back to the user's identity. And, should the anonymity become a problem, it can be circumvented by verifying the subject's identity already in phase 1 (but if this is omitted, we cannot perform it retroactively).

Based on the above, we can conclude that authorisation certificates offer a simpler solution to phase 2 than solutions based on identity certificates.

2.3. Using certificates in phase 1: Expressing the decision

This phase is a more natural application area for identity certificates. They are often used to get the unique name of the subject, which is then used in the ACL or in an attribute certificate. But as we saw, this approach presents some problems.

Another way of using identity certificates is to acquire the known user's public key to issue them an authorisation certificate. This applies to situations such as issuing rights to members of an organisation. It should be noted, however, that identity certificates are not always necessary for issuing authorisation certificates. For instance, we could receive the public key directly from the subject in a face-to-face meeting, in which case an identity certificate is unnecessary.

2.4. Additional advantage of authorisation certificates - delegation

If the certificate does not expressly forbid it, it is possible to delegate the rights listed in the certificate to other users without any help from the owner of the resource - a feature, which makes distributed management easier to organise than in centralised solutions. In fact, regular users can delegate their own rights. For example, this means that we can implement a scheme, where a parent can issue a copy of her credit card to a child and limit the amount the child can charge from the card, while still keeping her own credit card [9].

The downside of this flexibility is that the certificate chains can become very long and evaluating them is no longer trivial. The solution is to view the chains as a means of implementing the granting of rights and then let the verifier automatically create a reduction certificate that replaces the chain with a single certificate, thus making the usage of the right efficient.

3. The SPKI Certificates

The Internet Engineering Task Force (IETF) has been developing SPKI as a more flexible alternative to X.509. The key idea is that anyone (or anything) with access to a resource can authorise others to use the resource by issuing them an authorisation certificate. So, compared to X.509, where only CAs issue certificates, in SPKI any person, computer, etc. can issue certificates - and also has to be able to manage their validity.

Altogether there are six validity options in SPKI certificates. The simplest and the only locally evaluateable is the validity period. In addition, the current SPKI structure includes three online validity checks: CRLs, revalidations and one-time checks. Furthermore, [10] proposes formats for two additional online validity checks: limit and renew. As we shall later see, the different methods can be ordered by increasing capability. Therefore, using more than one online method in the same certificate is usually redundant since the most capable suffices (although the selected method can be used several times).

The author's model for the lifecycle of an SPKI certificate is depicted in Figure 3. Each new certificate begins its life in the suspended state (transition 1), but the certificate moves to the available state when its validity period, crl and reval permit, possibly even immediately (transition 2). In the available state, the certificate can be used, provided that one-time and limit agree (transition 3). Should the crl or reval methods be used to revoke the certificate, it moves to the suspend state if it can later become available again (transition 4), or to the expired state if it no longer can be made available (transition 5 and 6). Finally, the certificate should naturally expire as dictated by the validity period (transitions 7 and 8). The renew method (transition 9) complements the model by forming a chain of shorter lived certificates – once a short-lived certificate expires, the subsequent one is ready to take its place (though it could be argued that the validity periods of consecutive certificates might be allowed to overlap).

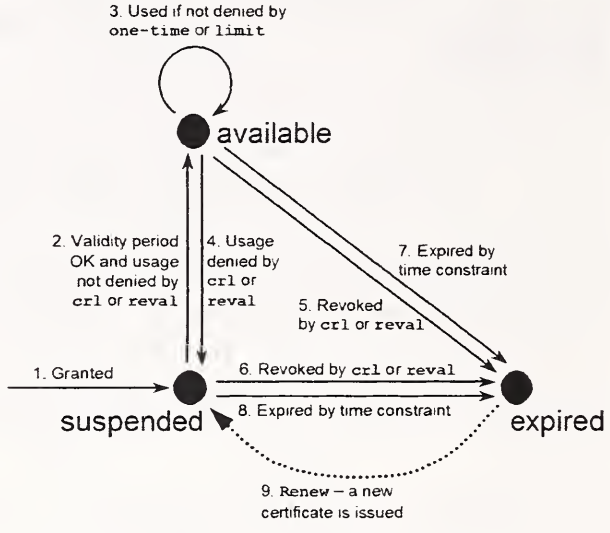


Figure 3. The lifecycle of an SPKI certificate.

3.1. Validity periods

In SPKI, the validity period definition consists of two parts:

```
<not-before>::
    "(" "not-before" <date> ")" ;

<not-after>::
    "(" "not-after" <date> ")" ;
```

Both parts are optional, and if either one is missing, the certificate is assumed to be valid for all time in that direction.

```
<valid-basic>::
    <valid-date> | <valid-dates> ;

<valid-date>::
    <not-before> | <not-after> ;

<valid-dates>::
    <not-before> <not-after> ;
```

There is an additional type of validity period called ``now", which has a length of 0, and can only be the result of an online check. It is interpreted to mean that the certificate is valid the moment the validation request was made, but it states nothing about the future. If the same certificate is used repeatedly, the online check has to be repeated, as well.

To facilitate the decision of whether or not a certificate is valid at a particular instance of time, all the different validity conditions end up being converted to validity periods as specified above. So, validating a certificate is relatively straightforward: check that the validity period stated in the certificate, as well as the online checks (converted to validity periods), are all valid at the time of use, and the certificate as a whole is then valid, and, therefore, grants the included permission.

3.2. Online checks

All the online checks are defined using the following format:

```
<online-test>::"(" "online"
    <online-type> <uris> <principal>
    <s-part>* ")" ;
```

where `<online-type>` can be `crl`, `reval`, `one-time` or `limit`. The `<uris>` specify one or more URIs (Uniform Resource Identifier [6]) that can be used to request revalidation; e.g. in the case of `crl`, the URI points to the crl file. `<principal>` specifies the public key used for verifying the signature on the online reply. The `<s-part>` is optional, and may contain parameters to be used in the online check.

Next, we'll go over the individual methods and their reply formats.

3.3. CRL

CRL (Certificate Revocation List) is based on the idea that a certificate is valid unless it appears on the specified CRL. SPKI includes both traditional and delta CRLs in its specification. These must also be signed by the aforementioned `principal`. The CRL formats are specified below.

```
<crl>::"(" "crl" <version>?
    "(" "canceled" <hash>* ")"
    <valid-basic>")" ;

<delta-crl>::"(" "delta-crl" <ver-
    sion>? <hash-of-crl>
    "(" "canceled" <hash>* ")"
    <valid-basic> ")" ;
```

3.4. Reval

Reval is based on an opposite idea: the certificate is invalid unless the prover can provide a current ``bill of health", which testifies that the certificate can be considered valid for the stated period. [10] specifies the reply format:

```
<reval-reply>::"(" "reval"
    <version>? "(" "cert" <hash> ")"
    "invalid"? <valid-basic> ")" ;
```

The reply identifies the original certificate in the hash and gives a confirmed (in)validity period for that certificate. The reply must be signed with the key given as `<principal>` in the original certificate.

3.5. One-time

One-time is based on the idea that it is impossible for the issuer to predict anything about the future validity of a certificate and, therefore, the user has to check the validity with every use of the certificate. The certificate contains a URI to the server, and the reply is ``yes" or ``no" with a time period of ``now".

```
<one-time-reply>::"(" "one-time"
    <version>? "(" "cert" <hash> ")"
    "invalid"? "(" "one-time" <nonce>
    ")" ")" ;
```

Again, the hash must correspond to the original certificate, and the reply message must be signed by the principal given in the certificate.

3.6. Limit

Limit is meant to enable quotas, i.e. it can be used to limit the usage based on suitable properties, like the number or length of usage. It is otherwise similar to one-time except that the server will not reply to queries, unless the user is able to prove that she is authorised to use the resource in question by presenting a suitable certificate chain. The limit query sent to the online server is of the form:

```
<limit-query>::"(" "test" <version>?
    "limit" <cert> <request>? <chain>
    ")" ;
```

```
<request>:: "(" "request" <s-part>
    ")" ;

<chain>::"(" "chain" <cert>+ ")" ;
```

Above, <cert> is the certificate whose online test(s) are to be made, <request> specifies the amount of resources requested, and <chain> proves that the verifier is entitled to ask about the validity of the certificate. The last certificate of the chain must be the validation certificate, which contains the <nonce> that is to be included in the reply to the query.

```
<limit-reply>:: "(" "limit"
    <version>? "(" "cert" <hash> ")"
    "invalid"? "(" "one-time" <nonce>
    ")" <context> ")" ;

<context>:: "(" "context" <hash> ")"
    ;
```

where <hash> is a hash of the concatenation of the canonical forms of <request> and <chain>.

3.7. Renew

Renew offers an alternative approach to revocation. Instead of issuing long-lived certificates and then worrying about their validity, we issue a string of short-lived certificates, which together cover the lifetime of a long-lived certificate. This simplifies matters, as the short-lived certificates can often operate offline and the network connection is required only to automatically fetch the subsequent certificate.

If everything is in order, the reply contains the next certificate:

```
<renew-reply>:: "(" "renew" <ver-
    sion>?  <cert> ")" ;
```

If, however, the right has been cancelled, the reply is of the form:

```
<renew-reply>:: "(" "renew" <ver-
    sion>? "(" "cert" <hash> ")"
    <valid-basic>? ")" ;
```

Again, the hash must correspond to the original certificate and the validity period states a period of time during which renewal requests will be denied (i.e. the conceptual long-lived certificate is not valid during this period).

4. Related work

The majority of work done in the field of certificate revocation has so far concentrated on identity certificates, in particular X.509 identity certificates. There exist several RFCs and Internet drafts that deal with X.509 certificate management and validation [5][1][2][3][4][14][12]. As to revocation methods, most of them concentrate on the CRL concept, and on how to effectively use it, but lately the trend has been to introduce other methods including online methods.

As to research, the majority of work has concentrated on evaluating the efficiency of CRLs and implementing improved, yet similar solutions. Further, some different solutions have been proposed [13]. Some work has also concentrated on the risk models and on the evaluation of different mechanisms in light of these risks [15][11]. Unfortunately, compared to SPKI authorisation certificates, there are a few significant differences in the X.509 model, which prevent us from directly applying the same solutions:

- The number of certificate issuers. In X.509, the number of CAs that issue certificates is orders of magnitude smaller (in SPKI, every human, computer etc. can issue certificates). This makes CRLs, which aggregate revocation information, much more feasible.

- Risk model. In X.509, the issuer and verifier are normally separate entities. The risk is taken by the verifier, yet the revocation decisions are made by the issuer. In SPKI, the risk takers are also issuing the certificates and can therefore control the revocation decisions to balance the risk.

These issues have been discussed in more detail in [10]

5. Choosing the validation and revocation methods

The phases of access control were presented in Figure 1. In [10] we have discussed the revocation problems at the time the certificates are used (phase 2 in Figure 1). These include the problems of authenticating the participants and providing undeniable evidence, also for liability reasons. In this paper, we focus on phases 1, 3 and 4. In phase 1, the essential problems include choosing the right validation methods, choosing the servers to implement them, informing the servers about the validity rules, and possibly paying the server's owner, if the servers are operated by a third party. In phase 4, the

Table 1: A summary of the online methods

Method	Typical use	Processing overhead	Revocation speed
Limit	Quota	High	Immediate
One-time	Limit usage on non-user specific factors	Moderate	Immediate
Reval	Revocation	Low	After current reval validity period
CRL	Revocation	Low	After current crl validity period
Renew	Revocation	Low	After current certificate expires

problems include things like informing the server about the revocation decision and providing undeniable proof, again for liability reasons.

5.1. Validity period

Phase 4 is simply a mechanism for making sure that certificates do not remain valid indefinitely, but instead automatically expire after a reasonable time. As a rule, it is a good practice to always include an expiration date in a certificate (only in very rare situation are there good reasons to make it a permanent certificate). In most of the cases, the matters themselves tend to change over time, so it makes sense to periodically reissue the certificates, if the rights are still required. Otherwise, the issuer is stuck with a growing number of certificates, which cannot be purged from the systems, as they are still officially valid.

5.2. Choosing the online method

This section discusses some of the main criteria in choosing the most suitable revocation method for a particular situation. Most, if not all, of these choices should be made by the designer of the system - they should not be left to the end users. [9] provides further examples of cases for each method and how they affect the end user. The results of this discussion have been summarised in Table 1.

An authorisation certificate is essentially a ticket granting the specified right to the indicated recipient. Now, the certificate is always valid unless its validity is somehow limited by listing conditions in the validity field of the certificate. Once the certificate has been issued, there is no practical method of getting it back from, say, a misbehaving user. The only recourse the issuer has is to include some limiting conditions in the validity field when the certificate is created. Here lies the difficulty: all possible future problems have to be anticipated and suitable countermeasures devised at the creation time. This is almost a mission impossible, because delegation will take place - the final user cannot be known until the time the certificate is used.

The choice of the most suitable validation/revocation method depends on what we want to achieve with it. We typically have two different goals: to control the amount of usage either discriminately (limit) or non-discriminately (one-time), or just to enable the revocation of the right in case the circumstances change, there is misuse of the right, etc. With the proposed changes to SPKI, any of the online methods can be used for the latter.

In the latter use, one important aspect is how fast we want our revocation command to take effect. CRLs and reval are both issued with a validity period, which is then the worst case time the issuer has to wait for her command to take effect. On the other hand, making the period very short does have performance implications, as the users are then forced to be online more often and fetch the latest validity statement. The issuer can naturally vary the validity period depending on the rate of problems or some other factor, but essentially both methods are best suited for situations, where the validity period does not have to be very short. This is particularly true about CRLs, where the validity period has to be the same for all certificates on the list, thus making it less practical to shorten the period if one of the certificates is showing signs of misuse. Processing overhead for the online server is fairly low with both of these methods, as the same reply can be used throughout the validity period.

On the other hand, a typical end user, e.g. someone using a certificate-based credit card, is less interested in the performance problems and more interested in the system behaving in an intuitive manner: when the parent presses the button to revoke the child's credit card, the revocation should take effect immediately, not after some arbitrary time. Even if security-wise this time might not be that important, compared to the time it might have taken for the parent to realise that security has been breached and that the certificate should be revoked, the delay is still a source of anxiety to the parent and should whence be minimised. For that reason, a method like CRL or reval is not good: they sacrifice the sense of control for the benefit of reduced overhead.

The only additional advantage they offer is support for offline operation, which is not necessary in all situations. On the other hand, the delay does not have to matter to the end user – the possible misuse and its costs can be included in the business model of the system, similarly to the existing credit card systems [9].

One-time is more suitable in a situation where we essentially want our revocation decision to take effect immediately or at least with a very short delay. On the other hand, we pay a price in performance for this convenience – every instance of usage requires network connection, as well as an individual reply from the server. So, if the certificate is used very often and performance really becomes a problem, we might consider using a lighter method and taking care of the misuse with the business model as mentioned above.

The other use for one-time, namely, controlling the amount of use, is another matter. In this case, we consider the certificate to be a recommendation, but the actual right depends on the circumstances, like the time of day or current load on the system. In this case, we are most likely more than willing to accept the performance penalty in exchange for the additional functionality.

Finally, limit is most likely used for controlling a quota; the possibility of revocation is just a fringe benefit. In this case, we pay an even higher price in performance, as its usage requires a two-phase negotiation with individual replies, but the new possibilities should more than outweigh that.

6. Background for the validity management protocol

In this section, we go over some of the key questions in designing the protocol.

6.1. Who can issue commands?

One of the basic things is naming the principal(s) that are allowed to issue revocation commands. The most obvious solution would be to state that the principal, who issued the certificate, is implicitly assumed to have the right to revoke it. However sometimes it would make sense to authorise others to revoke a particular certificate, for instance in a situation, where it is imperative that the certificate is revoked as fast as possible after a breach but the original issuer is not available to perform the revocation.

6.2. Requesting status information

The issuer might be interested in following how the certificate is used, particularly if it contains one-time or limit conditions, or if there are several individuals with the ability to revoke the certificate.

6.3. Auditability

The commands and their replies have to be auditable in case there is dispute as to the correct replies given by the server.

6.4. Support pre-evaluated answers and dynamic answers

In some cases, the answers are known in advance, e.g. when we revoke a certificate. In other situation, like with one-time and limit, we want to evaluate the answer at the time of usage.

7. SPKI Validity Management protocol

The protocol has been defined in XML and corresponding DTD can be found in appendix A. It defines the structure and contents of the messages between the issuer and online server. All messages are signed, which guarantees message integrity and authentication. Further, to protect against replay attacks and to guarantee confidentiality, a secure transport layer is used to carry the messages.

The protocol consists of just two messages: a command and a corresponding reply.

7.1. The Command

The command has the following structure:

```
Server_update cert, chain?,
    online_test_hash, de-
    lete_request*, test_definition*,
    status_query*, signature
```

`Cert` is the certificate, whose online condition is being managed. `Chain` is an optional field containing a list of certificates that proves that the current command issuer is authorised to send the command (this is required only if the command is sent by someone other than the certificate issuer). `Online_test_hash` identifies, which one of the possibly multiple validity conditions in the certificate is being managed.

The following three fields form the main part of the message. Even though they all are optional, at least one of them must be included in the command for it to be valid. The first, `delete_request`, defines which already defined rules are to be deleted. Each `delete_request` contains a validity period; all rules applying to that validity period are to be deleted.

The next part, `test_definition`, issues the new validity rules. There are two types of rules: pre-evaluated answer to be distributed at the specified time, and dynamic code that is to be evaluated by the server when a request is made. The pre-evaluated answer is further divided in three classes: a yes_no_answer is used for reval and crl, i.e. methods that reply with a validity period, Now_answer is used for one_time and new_cert_answer is used with renew. Limit always requires a dynamic_condition.

The final part, `status_query`, requests information on the validity status. It defines validity period for which we want the status information. Further, with the verbose flag the server is instructed to include in the reply the rule used to deduce the status.

The command ends with a `signature`.

7.2. The Reply

The reply follows a similar structure:

```
server_reply cert_hash,
    online_test_hash, delete_reply*,
    test_definition_reply*,
    status_reply*,service_status,
    signature
```

`Cert_hash` is a hash of the certificate in question. `Delete_reply` and `test_definition_reply` contain status codes about the success of the corresponding commands. Finally, `status_reply` contains status information for the requested periods and optionally the rules for deducing those.

8. Conclusions

In this paper, we have discussed the problems of managing the online validation and revocation of SPKI authorisation certificates. Due to their nature, authorisation certificates are well suited for granting rights, but limiting or revoking them presents a bigger challenge.

All the existing solutions to these problems are based on online servers that give authoritative statements about the validity of a certificate. We have discussed the advantages and drawbacks of the various solutions. Finally, we have presented a protocol for managing the online servers.

9. References

[1] C. Adams, P. Sylvester, M. Zolotarev, R. Zuccherato: Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols. Request for Comments: 3029, February 2001.

[2] C. Adams, S. Farrell: Internet X.509 Public Key Infrastructure Certificate Management Protocols. Request for Comments: 2510, March 1999.

[3] C. Adams, S. Farrell: Internet X.509 Public Key Infrastructure Certificate Management Protocols. Internet Draft, December 2001.

[4] Ambarish Malpani, Russ Housley, Trevor Freeman: Simple Certificate Validation Protocol (SCVP). Internet Draft, March 2002.

[5] A. Aresenault, S. Turner: Internet X.509 Public Key Infrastructure: Roadmap. Internet Draft, January 2002.

[6] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform Resource Identi_ers (URI): Generic syntax. Request for Comments: 2396, August 1998.

[7] Carl M. Ellison, Bill Franz, Butler Lampson, Ronald L. Rivest, Brian M. Thomas, and Tatu Ylönen. Simple public key certificate. Internet draft (expired), IETF SPKI Working Group, March 1998.

[8] Carl M. Ellison, Bill Franz, Butler Lampson, Ronald L. Rivest, Brian M. Thomas, and Tatu Ylönen. SPKI certificate theory. Request for Comments: 2693, September 1999.

[9] Kristiina Karvonen, Yki Kortesniemi, Antti Latva-Koivisto. Evaluating Revocation Management in SPKI from a User's Point of View, Proceedings of Human Factors in Telecommunication 2001, November 2001, Bergen, Norway

[10] Yki Kortesniemi, Tero Hasu, Jonna Särs: A Revocation, Validation and Authentication Protocol for SPKI Based Delegation Systems, Proceedings of Network and Distributed System Security Symposium (NDSS 2000), 2-4 February 2000, San Diego, California

[11] Patric McDaniel and Aviel Rubin. A Response to "Can We Eliminate Certificate Revocation lists". In Proceedins on the Financial Cryptography '00. The International Financial Cryptography Association (IFCA)., February 2000.

[12] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. Request for Comments: 2560, June 1999.

[13] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. In Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas, January 1998. Usenix Association.

[14] Denis Pinkas, Russ Housley: Delegated Path Validation and Delegated Path Discovery Protocol Requirements (DPV&DPD-REQ). Internet Draft, April 2002.

[15] Ronald L. Rivest. Can we eliminate certificate revocation lists? In Proceedings of the Second International Conference on Financial Cryptography, Anguilla, British West Indies, February 1998.

Appendix A: The DTD of SPKI Validity Management Protocol

```
<!--
   DTD for a SPKI online test management messages.
-->
<!ELEMENT hash                  EMPTY>
<!ATTLIST hash          data CDATA #REQUIRED>
<!ELEMENT cert_hash     hash>
<!ELEMENT cert          EMPTY>
<!ATTLIST cert          data CDATA #REQUIRED>
<!ELEMENT chain                 (cert+)>
<!ELEMENT online_test_hash      hash>
<!ELEMENT reason                (#PCDATA)>
<!ELEMENT no            EMPTY>
<!ELEMENT notbefore     (#PCDATA)>
<!ELEMENT notafter      (#PCDATA)>
<!ELEMENT date                  (#PCDATA)>
<!ELEMENT valid                 (notbefore?, notafter?)>

<!ELEMENT yes_no_answer no?, valid>
<!ELEMENT now_answer  no?, valid>
<!ELEMENT new_cert_answer       cert, notbefore>
<!ELEMENT currently_in_use      EMPTY>
<!ELEMENT dynamic_condition     valid?>
<!ATTLIST dynamic_condition
                        type PCDATA #REQUIRED
                        data CDATA #REQUIRED>

<!ELEMENT crl_test      yes_no_answer | dynamic_condition>
<!ELEMENT reval_test    yes_no_answer | dynamic_condition>
<!ELEMENT one_time_test now_answer | dynamic_condition>
<!ELEMENT renew_test    new_cert_answer |
                        dynamic_condition>
<!ELEMENT limit_test    dynamic_condition>
<!ELEMENT limit_status  (#PCDATA)>
<!ELEMENT service_status        (#PCDATA)>

<!ELEMENT test_definition ( crl_test | reval_test | one_time_test |
    renew_test | limit_test)>
<!ELEMENT test_definition_reply   reason>

<!ELEMENT status_query verbose?, valid?>
<!ELEMENT status_reply  (yes_no_answer, currently_in_use?) |
    now_answer |
(new_cert_answer, currently_in_use?) | limit_status, dy-
    namic_condition?>

<!ELEMENT delete_request valid>
<!ELEMENT delete_reply  reason>

<!ELEMENT signature     EMPTY>
<!ATTLIST signature     data CDATA #REQUIRED>

<!ELEMENT server_update cert, chain?, online_test_hash, de-
    lete_request*, test_definition*, status_query*, signature>
<!ELEMENT server_reply cert_hash, online_test_hash, de-
    lete_reply*, test_definition_reply*,
    status_reply*,service_status, signature
```

Extended Validation Models in PKI: Alternatives and Implications

Marc Branchaud
RSA Security
Vancouver, BC, Canada
marcnarc@rsasecurity.com

John Linn
RSA Laboratories
Bedford, MA, USA
jlinn@rsasecurity.com

1. Introduction

The fundamental goal of PKIs is to provide a means for participating entities to establish and manage trust in other entities, either within or across domain boundaries. As PKIs have evolved, so has the set of alternate methods supporting validation of entities, their certificates, and their keys. Validation processing determines whether or not the acceptance of a certificate or key represents a suitable risk to a relying party. As such, it is a central and necessary basis to support reliance on PKI-based authentication.

Increasingly, PKI designers seek to distribute validation-related information and processing among cooperating components, reducing the complexity at individual relying parties. These techniques afford the potential for great power, but also imply fundamental shifts in the trust relationships among the entities involved within a PKI. In this paper, we examine traditional technology practice in the field, consider newly emerging alternatives and their characteristics, and look ahead to candidate future directions and their implications.

2. Existing PKI Practice

Early work in PKI definition culminated in Version 1 of CCITT Recommendation X.509 [CCIT88]. This work, initially instigated as a supporting mechanism to authenticate directory queries and responses, assumed the availability and use of a directory as a repository for certificates and Certificate Revocation Lists (CRLs). Use of certificates and CRLs, signed by responsible Certification Authorities (CAs), made it unnecessary for the directory to be strongly trusted for security purposes; no compromised directory could forge an entry that a verifier would accept. As a central premise, no CA or other security-relevant component operating as a CA's agent needed to be accessible on-line in order to support authentication with certificates once issued.

In the late 1980s and early 1990s, related work in the Internet community concerned usage of X.509 certificates to support e-mail protection, a project known as Privacy-Enhanced Mail (PEM) [Linn93] [Kent93]. PEM was designed to operate in environments common at that time, where communications facilities were often costly and confined to store-and-forward messaging. For these reasons, even fixed-site users often accessed their messages in an off-line mode. As a result of these factors, the PEM design placed a high priority on making messages self-contained for processing purposes. Facilities were designed so that certificate chains could be attached to messages and so that CRLs could be obtained via e-mail. Further, PEM contemplated a hierarchic certification model, within which it was straightforward for a sender to predetermine the path elements that would be acceptable to message recipients.

CRLs are the "traditional" and most widely standardized revocation facility for certificate-based infrastructures, and have an existing and growing installed base. Their evolution has continued as additional extensions have been defined, particularly to allow partitioning of CRL data across multiple objects in large-scale environments. They provide a means for propagating revocation information in a signed fashion, allowing its storage in directories and other stores that need not be trusted for security purposes. Nonetheless, CRLs have long been one of the most controversial components of PKI systems. Their disadvantages include limited timeliness, the need for widespread propagation of potentially large objects (within which most individual verifiers are interested in only a small portion of the content), and the fact that concise proofs of validity are not directly available for presentation along with certificates.

The IETF PKIX (Public Key Infrastructure using X.509) working group was established in 1995 and became the central forum for defining PKI facilities for use by Internet applications. In addition to profiling X.509 certificate and CRL constructs for Internet usage [Hous99], it also undertook the definition of the Online

Certificate Status Protocol (OCSP) [Myer99] as an alternative to CRLs, enabling responder servers to provide revocation information for certificates in the form of signed responses to per-certificate queries. CAs can explicitly delegate their revocation reporting to separate responders acting as their agents, representing this delegation with an extension in the certificates that they issue to the responders. Through this delegation, the CA private keys used for certificate signing can remain in off-line systems, though the responders' private keys must be accessible to sign responses to on-line queries.

The OCSP semantics were designed to enable migration from CRLs; OCSP responders can use CRLs as a source of revocation information or can be more directly coupled into CA databases. Like CRLs, OCSP responses are signed and can be saved along with validated messages to enable revalidation at a future point in time. OCSP queries determine whether one or more individual certificates have been revoked or are in suspension. It explicitly excludes validation of the certificate's signature, timeliness, or path-level validation, though extensibility for additional validation services is possible within the protocol framework. OCSP requestors must perform path-level processing, as the public key of a certificate issuer is required in order to construct a status query for a certificate. Variant and alternative protocols have been proposed with broader functionality, and will be discussed next.

3. Delegated Validation

The IETF PKIX working group is currently developing requirements and examining proposals for delegating certification path construction and validation from a relying party's client to a server. This work is at least partially motivated by the success of OCSP, which has seen wide adoption because it allows applications to offload the task of determining a certificate's status. Similarly, delegated path processing is seen as a way to offload the tricky and onerous work of discovering and validating certification paths.

PKIX is exploring two aspects of server-side path processing. In the first, Delegated Path Discovery (DPD), a server is given a target end-entity certificate and one or more trust anchors. The server then sets about discovering candidate paths between the trust anchor(s) and the target certificate. These certification paths are returned to the client, which will then set about validating the chains using the regular X.509/PKIX rules, including checking the status of each certificate, using CRLs or OCSP. In this model, the client need not trust the DPD server, as the client performs all cryptographic, status and path validation itself.

The second aspect, Delegated Path Validation (DPV), is more interesting from a trust management perspective. With DPV, the client expects the server to not only discover candidate certification paths but to also completely validate and status-check these paths. The client is essentially looking for a Boolean response as to whether or not it can accept the target end-entity certificate. The client must completely trust its DPV server, as it abdicates to the server all responsibility for determining acceptable paths. Even where the client provides the server with its own policy inputs or trust anchors, such data are primarily advisory as the client can neither control nor examine the server's internal processing.

It is this complete delegation of responsibility that fundamentally distinguishes DPV from DPD and OCSP. The delegation model of the latter two protocols expects their clients to intelligently participate in the PKI, and so their servers can only return copies of CA-sourced information. OCSP and DPD servers must operate within the confines of that information, and are essentially mechanistic publishing and data-gathering tools. The clients will verify that their results come from an authoritative source. On the other hand, DPV servers are free to draw on other sources of information in order to synthesize appropriate responses. DPV servers can act with much more autonomy, as their clients have no way to verify the "correctness" of their behavior.

A number of advantages are gained when clients are willing to abdicate their responsibility and control to follow the DPV model. Not the least of these is a radical simplification of the client applications. Clients can do away with path validation and discovery code, which may include support for multiple protocols and algorithms. In the limit, clients need not even be able to parse a certificate's contents in order to use some DPV services.

Another advantage of the DPV delegation model is that it provides domain administrators with a convenient central point to manage inter-organizational trust on behalf of the domain's relying parties. Trust relationships can be switched off and on for all or part of a domain according to whatever criteria the domain's policies dictate, and without the need for any reconfiguration or even notification of the domain's clients. This property alone may motivate enough organizations to deploy DPV, enabling rapid, widespread adoption.

It is necessary to recognize, however, that reliance on DPV implies a dependency on on-line service availability in order to perform validation. Further, aggregation of queries within a DPV server may present a convenient point for privacy-relevant data about client requests to be collected. Nonetheless, DPV's operational

characteristics relative to off-line validation are motivating growing interest, and broad adoption appears likely.

The DPV delegation model turns each DPV server into a trust anchor for its clients. This distributes the trust anchors throughout the PKI, which in turn reduces the number of clients that must be reconfigured following the compromise of any particular trust anchor. This effect can also be achieved without DPV, through the deployment of general CA hierarchies where (as opposed to top-down hierarchies), each CA certifies its parent as well as its children. In a general hierarchy any CA can act as a trust anchor that provides connectivity with the entire PKI, and the compromise of any CA will only require reconfiguration of its trusting clients as well as its parent and child CAs. We note, however, that such general CA hierarchies have proven to be rare in practice, whereas the DPV model expresses this property naturally.

Two candidate protocols have so far emerged to support DPV-style services. These are the Simple Certificate Validation Protocol (SCVP) [Malp01] and the XML Key Management Specification (XKMS) [W3C01].[1] XKMS in fact provides more than just delegated validation services; however, this section will focus on the DPV-style services defined in "Tier 2" of the XKMS XML Key Information Service Specification (X-KISS). We note in passing that extensions to OCSP to support a DPV service have been proposed. As these are not materially different from the services proposed in SCVP, we do not include the OCSP extensions in our analysis.

SCVP and Tier 2 of XKMS (hereafter referred to simply as XKMS) have significant syntactical differences, both in terms of each protocol's messages and in terms of how a client specifies the "certificate" to be validated. SCVP is an ASN.1 protocol that exclusively supports X.509 certificates, while XKMS is an XML protocol that supports X.509, PGP and SPKI certificates. XKMS can also identify keys by reference (URI) or even a simple name. (The relative merits of ASN.1 or XML are beyond the scope of this paper, and are irrelevant to delegated validation.)

Fundamentally, both protocols enable a client to delegate validation operations to a server. Each allows the client to request various types of assertions from the server regarding the certificate in the query. SCVP's focus on X.509 limits these to certification paths and assertions of certificate revocation status, while XKMS allows clients to also request raw public key values (on the assumption that the server has validated the keys according to the appropriate criteria). Both protocols allow the client to also request various forms of evidence to support the assertion in the response, including certification paths and revocation status information.

In delegating validation operations, the client in either protocol trusts that the server will act appropriately on its behalf. Neither protocol explicitly defines server behavior, although both imply that the server should perform the operations that a non-DPV client would in order to validate the certificate (e.g. perform PKIX-style certificate path discovery and validation when queried about an X.509 certificate). As with client-based validation, extensions such as nameConstraints can be incorporated in cross-certified paths in order to limit the transitivity of trust. SCVP allows the client to provide some input into the server's processing, such as policy identifiers, trusted CAs and certificate revocation information. SCVP clients can also specify a "configuration identifier" to, for example, inform the server of the context of the client's query.

XKMS does not allow its client to provide similar information. This omission may not be as glaring as it appears, for we note that in a delegated validation system clients will not themselves validate the evidence included in a response (though they will authenticate the response itself). Thus the server is free to ignore the client's ancillary inputs, or it can tailor the evidence in its response to match those inputs, and the client is not in any position to detect such manipulation. (If the client were able or willing to verify the evidence itself, it would not really need a delegated *validation* service.) This is what we mean when we say that a DPV client must trust its server. Any ancillary input in a request can be ignored, and any evidence in a response is mainly useful to third parties auditing the response. XKMS, as it lacks facilities for clients to provide ancillary input, merely makes these facts more explicit.

This need to completely trust DPV servers does not mean that delegated validation cannot be a good and useful technology. It does, however, change the nature of PKIs in ways that will be explored in the rest of this paper. For now, we close this section with the notion that some ancillary input can still play an important role in a DPV system. Although information such as trusted CAs and revocation information might help a server, we expect that servers will typically have their own resources for obtaining such information, and will be configured with the trusted authorities of the clients they serve. What will be truly useful is an indication of the client context. This would allow the server to perform different levels of validation depending on whether, for example, the client is merely reading e-

[1] The specifications for both SCVP and XKMS are still in draft form and subject to change. The analysis presented here considers the protocols as specified in January 2002.

mail or is performing an expensive purchase. Such a context indicator could in fact instruct the server to employ different trusted authorities and/or different path discovery and processing rules. All this could be triggered by a single identifier communicated from the client to the server, the values of which could be standardized or determined by private agreement.

4. Recursive and Chained Validation

Current standardization initiatives [Pink01] emphasize use of DPV between clients and their associated DPV servers, but it is possible to envision extending the paradigm so that DPV servers themselves consult other DPV servers to perform validation. A generalized PKI structure could incorporate four tiers of elements generating or processing validation data: issuing CAs, their OCSP responders, DPV servers trusted by relying parties (RPs), and the RPs themselves. If and as use of DPV becomes common, DPV-based queries might even be extended back to DPV services associated with CAs. In this fashion, DPV could be applied as a means to publish CA status information, eventually eliminating the OCSP tier. Potentially, use of CA-provided DPV could make non-DPV status checking facilities moot and could reduce the number of protocols required within an overall PKI. It remains likely, however, for DPV to remain under RPs' jurisdiction, querying OCSP responders maintained on behalf of issuing CAs. Independent of whether CAs export their status information via CRLs, OCSP, or DPV, it is possible for multiple layers of DPV interaction to be interposed between an RP and the information that an issuing CA provides for one of its certificates. This section discusses alternatives and issues arising in such *inter-server delegated* models.

When large-scale PKIs combine OCSP and DPV, trust may be delegated in two directions: from CAs, via their OCSP responders, and from relying parties, via hierarchies of DPV servers, each invoking the other's services. Borrowing the metaphor of a weather chart, a path extending from a relying party to a remote issuer CA can be thought of as crossing a "trust front", delimiting zones of responsibility affiliated with the relying party from zones affiliated with the issuer. In the general case, path validation requires information provided both from the issuing CA domain (i.e., certificates and their associated revocation information) and from the RP domain (to reflect its trust relationships and policies). When delegations to responders or validation servers extend from either domain, the delegating domain must trust its delegates to represent its interests, accurately process data obtained from other domains, and reflect the data that it's authoritative to provide.

In a DPV environment, each RP will normally focus its validation requests on a single trusted DPV server (although it may be replicated to ensure availability), which will be responsible for validating any and all certificates that its RPs receive. A given DPV server may select to delegate among a set of other DPV servers, depending on the particular certificate being queried and on the policies under which validation is being performed. If different clients use different DPV server paths to validate a particular certificate, it is possible that the information they receive via different sources (and their associated paths) may yield different results. This prospect arises whenever an RP obtains status information indirectly via active intermediaries, rather than by accessing the information directly from its source.

We distinguish three forms of interaction that can take place among delegated validation servers:

- *Chained* queries, within which a specific server is recognized as authoritative to respond to a query about a particular certificate. All queries about that certificate that are received by other servers are forwarded to the authoritative server, and the information obtained by a requesting server can remain in a form that is traceable to the authoritative domain.
- *Referred* queries, which resemble chained queries except that the requesting server is redirected to the authoritative server rather than having the query mediated on the requestor's behalf by the server that initially received it. As with chained queries, the information obtained by a requesting server can remain in a form that is traceable to the authoritative domain.
- *Recursive* queries, where each server aggregates information obtained from other servers in order to respond to the queries it receives, but where a requestor must directly and fully trust the delegated validation server that it contacts as the provider for information about a set of domains. In this form of interaction, intermediaries distill the information issued by authoritative sources and actively integrate it with data maintained in their own databases, rather than transferring and processing it in a fashion that keeps it independently verifiable by relying parties.

Different DPV servers may employ different query models, satisfying different goals and constraints; further, a given DPV server may employ different models depending on the domain associated with a particular certificate that is to be validated. Of the models, recur-

sive queries imply the greatest level of trust delegation from the requesting server to the target, as they move the critical operations of aggregation and synthesis of validation data to the target. This reduces the amount of data that needs to be propagated to requestors, typically making protocol exchanges simpler and more compact. Analogous to end-entity client use of DPV, recursive inter-server DPV interactions imply that requesting DPV servers place fundamental trust in the target DPV servers to which they direct their queries. In a recursive DPV environment, multiple DPV services, operated by different domains, may be involved in determining the validity of a certificate. The domains responsible for operating DPV services may not directly correspond to the domains responsible for individual certification path elements, though validation policies could be applied to enforce such a constraint.

Referred and chained queries, in contrast, collect validation data for integration and processing by the requesting entity, whether an RP or a DPV server; as such, they transfer a larger volume of information for requestor processing. These approaches can enable independent auditability, can contribute to post-facto revalidation for non-repudiation purposes, and can help DPV servers to partition the impact of compromised data originating from particular sources. If signatures are applied and retained on the received data, requestors can preserve records of the validation data they obtain in a form that is provably traceable to authoritative issuers. It appears unlikely that many RPs will commonly revalidate determinations made by their DPV servers, unless on an exception or post-facto basis, as RPs prepared to perform this processing would gain relatively little benefit from using DPV. They would need to obtain (either through DPV itself or via out-of-band means) trusted keys with which to verify the signatures of remote servers, and to validate that those servers were authoritative to report status on behalf of specific domains. Even within a model where individual RPs elect to delegate trust to DPV servers, it may still be desired for those DPV servers to maintain validation data records on behalf of their domains.

One fundamental design choice for PKI designs and deployments is as follows:

- Is the set of certificates that a relying party can validate intentionally confined to those domains with which it (and/or its DPV server) has established direct working relationships; or
- Is broader validation sought as a goal?

If the scope of certificates to be validated can be constrained in advance, directly established relationships between individual DPV servers may suffice to support the RP community of interest. If universal validation is desired, however, DPV servers must be able to take recourse to a general certification hierarchy or mesh as a means to obtain trust connectivity among one another.

Recursive DPV distributes knowledge of suitable paths and sources for validation data among the set of DPV servers rather than collecting it within each domain that requests that certificates be validated. In a recursive model, individual entities need not perform end-to-end path discovery. This decentralization may prove valuable in enabling interconnected PKIs to scale to very large sizes. The distributed nature of recursive DPV echoes and accentuates a basic DPV characteristic; not only are its RPs unable to independently revalidate processing performed within their own DPV servers, they also lack independent assurance of the quality of information on which their servers depend. Overall, recursive DPV offers significant power and flexibility, but also implies significant growth in the set of on-line components comprising a distributed trusted computing base for certificate validation. Chained and referred DPV models distribute more data and processing complexity among participating components, but allow the participants to operate with a higher level of mutual suspicion among one another.

5. Implications and Future Directions

There are profound implications that arise when using inter-server delegated validation models such as those described in the previous section. Ostensibly, DPV services perform two distinct functions: certification path discovery and verification, and certificate status confirmation. The latter is necessary because of the nature of certificates. Specifically, a certificate conveys a binding that its CA believed to be true when the certificate was issued. However, typical certificate users often need to know if the CA still believes the binding to be appropriate when the certificate is used, rather then it was issued. Hence the desire to check a certificate's status.

An obvious effect of inter-server delegated DPV is that it eliminates the need for CRLs. This is because such a system constructs paths between domains online, through the involvement of each intermediate domain's servers. Since each domain is making an active assertion about which pathways are valid, and each includes certificate status as one of its validity criteria, separate mechanisms for obtaining certificate status are no longer necessary.

Active domain participation can be leveraged even further. For example, it can change the way that inter-domain trust is established and managed. This is currently achieved with cross-certification, but if a domain

is going to make an active statement about the status of its cross-certificates, it may as well instead make a statement about the relationships the cross-certificates represent. With active participation, domains need no longer rely on cross-certificates as a source of policy information, but can instead manage this data locally within their DPV servers. This allows for much finer control of the inter-domain relationship. At the coarsest level, the relationship can be present for some clients but not for others, or only at particular times. A more sophisticated application would enable the relationship under certain contexts, but not others. This would be the equivalent of having multiple cross-certificates between two domains that are issued under different policies, something that is theoretically possible but has yet to be seen in practice. The central administration benefits of DPV servers make practical the application of multiple policies to a relationship with another domain.

The effects of active domain participation can be felt even further. Consider that in a fully delegated DPV environment queries about a particular certificate eventually reach the issuer of that certificate, or an entity designated by that issuer to speak authoritatively on its behalf. This is necessary to obtain the status of the certificate. However, if the issuer of the certificate (or its designate) is making an active statement about the certificate's status, it could just as easily take advantage of the situation to make statements about the certificate's actual contents. In the limit, the certificate's issuer could, in response to a query, return certificate contents as separate elements rather than in the form of a certificate. So, for example, if the name of the certificate's subject has changed since the certificate was issued, the issuer could return the new name in its DPV response. The issuer could also return the subject's current public key, which would altogether eliminate the need to publish revocation information.

Taking this notion to its extreme conclusion leads us to consider the elimination of certificates entirely. Rather than obtaining a certificate, entities enrolling in a PKI could register their public key with an authority, which would give them an identifier of some sort. This identifier could then be presented, as is possible in XKMS, instead of a certificate in any challenge-response, key establishment, or digital signature verification protocol. The receiving party would submit the identifier to its local DPV server, which would resolve it (by eventually querying the authority that issued the identifier) into the registered public key that could be used to complete the protocol.

It is the online, active nature of inter-server delegated DPV that makes this possible. The ultimate scenario described above may or may not be practical, or even desirable, but it does highlight the fundamental shift that DPV services create in PKI. The original framers of X.509 did not contemplate domains making active statements about their certificates. Indeed, any online components (i.e. the X.500 directory) were specifically not trusted. DPV heralds a departure from that philosophy, and it will have profound effects on the very infrastructure itself.

6. Conclusions

On-line validation methods for PKI certificates are attracting increasing interest and adoption. Current standards directions emphasize the simplification of client-side processing. Two aspects are fundamental:

- Reduced volume of validation support data propagated to clients; and
- Reduced complexity of validation processing within clients.

In order to satisfy these goals, clients must delegate their trust to new services, relying on those services to perform validation for them. Rather than being eliminated, validation complexity will move to a new set of distributed components, operating as active intermediaries. When this delegation is performed, many "traditional" PKI assumptions will no longer hold, as new trusted points become active peer participants within distributed PKI-based architectures.

All on-line validation strategies can improve upon CRLs' schedule-driven revocation responsiveness, if their underlying information sources permit; DPV introduces the possibility of a different sort of latency as data is aggregated at intermediaries. Approaches differ significantly, however, in the scope of active components that must be trusted for validation purposes, and in the trusted properties that they must provide. As adoption of delegated validation proceeds, facilities to constrain these trust characteristics, preserving appropriate levels of mutual suspicion, are likely to be important.

Finally, delegated validation services represent a fundamental change in the assumptions that underlie a PKI. Most significantly, domain authorities can become active participants in the PKI, interacting dynamically with relying parties rather than merely making assertions at particular points in time. This has profound effects on the nature of PKI technology, leading to questions about the explicit need for revocation, and even about the nature of certification itself.

Acknowledgments

The authors thank the workshop's anonymous reviewers and Russ Housley of RSA Laboratories for their comments on this paper.

References

[CCIT88] CCITT Recommendation X.509 (1988), "The Directory - Authentication Framework".

[Hous99] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and CRL Profile", Internet RFC-2459, January 1999, http://www.ietf.org/rfc/rfc2459.txt.

[Kent93] S. Kent, "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", Internet RFC-1422, February 1993, http://www.ietf.org/rfc/rfc1422.txt.

[Linn93] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", Internet RFC-1421, February 1993, http://www.ietf.org/rfc/rfc1421.txt.

[Malp01] A. Malpani, P. Hoffman, R. Housley, T. Freeman, "Simple Certificate Validation Protocol (SCVP)", Internet draft work in progress, IETF PKIX working group, July 2001.

[Myer99] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Public Key Infrastructure: Online Certificate Status Protocol – OCSP", Internet RFC-2560, June 1999, http://www.ietf.org/rfc/rfc2560.txt.

[Pink01] D. Pinkas, "Delegated Path Validation and Delegated Path Discovery Protocol Requirements", Internet draft work in progress, IETF PKIX working group, November 2001.

[W3C01] P. Hallam-Baker, editor, "XML Key Management Specification (XKMS)", W3C Note, March 2001, http://www.w3.org/TR/xkms/.

Trust Assertion XML Infrastructure

Phillip Hallam-Baker
VeriSign Inc.

Abstract

The Trust Assertion XML Infrastructure (TAXI) is described. TAXI is a PKI research project that had the objective of developing technology that would assist the deployment of PKI. Parts of the TAXI architecture have since been realized in open standards, notably the XKMS [XKMS] and SAML [SAML] specifications, other parts of the TAXI architecture such as XTAML [XTAML] and XKASS [XKASS] have been published as research notes for public review and possible standardization at a later date. The paper describes the architectural principles underlying the design decisions taken in these specifications.

1 Cryptography and Trust

Public Key cryptography permits secure communication to be established between any parties provided only that each has trustworthy knowledge of the public key of the other. The means by which that trustworthy knowledge is obtained is known as Public Key Infrastructure (PKI).

PKI secures the interface between the abstract world of electronic communications and the concrete offline world. PKI is complex and subtle because the world is complex and subtle.

The deployment of PKI in the real world has been subject to numerous disputes about architecture, factional schisms and political intrigues. While some of these disputes have technical merit few have advanced the cause for PKI. The quest for the perfect PKI has too often been the enemy of deployment of a good PKI.

This paper describes the Trust Assertion XML Infrastructure (TAXI), a research project that was undertaken in the summer of 2000 with the objective of developing technology that would assist the deployment of PKI. Parts of the TAXI architecture have since been realized in open standards, notably the XKMS [XKMS] and SAML [SAML] specifications, other parts of the TAXI architecture such as XTAML [XTAML] and XKASS [XKASS] have been published as research notes for public review and possible standardization at a later date.

Standards documents intended to describe a normative specification should not provide any discussion of the architectural principles. This paper is intended to make good this omission and to explain how the different components of the TAXI architecture were intended to fit together. In view of the developments since the original TAXI architecture was developed this paper makes use of the terminology and concepts used in the XKMS and SAML specifications rather than those of the original documents.

1.1 Certificates

The traditional model of Public Key Infrastructure is based on the model proposed by Lauren Kohnfelder's in 1978 [Kohn78]. An email user A may obtain the public key of email user B by consulting a directory. The need for online access to the directory could be avoided by signing individual directory entries to form a 'certificate'.

The PKI most closely associated with certificates is X.509 [X.509], which realizes the Kohnfelder model in the context of the X.500 directory [X.500]. The influence of the Kohnfelder model is also seen in PKI proposals that attempt to escape from the certificate model including PGP [PGP], SPKI [SPKI] and even DNSSEC [DNSSEC]. All share the basic principle of using signed data to bind the public key of a user to a sign that identifies them. Regardless of whether the signed data is called a 'certificate', a 'key signing' or a 'signed record', the differences in how the signed data is generated and used are considerably less important than the similarities.

The X.509 specification was originally developed as a part of the OSI network standard developed as a joint standard of ISO/IEC and the ITU. As increasing use was made of the X.509 standard by Internet protocols an IETF working group was formed to describe the use of X.509 in

that context. Over time the IETF Public Key Infrastructure X.509 (PKIX) [PKIX] group has specified additional protocols that extend the use of X.509 so that the terms X.509 and PKIX are often used interchangeably.

1.1.1 Trust Topology and Names

For many years the PKI debate centered on the topology of trust. Certificate hierarchies, heterarchies and Webs of Trust were advanced each with merits and demerits. In the authors view this debate obscured rather than clarified the issues that should have been at the center of the debate, namely:

- The ability of relying parties to locate a public key for a particular purpose
- The ability of relying parties to locate a trust path that validates a public key
- The ability of relying parties to control the trust criteria that are applied

A highly constrained trust topology such as a strict hierarchy makes the process of locating keys and trust paths easier to implement and manage than a less constrained topology. An unconstrained topology in which all participants are peers appeals to the spirit of egalitarianism by obviating the need (but not precluding the existence) of centralized control.

While the trust topology debate has continued, real world deployment of PKI has largely converged on a single model in which multiple trust providers issue certificates and relying parties decide which trust providers to rely on.

1.1.2 Naming

A certificate binds a public key to a name; the question of naming has thus been at the center of many PKI debates. In the DNSSEC and the original X.509 architecture the certificate hierarchy precisely matches that of a hierarchical naming scheme.

A name is a signifier that bears only a conventional relationship to the signified [Sebok]. It follows therefore that if the trust providers are to be true peers they must have equal capacity to define naming conventions. This principle is embodied in the Rivest and Lampson SDSI paper [SDSI] that introduces a naming scheme in which all names are relative and "Alice" becomes "The person who Doug calls Alice".

This relativist naming scheme proposed appears unlikely to provide much value in practice. While all names are ultimately subjective the ability to communicate depends on the parties having established a vocabulary of shared terms. While names are defined in many ways and there is ultimately no single authority that is responsible for assigning names there is in practice little ambiguity. Names are chosen to facilitate communication. If Bob, Doug and Carol are in regular communication and each use the name 'Alice' to refer to a different individual a means of resolving the ambiguity will be found. It is more likely that the convention chosen will involve a property of the people called Alice that distinguishes them from each other than the speaker.

1.1.3 Beyond Email

The certificate-based model of PKI was developed to address the problem of sending secure email messages within the specific constraints of the early ARPANET. The X.509 specification was originally designed to support secure email in the context of the X.500 directory and X.400 mail. X.509 has since been extended to meet many requirements that were originally out of scope. In the process the X.509/PKIX specifications have grown larger and more complex.

Despite their complexity, the X.509/PKIX specifications are in several ways incomplete. In a commercial environment it is far more likely that Alice would issue a check in error than lose her safe key. PKIX provides no fewer than four methods of determining the validity status of a certificate. No mechanism is provided to determine the validity status of a signed document.

1.2 Client Complexity

One of the principal objections made to the deployment of traditional PKI is the complexity of the specification. Full support for the industry standard X.509/PKIX specification requires a very large and complex client implementation

that very few applications support directly (figure 1).

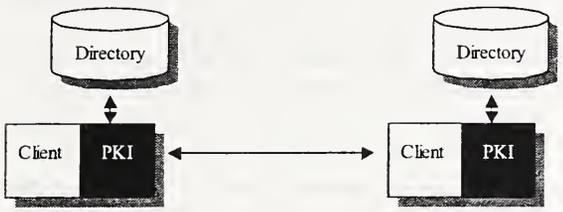


Figure 1 Client Complexity in Traditional PKI is High.

While PKI is ubiquitously supported in mainstream email, browser and operating systems software, 'sophisticated' PKI features such as cross-certification, OCSP etc. are not. Such features are typically only supported by PKI 'plug-ins' provided by third party PKI vendors. Plug-ins of this type have proved expensive to deploy and maintain, particularly since each PKI client must be configured with the location of the local PKI repository. A new plug in deployment is required each time there is a change to the PKI configuration, support for new PKI features is required or the base application is upgraded.

1.2.1 Historical Complexity and Necessary Complexity

Part of the complexity of PKIX is due to the process by which the specification developed. The CRL specification was developed as a certificate blacklist mechanism. As the number of certificates grew, CRLs grew to unacceptable size leading to various extensions to mitigate the problem. At the same time the Online Certificate Status Protocol (OCSP) was developed to provide real-time reporting of certificate status. Despite the close relationship between CRLs and OCSP the data formats and protocols associated with each are separate.

Although much of the complexity of PKIX could be reduced through a thorough re-design process, the main reason that the PKIX specification is complex is that it attempts to address a complex problem. In many instances it has been the attempt to address a complex problem with a too-simple solution that has led to complexity.

PKI is complex because trust relations in the real world are complex and cannot necessarily be reduced to a series of standardized machine-readable data formats. The choice for a PKI architect therefore is not whether there is complexity but how it is managed and where it is placed.

1.2.2 Directory as Certificate Repository

The close relationship between the X.500 and X.509 specifications led many to assume that digital certificates 'should' be stored an X.500 or LDAP directory. This assumption leads to the conclusion that the deployment of a PKI at either a local or global level is dependent on the deployment of a directory.

While many companies have deployed local directories these are almost without exception considered internal resources whose contents are company confidential.

While the X.500 or LDAP protocols might form a basis for a certificate retrieval protocol, the directory data model is not. The underlying principle of the directory data model is that the directory server supports a generic query mechanism to a hierarchical data structure. This model is ill suited to the needs of a certificate repository that is servicing highly specific queries against a heterachical PKI topology.

1.2.3 The Client Deployment Trap

The problem that appears to have brought deployment of new PKIX features to a halt is the client deployment trap. For a PKI feature to be useful every client must first support it. For mainstream application vendors to support a feature it must first be useful. None of the mainstream PKI enabled applications (Netscape Communicator, Microsoft Outlook, Lotus Notes) provide native support for cross-certification. The feature would have little value until it was widely supported and will not be supported in any degree until it provides value.

1.2.4 The End to End Principle

The *end-to-end principle* is one of the key architectural principles of the design of the Internet. Under the end-to-end principle the network core is as simple as possible, a packet switching network that provides no guarantees as to the reliability or order of packet delivery.

Sophistication is achieved at the ends of the communication where acknowledgement of received packets is made and packets are reassembled into order.

The end-to-end principle has been applied to security to establish the doctrine that security enhancements should be applied at the 'ends' of the communication. For example an email message should ideally be secured from the sender to the recipient.

The difficulty raised by this interpretation of the end-to-end principle is that the ends of the communication are devices while the ends of the trust relationship are people and/or organizations. The sophisticated management of trust relationships is complex and subtle and has proved to be beyond the level of complexity that developers of client applications will tolerate.

Properly understood, the end-to-end principle argues that complexity must be eliminated where possible and where it cannot be eliminated must be confined to those parts of the network infrastructure that are capable of supporting it.

1.2.5 Trust Management

The use of cryptography and PKI typically appeals to individuals of independent character. As a consequence PKI architectures have emphasized the role of individual choice in the configuration of their trust relationships. This approach is a poor match to enterprise needs where trust relationships between enterprises are by definition established at an enterprise level.

The PKI approach that requires PKI configuration to take place at the client end does not meet the needs of enterprises attempting to manage their trust relationships at the enterprise level. The client centric model of PKI requires that all trust relationships be expressible in a data format supported by the client and that the client support all the necessary location and retrieval protocols.

As the number of PKI enabled devices increases the trust management problem increases. Even highly motivated individuals managing their personal devices are unlikely to want to maintain their trust configuration separately on the laptop, desktop, handheld, mobile phone etc.

1.3 New Challenges

Despite the numerous objections made against it, the deployment of PKI has been a success by most ordinary measures. Millions of people use PKI each day, in most cases without being aware that they have been using it. PKI is already established that provides to anyone with a need secure email, a secure means to make payments over the Internet, a Virtual Private Network.

Although PKI has succeeded by most ordinary measures it has failed against its perceived potential. Security remains an optional extra used in cases of need, PKI enabled cryptography has not yet become the ubiquitous default.

This qualified success poses a considerable challenge to the deployment of alternative approaches. Attempts to replace X.509 completely have largely failed completely or been confined to a single narrow area of application. New PKI infrastructure can only be justified if it enables new applications of cryptography that were impossible or impractical with the existing infrastructure.

1.3.1 Constrained Devices

As the cost of processing power has decreased the number of devices with embedded CPUs has increased dramatically. Far from eliminating the constraints of CPU power on PKI, improvements in processor performance have increased them as manufacturers attempt to embed PKI into mobile phones, personal organizers and all manner of network devices.

In addition to lacking the processing capability to support sophisticated a sophisticated PKI client implementation, constrained devices often lack user interface capabilities that are appropriate to the task. The task of adding a root certificate into a PC web browser is supported by a rich user interface that presents the user with all the necessary information. While it is possible to add a root certificate into a mobile phone with a 20-button keypad and a 20-character display, it is unlikely that the process can be made acceptable to many consumers.

X.509 has been adapted to meet the constraints of wireless use in the WAP specification [WAP]. The modifications include the use of compressed 'WAP Certificates' and a messaging protocol

that uses a certificate identifier in place of the certificate itself to save bandwidth on constrained links.

1.3.2 Financial Transactions

Financial services applications operate under requirements that are quite distinct from the email application that has traditionally formed the PKI paradigm. Unlike email applications, financial services applications can depend upon the availability of network connectivity at all times. Financial services have a well-defined trust model that is backed by regulation, insurance and contracts that define the liabilities of the parties and operate in an environment in which the precise timing of operations can transfer liability from one party to another.

As a result of these different constraints financial services applications have traditionally been at the cutting edge of PKI, leading to developments such as the Online Certificate Status Protocol (OCSP) [OCSP].

The Identrus architecture [Identrus] applies PKIX and OCSP to provide real time validation of public keys in the context of the 'four-corners' model common to many financial transactions. This architecture demonstrates a significant limitation of the use of the certificate model designed to support offline messaging to an online application. Relying applications must support OCSP processing *in addition* to certificate processing, the PKIX architecture does not support the use of an online protocol *instead* of certificate processing.

1.3.3 Web Services

Web Services [SOAP] are a set of industry standards based on XML that allow applications running on different machines to exchange data. The goal of Web Services is to reduce or eliminate interface costs, the cost of exchanging data between computer systems. Interface costs represent two of the largest costs of running information systems:

- Computers generate messages that are sent to the customer by letter post or fax and entered manually into another computer system
- The largest cost in the deployment of a new software system is often interfacing the new system to the legacy systems already deployed.

Web Services offer the promise of enabling a new and more cost effective IT strategy in which communications that currently require human intervention are automated. Web Services have the potential to change the way that Enterprises communicate both internally and externally.

While X.509 certificate meet some of the PKI requirements of Web Services the use of an ASN.1 based PKI to support an XML based messaging infrastructure is unsatisfactory. While the overhead required to support ASN.1 and X.509 on a server platform is quite reasonable, the same overhead is unreasonable for many of the intended clients.

2 Trust Assertion XML Infrastructure

The TAXI architecture is based on the following principles:

- Minimize the complexity of client deployment, configuration and management.
- Separate the client implementation from the structure of the underlying PKI.

The TAXI architecture makes extensive use of the XML Signature [XML-SIG] `<KeyInfo>` element that allows a public key to be identified using practically any means including:

- The Public Key parameters (e.g. RSA modulus and exponent)
- Any naming scheme (e.g. URI, X.500 Common Name)
- X509 Certificate, CRL, OCSP token
- SPKI, PGP key signing.
- A URL for the retrieval of any of the above

2.1 Architecture

The TAXI architecture is divided into four tiers that represent increasing complexity from the first to the fourth as follows:

Tier 1 Location

The location service is a Web service that allows a client to locate information concerning a public key analogous to

the directory function in the PKIX model

Tier 2 Validation
The validation service is a Web service that allows a client to delegate both the retrieval and processing of public key information. The validation service is analogous to a highly extended form of the PKIX OCSP protocol.

Tier 3 Trust Assertion
A trust assertion contains a unique identifier, one or more statements, conditions and advice. Trust assertions combine the roles of PKIX certificates, attribute certificates and in some instances signed documents themselves.

Tier 4 Status Assertion
A Status Assertion is an assertion that makes a statement about the validity of one or more other assertions. Status Assertions combine the roles of CRLs and OCSP in the PKIX model.

2.2 Specifications

2.2.1 XKMS

The XKMS specification consists of a registration protocol and an enquiry protocol. These protocols may be used independently.

The XKMS enquiry protocol is the XML Key Information Service Specification (X-KISS) which supports two service tiers:

Tier 1: Locate
The client sends one `<KeyInfo>` element to the service and requests that the trust service provide a `<KeyInfo>` element that identifies the same key but is in a different format (e.g. X.509 certificate converted to key parameters).

Tier 2: Validate
The trust service validates the trustworthiness of the information returned according to service specific criteria.

2.2.1.1 Tier 1 Location

A client receives a signed XML document. The `<KeyInfo>` element in the signature specifies a retrieval method for an X.509 certificate. The client lacking the means to either resolve the URL or parse the X.509 certificate to obtain the public key parameters delegates these tasks to the trust service (Figure 2).

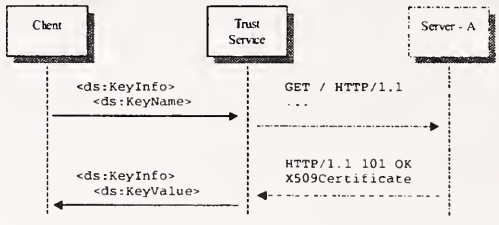


Figure 2: Key Location Service

2.2.1.2 Tier 2 Validation

The Validate service allows a client to delegate all trust processing functions to a trust service. As with the Locate service the client creates a query that specifies the information the validation service is to locate. Unlike the location service however the validation service is responsible for ensuring the trustworthiness of the data returned before relying upon it.

A client receives a signed XML document and queries the trust service to determine whether the signing key is trustworthy. In this case an X.509 certificate authenticates the signing key. The Trust Service builds a certificate trust path, then validates each certificate in the path against the relevant Certification Revocation List. The client is shielded from this complexity however and the trust service returns only the information of specific interest to the client; the key parameters, the data bound to the key and the validity of the binding (Figure 3).

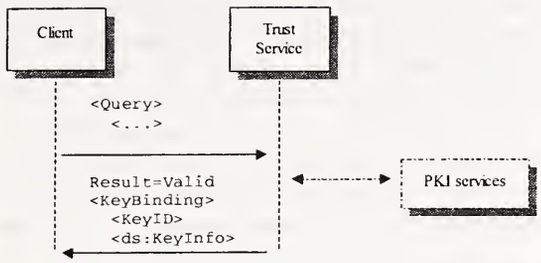


Figure 3 Key Validation Service

Delegation of trust processing functions to a trust service makes enterprise-wide control and oversight of PKI configuration possible. This is essential in Business-to-Business applications where the important trust relationships are between enterprises and not between individuals or the applications they use.

2.2.2 X-KRSS

XML Key Registration Service Specification (X-KRSS) defines a protocol for a trust service that accepts registration of public key information. Once registered, the public key may be used in conjunction with other web services including X-KISS.

X-KRSS is designed to support all of the functions associated with the public key lifecycle:

Registration. The registration function supports registration of an association of a public key and additional data (such as a name) to create a 'key binding'. Private keys may be generated either locally by the client (desirable for signing keys) or by a central key generation service (desirable in cases where key recovery is supported). Requests may be authenticated with either a limited use shared secret or a digital signature.

Renewal. XKMS allows a PKI to be operated without digital certificates ever being issued, eliminating the need for certificate renewal. In cases where certificates are issued by the underlying PKI renewal processing may be performed automatically without the need for client interaction.

Revocation. An authorized party may request that the trust service revoke a key binding. This may be necessary because the key has been compromised or because information contained in the key binding is incorrect.

Recovery. Private key recovery is essential when an end user has lost their private key and requires access to their encrypted data. The X-KRSS recovery function provides an authenticated means of re-issuing a private key to a user.

X-KRSS may be configured hierarchically in the manner of a Local Registration Authority. This allows a registration request to be authenticated by a local trust service then passed on to another trust service where actual processing is performed.

2.2.3 SAML

The Security Assertion Markup Language [SAML] specifies both the TAXI Tier 3 trust assertion framework and specific assertion statements to support federated authentication and authorization applications.

Each trust assertion is encoded in a common XML package, which at a minimum consists of:

Basic Information.
Each assertion must specify the version of the SAML assertion syntax, a unique identifier that serves as a name for the assertion, a unique identifier for the issuer and the time instant of issue.

The Asserted Statement(s)
The statement(s) that are asserted by the issuer of the assertion.

In addition an assertion may contain the following additional elements:

Conditions.
The assertion status may be subject to conditions. The status of the assertion might be dependent on additional information from a validation service. The assertion may be dependent on other assertions being valid. The assertion may only be valid if the relying party is a member of a particular audience.

Advice.
Assertions may contain additional information as advice. The advice element

MAY be used to specify the assertions that were used to make a policy decision.

Relying applications may ignore advice elements but are required to understand all the conditions elements in an assertion if they are to rely on it.

SAML defines three Assertion Statements as follows:

Authentication Assertion

An authentication assertion contains a statement made by the issuer that asserts the subject was authenticated by a particular means at a particular time.

Authorization Decision Assertion

An authorization decision assertion contains a statement made by the issuer that asserts the request for access by the specified subject to the specified object has resulted in the specified decision on the basis of some optionally specified evidence.

Attribute Assertion

An attribute assertion contains a statement made by the issuer that asserts the specified subject is associated with the specified attribute(s).

2.2.4 XTAML

One of the most common objections made to the XKMS trust service model is that it does not provide a means of establishing and maintaining the trust relationship between the client and the trust service. XKMS cannot eliminate the need to implement X.509 if a certificate is still required to secure this trust relationship. XTAML is designed to meet this need in the context of a large scale PKI deployment in which a root of trust might be embedded in a large number of devices and consequently there is a need to be able to manage the private keys associated with the root of trust itself in a highly controlled offline environment that is independent of the online private keys used to authenticate actual Web Services transactions.

By design XTAML supports only the most limited delegation model. In the X.509 model a certificate signing certificate may be used to delegate a signing authority that is restricted to particular domains and/or certification policy. In contrast the XTAML delegation model provides only 'all or nothing' delegation required to support the requirements of online/offline key management.

The XML Trust Axiom Markup Language (XTAML) defines SAML Trust Assertions that support the management of trust axioms. A trust axiom is a 'root of trust' analogous to a root certificate in a certificate based PKI. An important application of trust axioms is managing the trust relationship between a client and a trust service.

XTAML defines SAML statement elements for specifying axiomatic and delegate keys and for asserting the validity status of another assertion. A new condition element is defined that makes the validity status of an assertion dependent on online verification. Two new advice elements are defined to allow an assertion to provide advice on the reissue of the assertion and for issue of related assertions.

2.2.5 XKASS

Another objections made to the use of XML Signature to authenticate Web Service requests and responses such as XKMS is the processing overhead required to create and verify digital signatures.

XKASS provides a means of using a lightweight Message Authentication Code (MAC) to authenticate Web Service messages by means of a shared secret established through a key agreement mechanism. The design of the XKASS is similar to the Just fast Keying [JFK] proposal made to the IETF IPSEC working group but requires only one round trip in the typical case instead of two made possible by a different approach to the handling of Denial of Service attacks.

2.3 Assertion Calculus

The SAML specification defines a framework for encoding Trust Assertions but does not provide a general framework for defining the semantics of assertion statements. One means of attaching specific semantics to an assertion statement is by means of an assertion calculus that sets out the rules by which a set of assertions are reduced to specific actions in response to a query.

Each assertion calculus is specific to an application such as access control or management of financial instruments. The laws of the assertion calculus comprise a formal specification

For example in an access control application an attempt to access a resource would generate a query of the form:

[Q1] Is *Alice* granted *Read* access to the *Accounts* file?

Given the assertions:

[A1] *Alice* is a member of the *Finance* group

[A2] The *Finance* Group is granted *Read* access to the *Accounts file*

The query may be answered by applying the rule

[R1] IF (*P* is a member of the *Q* group) AND (The *Q* group is granted *X* access to *Y*) THEN
P is granted *X* access to *Y*.

[P1] Applying R1 to A1 and A2, substituting *Alice* for *P*, *Finance* for *Q*, *Read* for *X* and *Accounts file* for *Y* we obtain:

[A3] *Alice* is granted *Read* access to the *Accounts file*

If the rules of the assertion calculus are labeled and specified in a suitable form the proof might be encoded in XML and attached to assertion A3 encoding the conclusion as advice.

While an application might employ a general purpose theorem

One of the principal advantages of the assertion calculus approach is that it allows an assertion generator to incorporate an integral verification step that independently verifies the correctness of the assertion by verifying the proof. Such a process is well within the capabilities of current formal methods tools, which cannot currently be said for the process of generating a proof in an arbitrary calculus.

3 Applications

The TAXI architecture reduces the complexity of a large number of PKI applications of which we present a representative sample only.

3.1 Facilitating Deployment of Traditional PKI

The principal design goal for TAXI was to facilitate the deployment of traditional PKI by eliminating the need for a 'fat client' to support sophisticated PKI functionality. This goal is realized in the XKMS specification that allows a simple client to access a sophisticated PKI by means of the XKMS Web Service interface.

XKMS enables deployment of sophisticated PKI topologies such as the Federal Government Bridge CA [FBCA] without the need to deploy PKI plug-in applications to support the specific topology.

3.2 Wireless LAN Configuration

A wireless LAN protocol such as 802.11b allows a user within range of an access point access to a LAN without the need for a physical connection. By eliminating the need for physical access a wireless LAN protocol removes a control that mitigates two significant security risks, first anyone within range might intercept network traffic, second unauthorized use of the network.

Recent analysis [Borisov01] of the 802.11b WEP cryptographic protocol [WEP] has demonstrated that the WEP protocol provides inadequate protection against the interception risk and little protection against the unauthorized use risk.

The risk of unauthorized use arises from the fact that every user of the network shares the same authentication key. The risk of unauthorized use could have been controlled if a sufficiently lightweight PKI had been available to the developers. For example XKMS might be used to permit network interface cards to be granted or denied access to the network on the basis of a private key embedded in the card.

3.3 Negotiable Financial Documents

Many financial transactions are represented by the exchange of negotiable documents. In many cases these documents are bearer instruments.

For example a ship has fulfilled its obligations to the dispatcher when it discharges its cargo to the first person to present a valid bill of lading at the destination port.

Replacing paper documents with electronic representations offers many advantages including lower costs for the carrier and its customers. In addition an electronic instrument is more readily traded than one restricted to physical form.

A tier 3 trust assertion may be used to create an electronic bill of lading that tracks the current ownership of a specific asset (e.g. a cargo) and manages transfer of that asset from one owner to another by means of tier 4 status assertions. A potential purchaser of a cargo may determine if the seller is currently the owner of the cargo by validating the assertion stating ownership.

3.4 Commercial Registry

Many business applications involve some form of registry. For example in the US it is possible to gain security for a debt by registering a charge against assets of the debtor in a commercial registry.

A commercial registry does not normally require exceptional levels of availability, it is however essential that the registry ensure an exceptional level of data authenticity and persistence. Although the human interface to such a registry is likely to require customization to the applicable law, the type of assets registered, language, etc. the functions requiring exceptional levels of data authenticity and persistence are common to all registries.

Entries in the commercial registry may be represented by tier 3 trust assertions. Discharge or voiding of entries may be represented by means of tier 4 status assertions.

4 Acknowledgements

Thanks are due to Warwick Ford, Barbara Fox, Brian LaMachia, Jeremy Epstein, David Solo and Mack Hicks for their many helpful comments on the original TAXI research project. Thanks are also due to the members of the SAML and XKMS working groups who have helped to turn theory into practice, in particular Stephen Farrell, Shivram Mysore, Eve Maler, Joe Pato, Jeff Hodges, Prateek Mishra, David Orchard, Hal Lockhart, Carlisle Adams, Tim Moses, Bob Blakely, Marlena Erdos, Scott Cantor, Chris McLaren, Krishna Sankar, Irving Reid, Daniel Ash, Joseph Reagle and Blair Dillaway.

5 References

[DNSSEC] Eastlake, D. and C. Kaufman, *Proposed Standard for DNS Security*, RFC 2065, January 1997.

[FBCA] W. T. Polk and N. E. Hastings, *Bridge Certification Authorities: Conecting B2B Public Key Infrastructures*. NIST 2001 http://csrc.nist.gov/pki/documents/B2B-article.pdf

[Identrus] Identrus, web site http://www.identrus.com/

[JFK] W. Aiello, S.M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A.D. Keromytis, O. Reingold *Just Fast Keying (JFK)*, Internet draft http://www.ietf.org/internet-drafts/draft-ietf-ipsec-jfk-00.txt

[Kohn78] Kohnfelder, L. M. (1978). *Towards a Practical Public-Key Cryptosystem. Laboratory for Computer Science*. Cambridge, Massachussetts Institute of Technology.

[LDAP] T. Howes, M. Smith, *The LDAP Application Program Interface*. RFC 1823 August 1995.

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

[PGP] Atkins, D., Stallings, W. and P. Zimmermann, *PGP Message Exchange Formats*, RFC 1991, August 1996.

[PKIX] Public Key Infrastructure X.509, Internet Engineering Taskforce.

[SAML] P. Hallam-Baker and Eve Maler, *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)* http://www.oasis-

open.org/committees/security/docs/draft-sstc-core-25.pdf

[SDSI] Ron Rivest and Butler Lampson, SDSI - A Simple Distributed Security Infrastructure [SDSI], http://theory.lcs.mit.edu/~cis/sdsi.html

[Sebiok] T. Sebiok, *Signs: An Introduction to Semiotics*. Toronto: University of Toronto Press, 1994

[SOAP] D. Box, D Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Frystyk Nielsen, S Thatte, D. Winer. *Simple Object Access Protocol (SOAP) 1.1*, W3C Note 08 May 2000, http://www.w3.org/TR/SOAP/

[SPKI] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. *SPKI Certificate Theory*, RFC 2693, September 1999.

[Borisov01] Nikita Borisov, Ian Goldberg, and David Wagner. *Intercepting mobile communications: The insecurity* of 802.11. In Proceedings of MOBICOM 2001, 2001. http://www.isaac.cs.berkeley.edu/isaac/mobicom.pdf

[WAP] WAP Certificate profile Specification, http://www1.wapforum.org/tech/terms.asp?doc=WAP-211-WAPCert-20010522-a.pdf

[WEP] *LAN MAN Standards of the IEEE Computer Society. Wireless LAN medium access control (MAC) and physical layer (PHY) specification. IEEE Standard 802.11, 1977 Edition*, 1977

[X.500] ITU-T Recommendation X.501: *Information Technology - Open Systems Interconnection - The Directory: Models*, 1993.

[X.509] R. Housley, W. Ford, W. Polk, D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*. RFC 2459, January 1999.

[XKASS] P. Hallam-Baker, *XML Key Agreement Service Specification (XKASS)*, May 2001, XML Trust Center Research note, http://www.xmltrustcenter.org/research/docs/X-KASS-31.pdf

[XKMS] W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, J. Lapp, XML Key Management Specification (XKMS), W3C Note 30 March 2001, http://www.w3.org/TR/xkms/

[XTAML] P. Hallam-Baker, *XML Trust Axiom Markup Language 1.0*, VeriSign Inc. September 2001. http://www.xmltrustcenter.org/

[XML-SIG] D. Eastlake, J. R., D. Solo, M. Bartel, J. Boyer , B. Fox , E. Simon. *XML-Signature Syntax and Processing*, World Wide Web Consortium. http://www.w3.org/TR/xmldsig-core/

Making certificates programmable

John DeTreville
Microsoft Research
johndetr@microsoft.com

Abstract

Certificates carry signed statements within a Public-Key Infrastructure (PKI). As we begin to build more complex and more open PKIs, the limited expressiveness of current certificate languages becomes a concern. While certificates are traditionally treated as simple data structures conforming to a given schema, we show an alternative derivation of the concept of a certificate in which certificates can contain control information in the form of program code. One example is program code written in declarative statements in a variant of the relational algebra, which can work together in rich ways.

1. Introduction

In a Public-Key Infrastructure (PKI)—such as X.509 [10] or SDSI/SPKI [13, 7]—distributed parties can communicate using persistent signed data structures called *certificates*. Certificates can carry authorizations that control access to distributed resources (saying, for example, that John Smith can access a particular Web site at his workplace) as well as more abstract data and rules that can provide support for authorization decisions (e.g., John Smith is a full-time programmer; programmers are employees; full-time employees can access the Web site). Certificates conform to an established syntax—such as ASN.1 for X.509 certificates [11] and encoded S-expressions or XML for SDSI/SPKI certificates [12]—and an established semantics.

As our ambitions for PKIs become greater, the expressiveness of their certificates can become a cause for concern. We might wonder whether our certificates—their syntax and their semantics—are expressive enough. Can they convey the necessary sorts of information to support the operation of the PKI? For example, if our certificates are very simple data structures that can work together only in a few restricted ways, it might be impossible to support a rich variety of authorization structures. While this may be seen as an advantage in some contexts (for example, if we might wish to constrain the uses of a PKI), it is certainly a potential shortcoming in a more open environment.

We might also wonder if our certificates and our certificate language are suitably well-defined. Ensuring the wide interoperability of certificates in an open PKI can be difficult or impossible in practice [9]. We note for example that certificates are often extended for new uses by simply adding new fields in a manner that can change the meaning of existing fields in subtle and perhaps unforeseen ways, breaking existing uses. Conversely, we might expect that a more regular design, based on fewer base concepts that can be used together in more ways, might improve interoperability while at the same time increasing expressiveness.

In this paper we rederive the concept of a certificate in a novel way, in which a certificate can contain program code, written in a simple declarative language, as well as data. The use of program code can increase the expressiveness of certificates while eliminating a number of special cases present in existing certificate languages, and

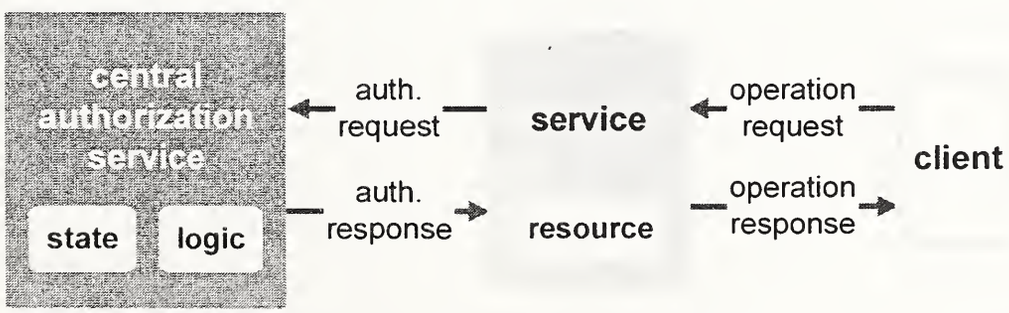


Figure 1: A hypothetical central authorization service

is one path toward deriving more powerful certificate languages that will allow us to build richer and more flexible PKIs.

2. A hypothetical central authorization service

The principal purpose of certificates—let us say—is to support authorization in an open distributed environment. Certificates therefore combine two distinct kinds of information. First, they include information directly related to authorization. For example, they may state that a certain group of people is authorized to access a shared resource, or that a certain person belongs to that group. They also include information required by their use in an open distributed environment. For example, they may include a validity interval, or an address to check for revocation, or information that supports the proper chaining of certificates.

To help separate these concerns, let us first consider a hypothetical environment where all authorization decisions have been centralized, as shown in Figure 1. Whenever a client requests an operation from a service controlling a resource, the service must determine whether this client is authorized to perform this operation; in this centralized model, the service simply passes an authorization request to the central authorization service, identifying the client, the resource, and the requested operation. Based on its encapsulated state and logic, the central authorization service authorizes or rejects the operation; if the operation is authorized, the service performs the requested operation and returns the result to the client. The central authorization service encapsulates the system's authorization information (its "state") and the authorization rules (its "logic") for all resources and for all clients, and it is used only as a "black box" that can only answer specific questions.

Such a centralized authorization service is of course impractical in many ways. Its performance and availability would be limited and it certainly could not scale to the size of the Internet. Worse yet, such a large-scale service would be impossible to administer, since it would combine information from thousands or millions of autonomous administrative domains and would hard-code the rules on how these domains operate and how they interoperate. It would be closed because third parties could not readily extend its state and logic.

Let us imagine, though, that our centralized distribution service is otherwise powerful enough to perform the needed authorization tasks, and that its only problems are those due to its centralized nature. How can we solve these problems, or at least ameliorate them? In other words, how can we decentralize (i.e., distribute) the authorization service?

3. Mobile code

One approach to decentralizing the authorization service is to make its state and logic *mobile*—that is, to encapsulate some piece of its state and logic in a certificate that can travel across the network to the service controlling the resource and execute there. There have been various proposals that support this sort of mobile code [4] and this approach is greatly simplified when the authorization process is purely functional—without side-effects—as is usually the case. We assume some mechanism for executing the code in the certificate safely at the receiving service.

Simply adding mobile code to our centralized design is not enough. It improves performance, and it improves availability, but it does not address the remaining problem of administering the system's global authorization state and logic. We can simply partition the state and logic, of course—and such a partition is clearly the solution—but the various partitioned administrative domains must still be able to interoperate. Below, we derive a architecture for partitioning that allows multiple administrative domains to interact in flexible ways. Our language for state and logic is purely applicative, thus allowing its safe execution at the recipient.

4. Certificates as cache entries

One way to improve performance and availability in any system is through the use of *caching*. Once a service sends a request to our hypothetical central authorization service and receives a response, it can cache the request-response pair to avoid requerying the central service for the same request in the future. Of course, the response must not depend on state that can change.

In their simplest form, certificates are an extension of the caching idea. As shown in Figure 2, a service can hold a certificate, signed by the central authorization service, encapsulating the request-response pair. It can use this certificate exactly as it would use the corresponding cache entry, but the certificate has several additional advantages.

- Cache entries are implicitly authenticated because the service (presumably) knows that the information in the cache came from the central authentication service, over an authenticated connection. In contrast, a certificate is explicitly authenticated because it carries a signature from the central authentication service. A service can trust a certificate received from another service, or even from a client. This feature further improves the performance and flexibility of the PKI.

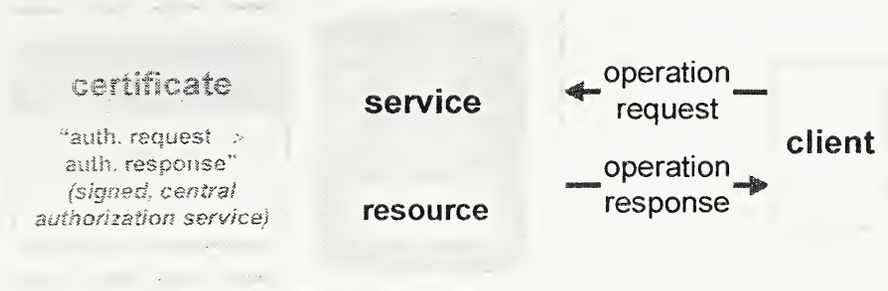


Figure 2: Certificate issued by a central authorization service

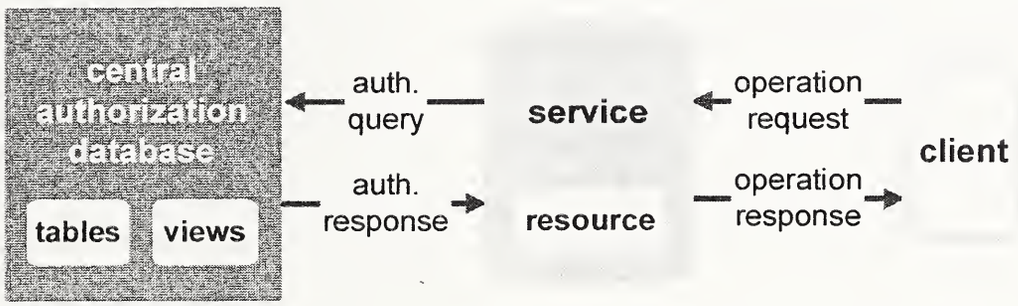


Figure 3: A hypothetical central authorization database

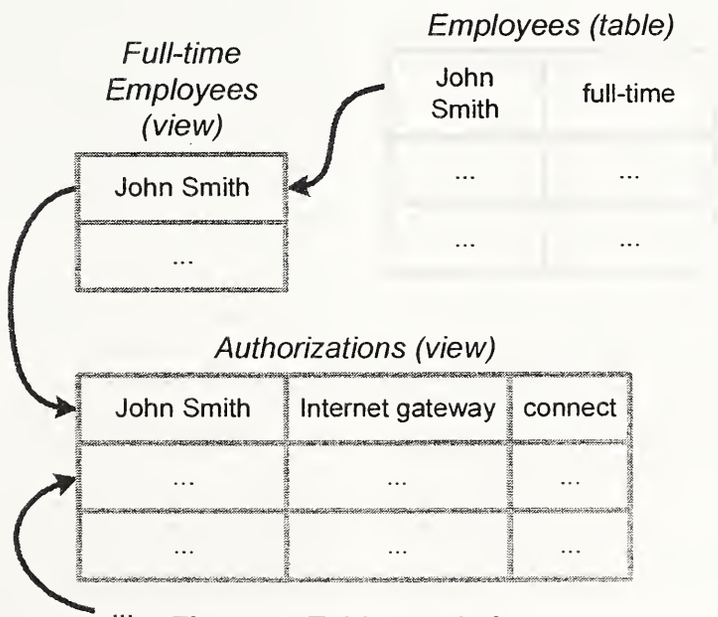


Figure 4: Tables and views in the central authorization database

- A certificate can potentially be obtained at a convenient time before it is needed. While a cache operates transparently, meaning that any request might need to contact the central authorization service, certificates allow us to explicitly collect—ahead of time—all of the information needed to authorize an operation, eliminating the need for the central authorization service to be available at the same time as each operation. This feature improves the availability of the PKI.
- Instead of supplying the response for one particular request, a certificate can contain wild cards, supplying the responses for a family of requests. For example, a certificate can say that a certain set of individuals—defined in some way—is authorized to perform a certain set of operations on a certain set of resources. This feature improves the performance and flexibility of the PKI. We will return to the idea of wild cards later in this paper.

In the simple use of certificates shown, the central authorization service remains a black box and does not expose or export its internal state and logic to its callers except in the form of request-response pairs. In the following sections we will make the black box more transparent by extending and regularizing the statements that certificates can carry.

5. Using a relational database to represent state and logic

To expose the internal structure of the central authorization service, it is necessary first to specify what forms the state and rules can take. In this section, we demonstrate how its state and logic can be modeled by a relational database [5, 8].

As shown in Figure 3, the central authorization database contains *tables* and *views*. Tables store data, while views are defined in terms of data that appear in tables and other views. The service receiving an operation request sends an authorization query to the database, and receives an authorization response.

Figure 4 shows the internal organization of one example database in further detail. Here, full-time employees are authorized to connect to an Internet gateway. An Employees table holds the names of the employees and their employment status. A Full-time Employees view is derived from the Employees table, and the final Authorizations view is further derived from the Full-time Employees view. In this simple example, the Employees table holds the raw data while the Full-time Employees view and the Authorizations view serve to encode the authorization logic.

When this example database is used, a service queries the Authorizations view at the authorization database, giving the client name ("John Smith," or more generally a public key), resource name ("Internet gateway"), and operation name ("connect") as keys. The database responds to the query by returning all matching rows. In this example, the database returns one row in case of authorization success, and zero rows in case of failure.

We can define the database views and queries in a number of forms, including relational algebra, which operates on tables and queries using operators like *select*, *project*, and *join*. In this paper, we extend the relational algebra with two additional operators.

- We add a *union* operator that combines tables or views with the same schema. Although the Authorizations view is shown here as a simple view on the Full-time Employees view, it would more generally be the union of a number of views, each of which might define authorizations on a particular resource, set of resources, etc.
- We also add *recursion*, to allow for the computation of transitive closures. This is useful for modeling authorization chains, as discussed below.

Because nonmonotonicity can be unsafe in a distributed environment, we additionally restrict our relational algebra to be monotonic by eliminating negation. It is a topic for future work to characterize those uses of nonmonotonicity and negation that nevertheless can be safely allowed.

While the schema of the Authorizations view must be partly standardized—and known to the services querying the authorization database—the schemas of the other views and tables need not be standardized at all. This can be seen as a significant advance over older PKI schemes like X.509 and even SDSI/SPKI. The tables can include arbitrary data with arbitrary structure, and the Authorizations view can be the result of arbitrary computations on these tables. (Of course, these computations must be expressible in our extended relational algebra; this is true for the classes of authorization problems that we have studied.)

Traditional security languages include special-case syntax and semantics for encoding extra conditions and information needed for authorization. Because of the use of arbitrary schemas and the power of the extended relational algebra, though, the authorization database can represent these conditions and information directly. For example, while SDSI/SPKI includes a mechanism for group membership, we note that our authorization database can model groups directly in the relational algebra, as in the example above. We can also represent different kinds of groups, such as groups of resources or groups of operations; this is impossible or limited in traditional languages. Similarly, we can model the idea of certification

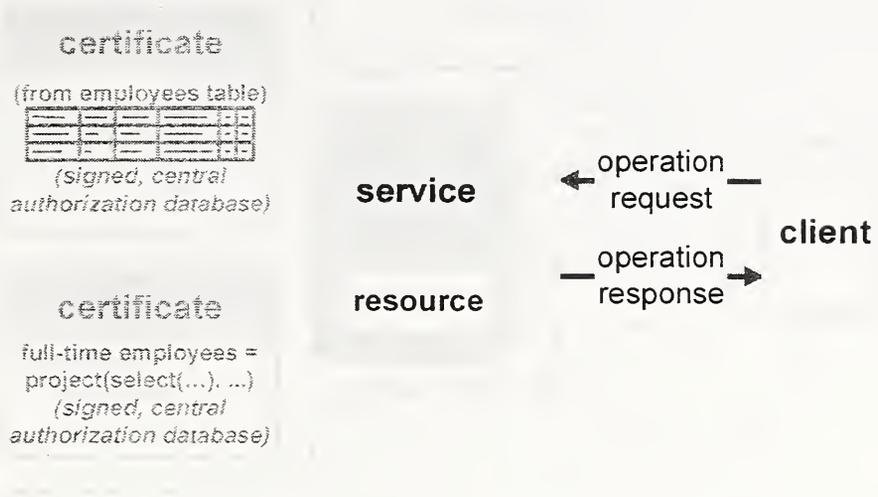


Figure 5: Certificates issued by a central authorization database

authorities and certificate chains, as in X.509 and SDSI/SPKI, directly in the extended relational algebra instead of building it into our language. (This requires the addition of recursion to the relational algebra, as discussed above.) Different administrative domains can be programmed to have different properties, and we can also generalize the use of one-dimensional chains to allow more complex and more general trust relations.

(We note that the relational algebra is closely related to the logic-programming language datalog [1]. The central authorization database can therefore be replaced by a program written in datalog or another logic-programming language, as in the Binder security language [6].)

Choosing to represent our authorization information and rules in a relational database system might seem as merely shifting our problems from one domain to another. However, there is a wealth of experience in designing good relational database schemas [2]—such as the use of normal forms—as well as formalizing the semantics of schemas. We believe that many of the problems of authorization are simplified by restatement in the context of databases, relational algebra, and logic programming. Furthermore, the greater generality of the database context can lead to a more general solution to the authorization problem.

6. Certificates as signed database excerpts

Certificates served to encapsulate request-reply pairs with our original central authorization service, and they play much the same role in conjunction with the central authorization database. However, since we can now expose some of the internal structure of the central authorization database—we can name its tables and its views and give their schemas and definitions—we can now store much richer information in our certificates.

As shown in Figure 5, services still use the certificates issued by the central authorization database in lieu of an on-line request and reply. Unlike the earlier use of certificates, though—in which certificates simply cached signed request-reply pairs—these certificates can store additional information which the services can use to derive future authorizations. Figure 5 outlines the two types of certificates that the central authorization database can now issue.

- The first type of certificate includes an excerpt—one or more rows—from a table or view. Here, the first certificate includes rows from the Employees table. This type of certificate states that the excerpted rows were found in the named table or view.
- The second type of certificate defines a view in terms of a relational algebra expression involving other tables and other views. Here, the second certificate includes the definition of the Full-time Employees view in terms of the Employees table.

These certificates are, of course, still signed by the central authorization database, and can be received from the central authorization database or from a client or other service. These database certificates name the table or view that their information comes from, and also include enough schema information to allow their interpretation at the service.

The database certificates can include enough information to derive the replies for many different requests. (This is an example of the wild-card feature described earlier.) Just as we do not require these certificates to include *all* of the rows of a table or view, they also need not

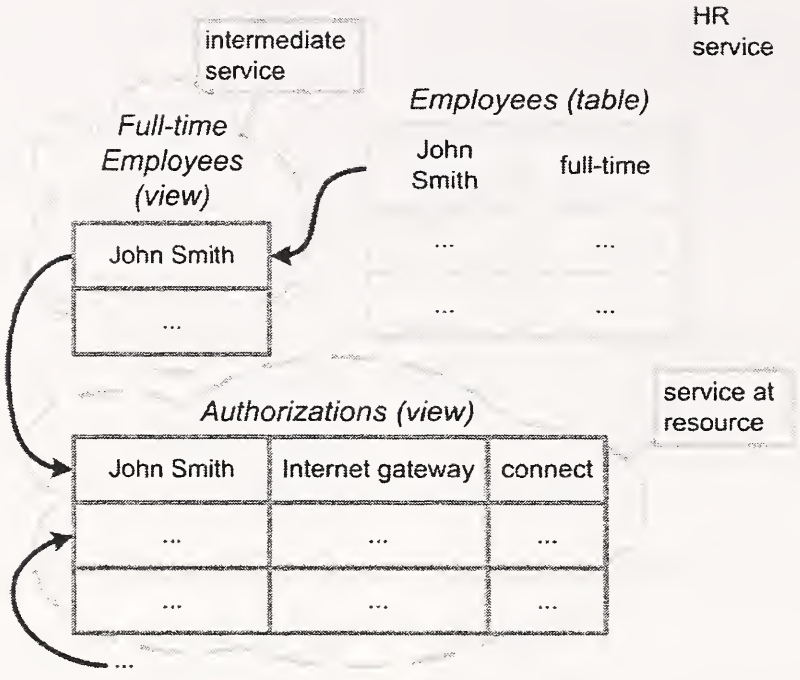


Figure 6: Tables and views in the distributed authorization database

contain the *complete* definition of a view. For example, a database certificate encapsulating the Authorizations view—which might be the union of a large number of views—can simply say that it includes one particular view. Database certificates therefore contain only partial information; they can say only that a given authorization does exist, and cannot say that it does not. (Extensions to partially eliminate this restriction are possible but are outside the scope of this paper.)

Constraining the structure of the central authorization service to be a relational database thus allows our certificates to include richer, more general forms of information. Our central authorization database can issue certificates whose meaning cannot be represented in X.509 or in SDSI/SPKI—as illustrated below—and it regularizes the treatment of existing features.

7. Distributing the database

Our central authorization database is still centralized, and while the use of certificates has reduced the problems of performance and availability, they still exist. Worse, we have not attacked the administrative problems inherent in a centralized architecture. To eliminate these problems, we now show how to partition the central authorization database into a distributed authorization database.

Figure 6 illustrates the operation of the distributed authorization database. The database still contains tables and views, but they are stored in multiple services on the network. In this example, for instance, a Human Resources (HR) service holds the Employees table, but the service controlling the resource itself can define the portion of the Authorizations view that it interprets. Yet another intermediate service can define the Full-time Employees view referenced by the Authorizations view.

Although most tables and views can be stored anywhere on the network, we require that the Authorizations view be distributed among the services that control the various network resources. The distributed authorization database thus follows the lead of the PolicyMaker language [3], in which the root of all authorization decisions is local by convention and is established administratively. Distributed certificates are still used in the same way as our earlier certificates. As shown earlier in Figure 5, a service controlling a resource can use multiple certificates to make authorization decisions. When these are distributed authorization certificates, they may come from multiple services.

As shown in Figure 7, certificates are signed by the services that issue them. Here, Employee certificates are signed by the HR service, while Full-time Employee certificates are signed by the intermediate service. The service at the resource need not sign its definition of the Authorizations view to use it, since it originates locally. Each definition of a view identifies the public key used by the tables or views it uses an inputs.

We have thus eliminated the need for the central database service to issue and sign certificates. Since multiple autonomous services can now issue certificates, we

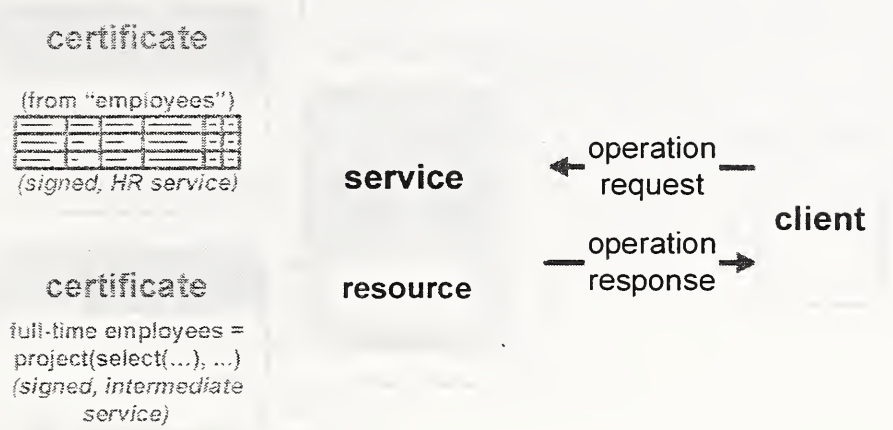


Figure 7:
Certificates issued by a distributed authorization database

can directly accommodate multiple administrative domains. Administrative domains can interoperate because they can explicitly refer to one another by the public keys of the issuing services. The resulting system is similar in many ways to traditional uses of certificates but it has some notable differences. In particular, references to public keys need not be constant, but can themselves be drawn from tables and views, as shown in Figure 8. Here, the policy expressed is that full-time employees can access the Internet gateway if authorized by their bosses. We combine our earlier Full-time Employees view with a Bosses view, as well as an Approvals view local to each boss.

Allowing views in one service to refer to tables or views in another allows the PKI designer to use an arbitrary number of levels of indirection. Since it is a folk theorem in Computer Science that any problem in computing can be solved by adding another level of indirection, we can expect that this will be a powerful technique, and that it will serve to make explicit and to extend some number of security assumptions that might otherwise be wired into the system architecture.

In particular, this distributed certificate structure provides a concrete interpretation of the abstract notion of "trust." One service trusts another if its views depend on tables or views from that other service. Because the database can hold the names of services (e.g., their public keys), we can organize services into groups or other more complex relations. For example, we might have a table of which services "trust" which others. Certification Authorities are no longer special entities in our PKI; we can choose to implement them in the same form as in traditional PKIs—that is, their certificates can continue to bind names to identities, or to delegate the power to issue further certificates—or we can choose different schemas that take advantage of our greater flexibility and generality.

9. Conclusions and future work

We have shown how certificates can be made more expressive and more precise by allowing them to include program code written in a language such as an enhanced relational algebra. While we have outlined the operation of such a system, much future work is clearly needed.

We have not touched on certificate revocation in this paper. While the standard techniques for revocation continue to apply, we would still like to understand how to make revocation programmable, as well as checking for revocation. More generally, we have assumed that the statements in our system has no side effects, which is clearly a poor assumption in many cases.

While making certificates programmable increases their expressiveness, greater expressiveness can always be misused and can in fact keep us from saying the right things by making it too easy to say the wrong things, or to understand the implications of our statements. Thus, the choice of a security language ultimately involves an engineering tradeoff between increasing generality and maintaining usability. Understanding this tradeoff again requires further experience.

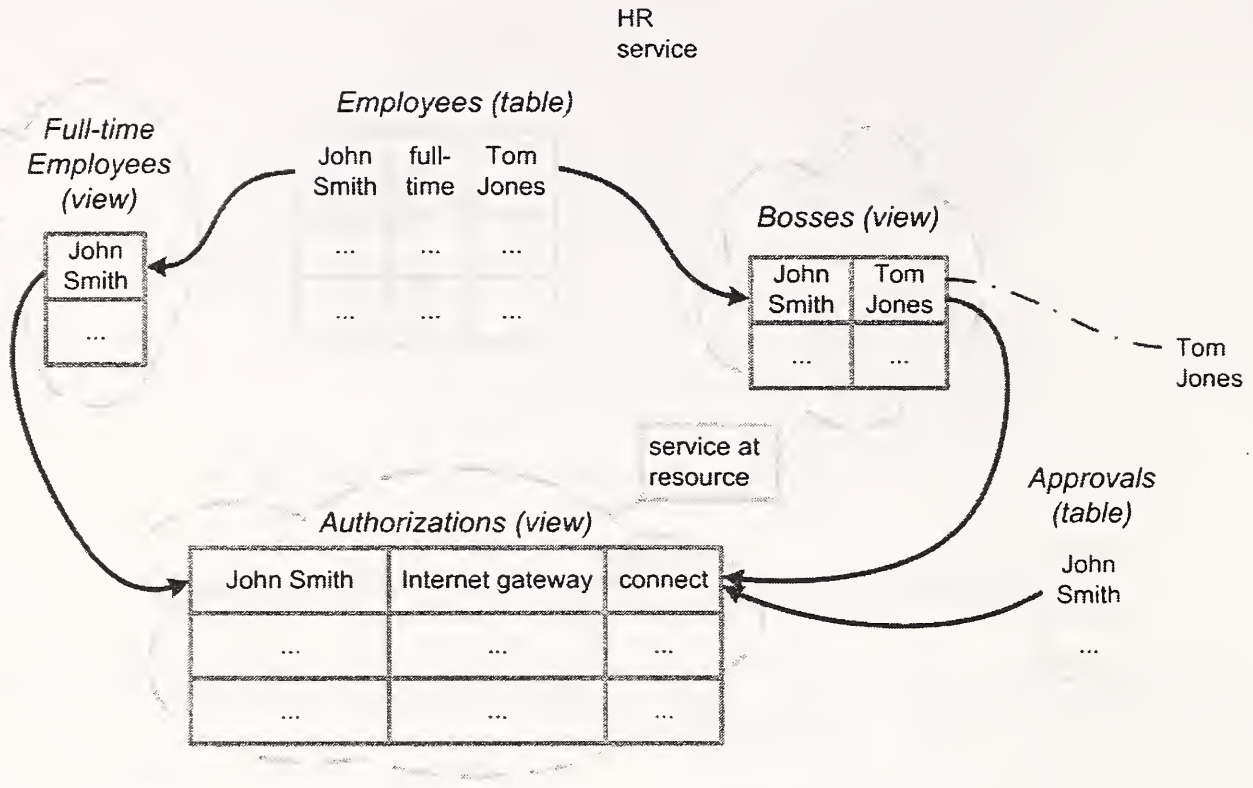


Figure 8: Adding another level of indirection

References

[1] M. Ajtai and Y. Gurevich. "Datalog vs. first-order logic." In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 142–146, 1989.

[2] J. Biskup. "Achievements of relational database schema design theory revisited." In B. Thalheim and L. Libkin, eds., *Semantics in Databases*, Lecture Notes in Computer Science, Vol. 1358, pages 29–54. Springer-Verlag, 1998.

[3] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. 1996 IEEE Symp. on Security and Privacy*, May 1996.

[4] L. Cardelli. "Abstractions for mobile computation." In J. Vitek and C. Jensen, eds., *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, vol. 1603 of LNCS, pp. 51–94. Springer-Verlag, 1999.

[5] E. F. Codd. "A relational model for large shared data banks." *Comm. of the ACM*, 13(6):377–387, June 1970

[6] DeTreville, J. 2002. "Binder: a logic-based security language." To appear, *Proc. 2002 IEEE Symp. on Security and Privacy*, May 2002.

[7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. "SPKI certificate theory." IETF RFC 1693, September 1999.

[8] J. Gray, *et al.* "System R: Relational approach to database management." *ACM Trans. on Database Systems* 1(2), pages 97–137, June 1976.

[9] P. Gutmann, "X.509 style guide," available at http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt, October 2000.

[10] ITU-T Recommendation X.509, "The directory: public-key and attribute certificate frameworks." March 2000.

[11] ITU-T Recommendation X.680. "Abstract Notation One (ASN.1): Specification of basic notation." December 1997.

[12] X. Orri & Mas, J.M. 2001. "SPKI-XML certificate structure." IETF Internet Draft, November 2001.

[13] R. Rivest and B. Lampson. "SDSI—a simple distributed security infrastructure," available at http://theory.lcs.mit.edu/~cis/sdsi.html.

A Distributed Credential Management System for SPKI-based Delegation Systems

Óscar Cánovas[†], Antonio F. Gómez[‡]
[†]*Department of Computer Engineering*
[‡]*Department of Information and Communications Engineering*
University of Murcia
30071 Murcia (Spain)
ocanovas@um.es, skarmeta@dif.um.es

Abstract

Traditionally, certificates have been used to link a public key to a particular name identifying that key. However, public key certificates are digitally-signed statements which can be used in order to assert many other types of information. SPKI has become one of the most outstanding proposals referring to authorization, and several applications have been based on SPKI certificates in order to provide authorization services to well-known scenarios in distributed systems. Most of these scenarios are based on delegation, where resource guards have an ACL with few entries granting keys belonging to some authorization or naming authorities the right to delegate all access to the controlled resources. These authorities can issue certificates delegating these permissions to other subordinates authorities, or to specific users. In this way, the structure generated reflects the system management process. However, generation of these certificates usually is system-dependent. In this paper, we present a management system that can be used in all SPKI scenarios based on delegation. This system addresses some problems related to scalability, certificate distribution, and interoperability. We define how certification requests can be expressed, how different security policies can be enforced using this system, which are the entities involved in a certification scenario, and we propose a mechanism able to exchange authorization-related information among these entities.

1 Introduction

Loren Kohnfelder defined "certificate" in 1978 as a digitally-signed statement holding a name and a public key, and nowadays the words *certificate* and *identity certificate* are still used as synonyms. However, a certificate is a record stating some information about the entity the certificate was issued to, and this information may be a role membership statement, or an authorization. Authorization certificates bind a capability to a key, and this capability can be used to determine what the entities are allowed to do.

One of the most outstanding proposals related to this type of certificates has been the SPKI/SDSI infrastructure [8]. SPKI/SDSI provides three types of digital certificates (ID, attribute, and authorization) that can be used in several security scenarios. In fact, there are several proposals which make use of SPKI certificates in order to provide authorization services to many different application environments, such as WLAN networks [10], CORBA distributed objects [12], or web servers [4]. Most of these scenarios are based on delegation, where resource guards have an ACL with few entries granting keys belonging to some authorization or naming authorities the right to delegate all access to the controlled resources. However, some of these proposals do not explain how certificates are issued by the authorities, and this usually is application-dependent. Although simple and not distributed approaches can constitute a good alternative for small scenarios, some problems derived from scalability or interoperability might arise in more complex environments [3]. Generation or revocation of these certificates should not be implemented using simple command-line applications. A structured and distributed system must be provided.

A system is necessary which addresses the problems related to scalability, certificate distribution, and interoperability. In this paper, we present

DCMS (Distributed Credential Management System). DCMS defines how certification requests should be expressed, how different security policies can be enforced using this system, which are the entities involved in a certification scenario, and how these entities can exchange authorization-related information. We have used the AMBAR (Access Management Based on Authorization Reduction) protocol [2] in order to perform that exchange, but similar protocols can be also used. This system is divided into the naming management system (NMS), which manages the issues related to SPKI ID certificates, and the authorization management system (AMS), which is responsible for those procedures related to SPKI attribute and authorization certificates. We believe that this system can lead up to the definition of an application-independent system which can be used in order to provide authorization services to many different scenarios based on delegation. DCMS also complements some proposed mechanisms for revocation and validation of SPKI certificates [11], and can make use of public repositories for certificate storage purposes [9].

We can find similar proposals in the literature. In [13], a security architecture is presented which is related to authentication, authorization and delegation in a distributed environment based on SPKI. This proposal differs from DCMS about object formats, and system structure. We use s-expressions in order to specify the authorization policies and requests, instead of HTTP-like messages and codes. We do not find necessary to use a different encoding since SPKI-like s-expressions are appropriate, straightforward, and standard. Moreover, we do provide a generic framework of authorities, proxies and protocols that can be used as guidelines to design and implement authorization management services. In fact, our system has been implemented using the Intel version 3.14 of CDSA (Common Data Security Architecture) [5].

This paper is organized as follows. Section 2 presents an authorization scenario based on delegation in order to clarify why DCMS is useful. Section 3 provides some basic background on the AMBAR protocol. Section 4 presents the entities involved in the naming management system (NMS), and shows the s-expressions that will be used in this system to specify certification requests and access control lists. Section 5 contains similar details concerning the authorization management system (AMS). Section 6 presents how system entities can interoperate using the AMBAR protocol. Finally, Section 7 makes some concluding remarks.

2 Motivation

In this section we are going to show a distributed system where SPKI certificates and delegation can be used to implement physical access control [3]. We will also explain why DCMS is necessary.

This distributed system is based on a RBAC (Role Based Access Control) model [15]. The central concept of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. This greatly simplifies management of permissions since the two relations are considered completely independent.

In this system, special devices named TICA are used, which are able to perform some access control operations like opening doors. They are located at the entrances of the different buildings and/or departments, and they can establish their own access control conditions, trusted entities, and authorization mechanisms. TICAs delegate authorization management to particular authorization authorities (AA). This is accomplished through authorization certificates issued by the TICAs for a set of specific AAs. These certificates basically give the AAs total authority over the device, and also permission to further delegate the access control is granted. TICAs can also delegate the authority by means of ACL entries containing the same information included in those certificates. Then, AAs usually create new attribute certificates giving a subset of permissions to the roles defined by any of the existing naming authorities (NA). Roles are managed by NAs, which issue ID certificates in order to state that a particular user has been assigned to a specific role. In this way, TICAs are the beginning of the authorization path, and not only the policy enforcement point. The device is able to make the security decision regarding the authorization data presented by the user requesting the access.

However, this certification management process must be designed and implemented using a scalable approach. An encoding for certification requests must be defined, and a mechanism is necessary which is able to exchange authorization-related information among the entities involved.

Using DCMS, once TICAs have delegated the authorization management task to the different au-

thorities, principals can request individual certificates in order to gain access. These requests can be generated and sent to the authorities by the principal itself, or can be submitted using trusted service access points (SAP). Authorities will issue the requested certificates depending on the authorization policy (authorities are the policy decision point). This policy can be represented using SPKI ACLs, a database or any other method, and it is system-dependent, although in the next sections we will assume that it is implemented using ACLs.

Figure 1 shows a particular scenario where TICAs delegate the authorization tasks to different AAs. Users make use of DCMS in order to obtain specific authorization certificates from these entities. In this case, we assume that authority A and the SAP are the AMBAR peers. Once the certificates are generated, these can be presented to the TICAs in order to gain access.

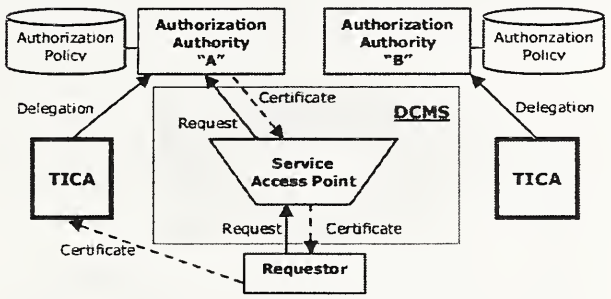


Figure 1: Use of DCMS

3 AMBAR Protocol

AMBAR (Access Management Based on Authorization Reduction) [2] is a protocol for secure exchange of authorization-related information based on public key cryptography. This protocol does not depend on a particular type of authorization or identity-based certificate, and it contains a negotiation phase designed to adapt the protocol to access control scenarios with different requirements (anonymity, confidentiality, credential recovery, etc.). In general, it provides functionality to transmit resource access requests, the authorization information related to those requests (credentials, ACLs), and results obtained from a certificate chain discovery method or compliance checker. ACLs can be transmitted in order to give some information to the client about which credentials would be necessary to access the resource. However, disclosure of security policies (ACLs are particular implementations of these policies) must be carefully performed since they can contain sensitive information [16].

It has been designed to be session-oriented in order to optimize those scenarios where the request/response messages are exchanged between the same client and server. In addition, it does not need to rely on any additional security protocol since it adds confidentiality and integrity to the data being transmitted.

The AMBAR protocol consists of different components organized, as Figure 2 illustrates, in two layers.

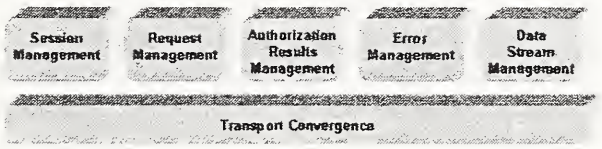


Figure 2: AMBAR Architecture

- **Session Management module (SM).** This module transmits the client and server security preferences, and generates the cryptographic data used by the TC layer to protect the subsequent communications. Clients and servers negotiate the following parameters:
  - *Symmetric cipher.* Parties select the symmetric cipher and its key length.
  - *Operation mode.* AMBAR supports two operation modes: anonymous client mode and fully identified.
  - *Identity-based certificates.* It is possible to select X.509, OpenPGP, or SPKI certificates.
  - *Authorization-based certificates.* AMBAR supports SPKI certificates, PKIX attribute certificates and KeyNote asserts.
  - *Credentials distribution.* Parties can select whether the credentials will be provided by the client (push), or will be obtained by the server from either a repository or an issuer (pull).
- **Request Management module (RM).** The RM module transmits two types of messages: messages related to authorization requests and credentials; and messages related to decisions and ACLs. Contents and the sequence of these

messages are determined by the negotiated operation mode and the method for distribution of credentials. As we mentioned previously, a session-oriented protocol allows some optimization to be performed. Therefore, the RM module could be responsible for optimizing authorization computations.

- **Authorization Results Management module (ARM).** The ARM module generates notifications and transmits the demanded resources. Negative notifications are transmitted by the server when the access is denied. If the access were granted, there would be two possible response messages: an affirmative notification if the client requested the execution of remote actions; or the controlled resource. It also enables (disables) the DSM module when an authorization request demanding the establishment (conclusion) of a data stream is granted.
- **Error Management module (EM).** Systems use the EM module to signal an error or caution condition to the other party in their communication. The EM module transmits a severity level and an error description.
- **Data Stream Management module (DSM).** The described request/response model is not suitable if we plan to use AMBAR as a transparent layer providing confidentiality, authentication and access control services. The DSM module, initially disabled, controls the transmission of arbitrary data streams, which are enabled once a request demanding the activation of this module is granted.
- **Transport Convergence module (TC).** The TC module provides a common format to frame SM, RM, ARM, EM, and DSM messages. This module takes the messages to be transmitted, authenticates the contents, then applies the agreed symmetric cipher (always a block-cipher), and encapsulates the results. The cryptographic data used to protect the information is computed by the SM module during the negotiation phase.

The AMBAR protocol is part of a complete authorization framework for certificate-based access control systems. It is implemented with the Intel 3.14 version of CDSA (Common Data Security Architecture) [5]. We have used the CSP (Cryptographic Service Provider) module built upon OpenSSL, and the X.509 and SPKI CL (Certificate Library) modules. We decided to use CDSA since this architecture provides all security services necessary to implement the framework and additionally, this provides integrity services which can be used to ensure component integrity and trusted identification of the component's source.

4 Naming Management System (NMS)

As we mentioned previously, DCMS is composed by two subsystems, NMS and AMS. In this section we are going to present the naming management system, which is responsible for the certification operations related to SPKI ID certificates. This type of certificates can be used to link a name to a particular principal (public key), and also to define group membership. NMS is very useful when authorization is based on group membership. In relation to the scenario presented in Section 2, we can imagine a TICA granting physical access to those principals which are members of group G. NMS can be used by principals in order to obtain an ID certificate for group G, which is issued by a particular naming authority.

Naming is not a requirement of distributed systems, but it is worth noting that large-scale SPKI-based delegation systems can be simplified using this mechanism. Naming is an optional tool for group management which can be useful to address scalability of complex systems.

4.1 Architectural elements

Figure 3 shows the three types of entities involved in NMS: requestors, service access points, and naming authorities. In this section we are going to give a brief description about these core entities, we introduce why they are necessary and how they interoperate.

- **Requestor**. A requestor is a principal demanding the generation of a new ID certificate. This entity must create a certification request and must send it to a particular naming authority (NA) in order to obtain the demanded certificate. This submission can be accomplished using a service access point or making use of an AMBAR connection between the

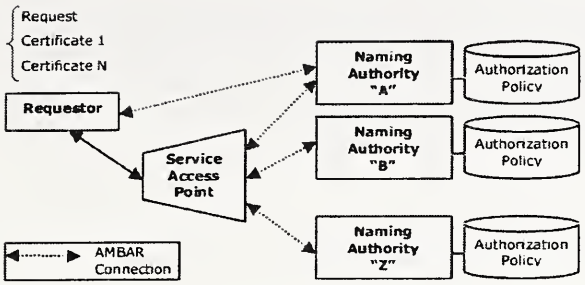


Figure 3: NMS entities

requestor and the NA. Other certificates can be attached to the request in order to demonstrate that the principal has permission to obtain the demanded certificate. There are two types of requestors: first, the principal demanding an ID certificate for a particular public key; second, the principal demanding an ID certificate for a particular name (e.g. a certificate stating that group B is a subgroup of group A). As we will see later, these two situations are managed following different approaches.

- **Service access point**. Requestors can make use of access points in order to submit their certification requests to the appropriate naming authorities. Access points are optional, but they are very useful since they provide several additional services to requestors. First, naming authorities can be hidden from users. Moreover, in some scenarios with many authorities, it might be complicated to know which are the appropriate naming authorities for a particular ID certificate (especially with group membership certificates). SAPs can learn that location information from digitally-signed statements containing information about the system structure and properties. It is simpler to distribute this type of information to few SAPs than to all the principals. Finally, they can provide a certification service to requestors without AMBAR capabilities. Communication between requestors and access points is system-dependent, and it ranges from secure connections to public terminals placed at buildings or departments.

- **Naming authority**. Naming authorities are the certificate issuers. They create ID certificates upon the requests received through the access points or directly from the requestors. NAs are controlled by a particular authorization policy, which can be implemented using SPKI ACLs or other mechanisms. Whenever a NA receives a request and its related certificates, it executes a certificate chain discovery algorithm [6] in order to determine whether the certification request must be granted or denied. Inputs to this algorithm are the request, the additional certificates, and ACL entries. If a certificate chain is discovered, the algorithm returns the information that will be used to generate the new certificate. Communication with NAs are performed using AMBAR. As we have previously mentioned, AMBAR provides functionality to exchange authorization-related information. Using this protocol, entities can be authenticated (identification of requestors is optional), messages are encrypted and authenticated, and some optimization can be performed in order to avoid unnecessary calculations and transmissions (previous messages and authorization decisions can simplify further requests).

4.2 S-expressions for certification requests and ACL entries

Certification requests for ID certificates must contain information about the issuer defining the name, the name itself, the intended subject, and validity dates. Encoding can be based on s-expressions [14] since there is no need for making use of new syntax, and this can simplify the authorization process. Thus, requests might be encoded according to the representation form recommended by SPKI for the *authorization tag* field [7]. However, it is worth noting that the data elements contained in a request are also contained in a SPKI ID certificate, and therefore the structure for this type of certificates can be used. It is not necessary to define a completely new structure in order to express certification requests. Moreover, as we will explain, the same structure can be used by ACLs in order to encode authorization policies. S-expressions that we have used for certification requests and ACL entries have the following format:

```
(cert-request
  (issuer (name NA_i N_i^j))
  (subject P)
  (valid ..)
)
```

- *cert-request*. This identifies the s-expression as a certification request.

- NA_i. This is the public key of the naming authority. This authority is responsible for issuing the ID certificates related to the name N_i^j.
- N_i^j. N^j is one of the names defined in the namespace of the authority NA_i.
- P. This is the principal (or principals) requesting the ID certificate. P might be:
  - A public key.
  - A set of entities. There are two possibilities in order to express a set of entities. On the one hand, we can use a group name, i.e., `(name NA N)`. On the other hand, we can use the *-operator *set*, such as for instance `(* set Q R)`, where Q and R must be public keys or names.
- *valid.* This specifies the requested validity period. The structure of this field is the one included in the SPKI standard.

If this s-expression is used as a certification request, P can only be a public key or a name, and it means that a new ID certificate is being demanded, whose issuer will be NA_i, P will be the subject, N_i^j will be the name linked to P, and will be valid during, at most, the specified validity interval. However, if this s-expression is included in the *tag* field of a SPKI-like ACL entry, it means that the principal (or principals) P are authorized to obtain an ID certificate from NA_i, where the name N_i^j will be linked to P (or each of the principals contained in P) during the specified validity period. Furthermore, N_i^j can make reference to several names when a *(* prefix)* form or a *(* set)* form is used.

Certification requests are encoded as sequences of two elements. The first element is the s-expression specifying the request, and the second one is a digital signature of that sequence. Signatures are encoded using the *signature* structure defined in [7], and they are generated using the requestor's private key. Requests have similar structure to certificates, but certificates are signed by issuers and requests are signed by requestors.

4.3 Some examples

In order to clarify how NMS entities cooperate to generate ID certificates, in this section we are going to analyze two certification requests. First, we explain how a principal can obtain an ID certificate. Then, we will show how subgroups can be defined using ID certificates whose *subject* field also is a name. In these examples, authorization policies are represented by ACLs.

4.3.1 ID certificates for principals

In this first example, P is a principal demanding an ID certificate stating P as a member of group N^j, which is defined by NA_i. P creates the next certification request:

```
(sequence
   (cert-request
     (issuer (name NA_i N_i^j))
     (subject P))
   (signature ..)
)
```

This request is sent to NA_i in order to obtain the demanded certificate. The request will be granted if NA_i can find a certificate chain from its ACL entries to the requestor's public key. The authority contains the next ACL:

```
(acl
  (entry
    (subject (name NA_l N_l^k))
    (tag (cert-request
      (issuer (name NA_i N_i^j))
      (subject (* set P Q R))
    ))
  )
)
```

This ACL specifies that only members of N_l^k can request an ID certificate for N_i^j. If P, Q, or R were members of N_l^k they could request their own certificates. Otherwise, N_l^k can be considered as a relaying party able to make the request. In this case, we will assume that P is a member of N_l^k, and therefore P must send the next ID certificate in order to be authorized:

```
(cert
  (issuer (name NA_l N_l^k))
  (subject P)
)
```

Finally, the naming authority uses the data obtained from the authorization decision in order to create the certificate (signature has been omitted).

```
(cert
  (issuer (name NA_i N_i^j))
  (subject P)
)
```

4.3.2 Subgroups

Subgroups are created using ID certificates whose *subject* field is also a name. This can be useful in order to establish group hierarchies by means of ID certificates. However, it is worth noting that a significant difference exists between generation of subgroups and creation of ID certificates for public keys. Generation of ID certificates is normally requested by the principals involved, but subgroup certificates cannot be requested by the subgroup itself. Authorized requestors are policy-dependent, but some appropriate candidates are the naming authority defining the subgroup, or even a subgroup member. In this example, the authorized requestor is the naming authority, but this has delegated the authorization to principal R in order to avoid signing certification requests with the same private key used to generate ID certificates.

This is the request sent by R to NA_i in order to define N_l^k as subgroup of N_i^j (it is signed using the private key of R):

```
(sequence
   (cert-request
     (issuer (name NA_i N_i^j))
     (subject (name NA_l N_l^k)))
   (signature ..)
)
```

Next ACL specifies that NA_l can request an ID certificate for N_i^j, and can also delegate that permission.

```
(acl
  (entry
    (subject NA_l)
    (propagate)
    (tag (cert-request
      (issuer (name NA_i N_i^j))
      (subject (name NA_l N_l^k))
    ))
  )
)
```

R also sends the next authorization certificate in order to demonstrate that NA_l delegated the permission to R:

```
(cert
  (issuer NA_l)
  (subject R)
  (tag (cert-request *))
)
```

Finally, NA_i uses the data obtained from the authorization decision in order to create the certificate.

```
(cert
  (issuer (name NA_i N_i^j))
  (subject (name NA_l N_l^k))
)
```

5 Authorization Management System (AMS)

Section 2 shown a scenario where authorization certificates can be used in order to gain physical access to buildings. The system was based on delegation, and users obtained this type of certificates from trusted authorization authorities. In this section we are going to present the authorization management system, which is responsible for certification operations related to SPKI authorization and attribute certificates.

5.1 Architectural elements

NMS and AMS are based on similar architectural elements. Requestors and access points are also part of AMS. Naming authorities are replaced by authorization authorities (AA), but they share some basic functionality. AAs create attribute and authorization certificates upon the requests received through the access points or directly from the requestors.

An AMS requestor is a principal demanding the generation of a new attribute or authorization certificate. This entity must create a certification request containing information about the authorization tag (the tag is completely application-dependent). Like

in NMS, there also are two types of requestors: first, the principal requesting an authorization certificate; second, the principal requesting an attribute certificate for a particular name. As we will see later, we consider that these two situations should be managed following different approaches.

5.2 S-expressions for certification requests and ACL entries

S-expressions used in AMS to specify certification requests are also based on the structure defined by SPKI for attribute and authorization certificates. The main difference between NMS and AMS s-expressions is the *tag* field. This field contains information about the particular authorization being requested (when it is contained in a certification request) or granted (when it is part of an ACL entry).

Certification requests are also encoded as sequences composed by the request itself, and its signature.

5.3 Some examples

In order to clarify how AMS entities cooperate to generate authorization and attribute certificates, in this section we are going to analyze two certification requests. First, we explain how a principal can obtain an authorization certificate. Then, we will show how attribute certificates can be generated. In these examples, authorization policies are also represented by ACLs.

5.3.1 Authorization certificates

In this first example, P is a principal demanding an authorization certificate containing a tag tag^A from authority AA_i . Next certification request is created by P:

```
(sequence
   (cert-request
     (issuer AA_i)
     (subject P)
     (tag tag^A))
   (signature ..)
)
```

This request is sent to AA_i in order to obtain the demanded certificate. The request will be granted if AA_i can find a certificate chain from its ACL entries to the requestor's public key. The authority contains the next ACL:

```
(acl
  (entry
    (subject P)
    (tag (cert-request
      (issuer AA_i)
      (subject P)
      (tag tag^B)
    ))
  )
)
```

This ACL specifies that P can request an authorization certificate containing the permission specified by tag^B (tag^A must be more restrictive or equal to tag^B). Finally, the authorization authority uses the data obtained from the authorization decision in order to create the requested certificate.

```
(cert
  (issuer AA_i)
  (subject P)
  (tag tag^A)
)
```

One of the main advantages of this proposal is that it is possible to specify a class of certificates, possibly infinite in size, without having to issue them all. The appropriate finite subset of that class can be issued on demand. The potential infinite size of the class comes from use of **-forms*.

5.3.2 Attribute certificates

Attribute certificates can be used to specify roles. The subject can be a name defining a role, and this type of certificate states the permission related to that role. Roles can be seen as various job functions in an organization, and users can be assigned to one role depending on their responsibilities. The role permissions use to be stable since roles activities do not change frequently. However, we must answer the question: "Who must the requestor of an attribute certificate be?"

Certificates are issued by authorization authorities, hence valid requestors are those specified by their authorization policies. AMS should keep inherent

policies to a minimum, in order to allow users of the system to design their own authorization policies. Therefore, valid requestors can range from role members to specific role managers. Nevertheless, we find the latter approach very interesting for complex systems since role management can be greatly simplified using specific administrators (role managers). Authorities can authorize role managers to request attribute certificates for a particular set of group names. This authorization can be expressed as:

$$AA_i \Rightarrow RM_i^1(N_l^k, N_f^g), RM_i^n(N_j^h)$$

This expression denotes that authority AA_i authorizes role manager RM^1 to request attribute certificates for the group N^k defined by NA_l, and for the group N^g defined by NA_f. AA_i also authorizes RM^n to request this type of certificates for the group N^h defined by NA_j.

We are going to see how this relation can be implemented using AMS. In this example, RM_i^1 requests an attribute certificate for N_f^g, with the authorization tag tag^A. This is the request sent by RM_i^1 to AA_i (it is signed using the private key of RM_i^1):

```
(sequence
   (cert-request
     (issuer AA_i)
     (subject (name NA_f N_f^g))
     (tag tag^A))
   (signature ..)
)
```

The authority contains an ACL implementing the above-expressed relation. This is the ACL:

```
(acl
  (entry
    (subject RM_i^1)
    (tag (cert-request
      (issuer AA_i)
      (subject (* set
        (name NA_l N_l^k)
        (name NA_f N_f^g)))
      (tag tag^B)))
  )
  (entry
    (subject RM_i^n)
    (tag (cert-request
      (issuer AA_i)
      (subject (name NA_j N_j^h))
      (tag tag^C)))
  )
)
```

Finally, the authorization authority uses the data obtained from the authorization decision in order to create the requested certificate.

```
(cert
  (issuer AA_i)
  (subject (name NA_f N_f^g))
  (tag tag^A)
)
```

6 Use of AMBAR in DCMS

Requests and certificates are exchanged using AMBAR connections. Although other protocols like SSL (Secure Socket Layer) can be used for this purpose, we find AMBAR a valuable approach since it has been designed to exchange authorization-related information. Entities making use of AMBAR do not pay attention to issues such as the encapsulation of requests or certificates. They create AMBAR connections in order to exchange this type of information, and AMBAR modules are responsible for encapsulation and protection. Furthermore, this protocol has been designed to be session-oriented in order to optimize those scenarios where the request/response messages are exchanged between the same client and server (such as for instance, access points and authorities).

In DCMS, there are two types entities which must make use of AMBAR: access points and authorities. Requestors can request their certificates using access points, and therefore AMBAR functionality is not a requirement for them. Authorities should not employ their private keys to establish AMBAR connections since it is not suitable to protect their communications making use of the same private key signing the certificates. Authorities should generate a new key pair for communication purposes, and they should issue a certificate authorizing the new key pair to act as their *network interface*. This certificate should include a tag `(tag dcms-com)`, and will be used by access points and requestors to validate that they are indeed exchanging information with the right authority.

AMBAR connections used in DCMS can perform authentication based on X.509 certificates, or SPKI certificates. Access points and authorities are always authenticated, but identity of requestors can be preserved using the anonymous mode. Credentials (additional certificates attached to the request) can be provided by access points or requestors (push method), or can be recovered from public repositories by authorities (pull method). Figure 4 shows an exchange (push) between an access point and an authorization authority, and how data are encapsulated in AMBAR messages. If the certification request is granted, the authority sends a *Resource* message containing the certificate. Otherwise, a *Negative Notification* message is generated. Negotiation is performed only once. Then, requests and results are exchanged using the previously-established channel.

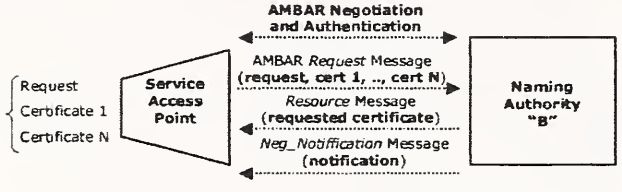


Figure 4: Communication between an access point and an authority

7 Conclusions

In this paper, we have proposed a management system that can be used in SPKI scenarios based on delegation. We present how certification requests for ID, attribute, and authorization certificates can be expressed, how authorization policies can be enforced in a distributed way, and which are the entities involved in a certification scenario.

We consider that our system provides strong mechanisms to address scalability-related problems. First, we have tried to keep inherent policies to a minimum, in order to allow system administrators to design their own authorization policies. What is more, following our approach, it is possible to specify a set of certificates without having to issue them all since they are issued on demand. Added to this, we make a clear distinction between requestors and subjects of certificates. We do not force both entities to be the same one, enabling therefore the participation of relying parties.

In order to complete our proposal, additional mechanisms must be designed, such as certificate revocation or certificate storage. Currently, we are also developing a new service of DCMS for automatic reduction of certification chains. Certificate reduction can be used to improve performance of authorization decisions and, as is commented in [1], to provide anonymity services.

8 Acknowledgements

This work is partially supported by TIC2000-0198-P4-04 project (ISAIAS), and by IST-2001-32161 project (Euro6ix)

References

[1] T. Aura and C. Ellison. Privacy and Accountability in Certificate Systems. Technical Report HUT-TCS-A61, Helsinki University of Technology, 2000.

[2] O. Canovas and A.F. Gomez. AMBAR: Access Management Based on Authorization Reduction. In *Proceedings of the International Conference on Information and Communications security (ICICS 2001)*, volume 2229 of *Lecture Notes in Computer Science*, pages 376–380. Springer Verlag, November 2001.

[3] O. Canovas, A.F. Gomez, H. Martinez, and G. Martinez. A Role-Based Implementation of Physical Access Control using Authorization Certificates. Technical Report UM-DITEC-2002-2, Department of Computer Engineering, University of Murcia, January 2002.

[4] Dwaine Clarke. SPKI/SDSI HTTP Server and Certificate Chain Discovery in SPKI/SDSI . Master's thesis, M.I.T., September 2001.

[5] Intel Corporation. *Common Data Security Architecture (CDSA).* World Wide Web, http://developer.intel.com/ial/security, 2001.

[6] J.E. Elien. Certificate discovery using SPKI/SDSI 2.0 certificates. Master's thesis, Massachusetts Institue of Technology, May 1998.

[7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *Simple Public Key Certificate.* IETF Internet Draft, draft-ietf-spki-cert-structure-06.txt edition, July 1999.

[8] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate*

theory, September 1999. Request For Comments (RFC) 2693.

[9] T. Hasu and Y. Kortesniemi. *Implementing an SPKI Certificate Repository within the DNS*, Poster Paper Collection of the Theory and Practice in Public Key Cryptography (PKC 200) edition, January 2000.

[10] J. Koponen, P. Nikander, J. Rasanen, and J. Paajarvi. Internet access through WLAN with XML encoded SPKI certificates. In *Proceedings of NordSec'00*, October 2000.

[11] Y. Kortesniemi, T. Hasu, and J. Sars. A Revocation, Validation and Authentication Protocol for SPKI Based Delegation Systems. In *Proceedings of Network and Distributed System Security Symposium (NDSS 2000)*, February 2000.

[12] T. Lampinen. Using SPKI Certificates for Authorization in CORBA based Distributed Object-Oriented Systems. In *Proceedings of NordSec'99*, pages 61–81, November 1999.

[13] Per Harald Myrvang. *An Infrastructure for Authentication, Authorization and Delegation.* PhD thesis, Department of Computer Science, University of Tromso, May 2000.

[14] R. Rivest and B. Lampson. *SDSI: A simple distributed security infrastructure.*

[15] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2), February 1996.

[16] K. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Proceedings of Network and Distributed System Security Symposium*, April 2001.

Scalability Issues in PMI Delegation

Scott Knight
Royal Military College of Canada
knight-s@rmc.ca

Chris Grandy
NDHQ/Directorate of National Information Systems (DNIS)
grandy.cc@forces.ca

Abstract

The Canadian Department of National Defence (DND) is shifting its methods for the delegation and exercise of authority from paper-based to electronic-based means. DND has deployed a commercial PKI but there is no general technical solution presently employed by DND for access control or electronic authorization of workflow in distributed processing environments. The aim of this research is to show how an authorization system, or privilege management infrastructure (PMI), can be used to support business processes DND. The results are expected to be applicable to large enterprises in general.

The research demonstrates how ITU-T standard X.509 can be used to support DND authority and delegation models. The investigation involves the analysis of the key authorizations within a specific DND problem domain. The X.509 standard and concepts from role-based access control form the basis of the PMI design. This involves the use of attribute certificates to control the specification and delegation of privileges. A novel interpretation of X.509 attribute certificates is proposed that provides separate hierarchies of responsibility for the management and delegation of roles. The results provide insight into, and quantification of, the complexity of the resulting delegation chains. The use of a roles based model for delegation is seen as being important to the scaling of PMI to service large enterprises with mature, complex authority structures. If the processing complexity can be managed, the flexibility of being able to model the actual privilege delegation paths in an organization is an advantage of a role-based model.

1. Introduction

Public-key infrastructure (PKI) has matured into a commercially supported, deployable technology. With a high degree of assurance current PKI products offer secure, reliable security services to support identification, authentication, confidentiality, and non-repudiation. These are powerful services but the adoption of PKI in enterprise environments has been slow. It is the opinion of the authors that wider proliferation of PKI will come with the ability to provide effective support for authority structures within an enterprise. The authority structures within an enterprise govern business process. Every legitimate task is performed under the approval of some authority that has ultimate responsibility for that part of the business process. In many cases there is a requirement that the entities performing a task must have the appropriate approval, or privilege, to do so. An attribute-certificate based privilege management infrastructure (PMI) is a mechanism that can be used to support enterprise authority structures. Attribute-certificate based PMI is an aspect of PKI and requires underlying services for the management of public-key certificates (PKCs). To this extent the proliferation of PMI can lead to more wide spread adoption of PKI. Although there are standards that define PMI services [X.509], and some commercial products that provide support, there is little attention in the literature paid to the issues of scalability in an enterprise environment. Also, there do not seem to be examples of attribute-certificate deployment models to support business process. This work examines these issues by proposing a PMI model to support authority structures in the Canadian Department of National Defence (DND).

DND has deployed a commercial PKI to be used to support the Government of Canada policy on Electronic Authorization and Authentication [Gov96]. The PKI is intended to support a variety of new systems and legacy systems, and to provide a unified mechanism for managing task authorization.

An attribute-certificate based PMI model is used to explore the complexity of the certificate chains that need to be verified when exercising privilege. The resulting certificate chains are quite complex and some chain pre-processing strategies are discussed to reduce the real-time privilege verification overhead.

This work is an extension of [Gra01]. Although the model discussed here pertains to DND it is believed that the work is relevant in a broader context and reflects authority structure and business process issues in large organizations in general. The rest of the paper is organized as follows. Section 2 reviews the significance of privilege management in the context of supporting an organization's security policy. An overview of privilege management within DND is presented in Section 3 to provide the context for the development of a role-based authorization model. Complexity issues arising from the model are discussed in Section 4. Finally, Section 5 concludes the paper and discusses further work.

2 Support Mechanisms for a Security Policy

In defining security policy the classic literature defines three security properties: confidentiality, integrity and availability. The security policy defines the access privileges a specified set of subjects have for objects in the system. The objects are the information resources that are protected by the system. In an information system the security policy is realized by implementing security mechanisms such as identification and authentication (I&A), access control, audit. Through the use of public-key certificates a PKI system can provide strong I&A support for a system. This mechanism provides good assurance of the true identity of the subjects. In most business systems there must be a determination of what kinds of access are permitted to the system objects. Currently access control and the format of the authorization database is application specific (stovepipes) and there is no unified way to deal with permission. A standard mechanism for the support of access control decisions can provide more complete support for security policy at the enterprise level. This support can be provided by attribute-certificate based PMI and the development of such mechanisms may lead to greater proliferation of PKI in general.

The authority structures in a specific enterprise environment have evolved over a period of time and represent efficiencies in the command and control of the organization. This is the case with the DND case study being examined. It seems reasonable to expect that the PMI would support the organization's authority structures and business process, and not expect that the organization would have to make large changes to its authority structures and business process to adapt to the PMI mechanisms.

2.1 Attribute-certificates

X.509 public-key certificates have some support for privilege management through the use of subject attributes. However in the following cases it is recommended that attribute-certificates are the more suitable mechanism [X509]:

a) a different entity is responsible for assigning particular privilege to a holder than for issuing PKCs;

b) there are a number of privilege attributes to be assigned to a holder, from a variety of authorities;

c) the lifetime of a privilege differs from that of the holder's PKC validity;

d) the privilege is valid only during certain intervals of time which are asynchronous with that user's PKC validity or validity of other privileges; or

e) delegation of authority is permitted, and for any specific delegation there may be differences in the kind of privilege that the delegating authority passes down to the delegated authority.

All these conditions are true in the case of the DND example. In complex inter and intra-organizational relationships, it makes more sense to manage authentication separately. It is reasonable to expect that PKC authorities will not have jurisdiction over privileges that are solely the domain of the process owner. One would expect this will become the rule rather than the exception as the market encourages the emergence of commercial CA services and PKI outsourcing providers [WH99].

It is also the case that the authority structures of the example environment have evolved to be heavily role-based. For example, a member of the Canadian Armed Forces normally has a career spanning decades. Personal identification information is static for long periods during this time. The member may serve in a number of different roles (concurrently and overlapping). The privileges associated with the roles may be defined and modified by different agencies than

those assigning the member to the role. It is expected that this is not unique to the example, and that there are a large number of enterprise environments where these conditions hold. The X.509 standard provides a mechanism for managing roles. This seems to be a natural mechanism to be used to model the required authority structures. The standard warns that the "use of roles within an authorization framework can increase the complexity of path processing." There is no indication in the standard of how complex the path processing can become, how the model will scale to larger organizations, or how the role delegation paths will effect the performance of privilege verifiers.

There are several factors that make X.509 attribute certificates (ACs) an attractive option for managing privileges. An X.509 AC can be managed in the same way as the X.509 PKC. ACs can also be digitally signed like PKCs. This authenticates the attributes and provides integrity protection so that the certificates cannot be modified. ACs are generalizations of identity certificates, PKCs (an identifier through the use of a public-key is just one of many possible attributes), and have naturally evolved from them [Bra00]. ACs are digital certificates that serve primarily to enable verifiers to establish attributes other than the identity of the key holder (such as access rights, authorities, adherence to standards, legal requirements, privileges, permissions, capabilities, preferences, assets, demographic information, and policy specifications). An authorization service, PMI, can be designed using attribute certificates which each point to a PKC. More comprehensively, a PMI includes people, policies, hardware and software interacting together to bind privileges to a user by issuing him attribute certificates [Ada99].

Because a PMI depends on the authentication provided by a PKI, a PKI must be available before a PMI can be implemented. Since ACs do not provide authentication,one cannot assign privileges to a user using attribute certificates if that user does not have at least one associated PKC.

The standard specifies that a privilege holder must present an attribute-certificate (AC) containing the appropriate attributes/privileges to a privilege verifier before access is granted to an information object (i.e. the privilege holder asserts a privilege). The privilege verifier acts as a reference monitor and controls access to the object. The decision to allow access is based on the security policy being enforced by the verifier and any applicable environment variables (e.g. time of day).

2.2 Delegation

Delegation is the conveyance of privilege, from one entity that holds such privilege to another entity. The model consists of four components: the source of authority (SOA), the attribute authority (AA), the privilege holder and the privilege verifier.

The SOA occupies the highest position in the authority hierarchy. Within a PMI, the source of authority (SOA) is analogous to the root CA in hierarchical PKIs. It is different in that there may be many sources of authority (one for each privilege or set of privileges) whereas there is only one root CA in a strictly hierarchical PKI. The SOA is the issuer of certificates that assign privileges to privilege holders and is present even in

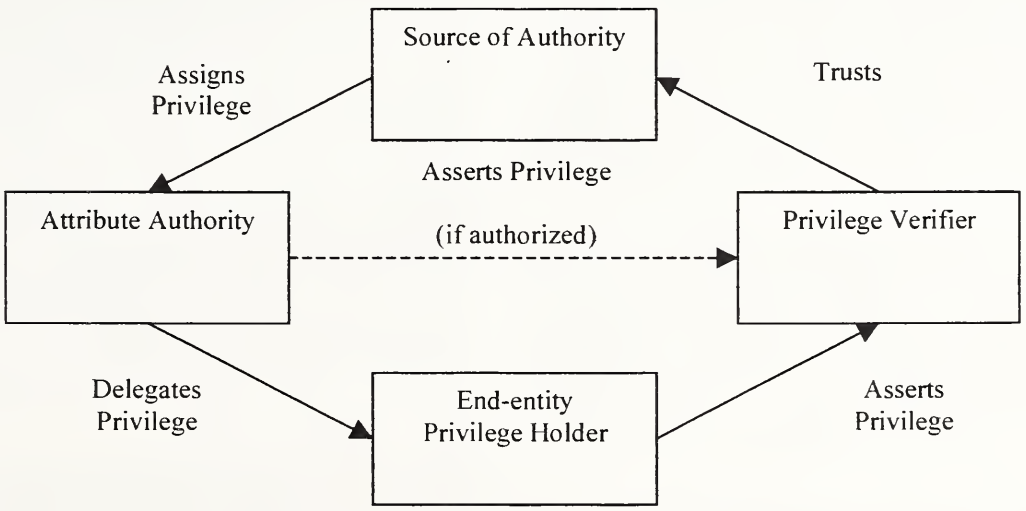


Figure 1 - The Delegation Model [Int00]

environments where delegation does not occur.

In Figure 1, the SOA authorizes an entity to act as an AA by assigning it a privilege and the authority to delegate that privilege. The AA further delegates that privilege to other AA's or end entities through the issuance of certificates that contain the same privilege (or a subset thereof). The AA is analogous to subordinate CAs within a PKI, but a CA issues public-key certificates whereas an AA issues attribute certificates. All entities that issue and obtain attribute certificates need to be authenticated; therefore, they will each require their own PKC. This means AAs will also require PKCs. Each of the intermediary AAs may, in certificates that it issues to further privilege holders, authorize further delegation by those holders also acting as AAs. The SOA may impose constraints on the re-delegation of a privilege. A delegator can also further restrict the ability of downstream AAs to delegate [Int00]. A universal restriction on delegation, known as the domination rule, is that no AA can delegate more privilege than it holds [Int00].

The privilege verifier trusts the SOA as the authority for a given set of privileges for the resource. Also, when delegation is used, the privilege verifier trusts the SOA to delegate some or all of those privileges to other holders. If the privilege asserter's certificate is not issued by the SOA, then the privilege verifier must locate a delegation path of certificates from that privilege asserter to the SOA. The validation of that delegation path must include checking that each AA had sufficient privileges and was duly authorized to delegate those privileges.

Processing an attribute certificate path in PMI is analogous to processing other certificate paths within a PKI. Validation is conducted with respect to attribute authorities rather than certification authorities, and the information pertains to privileges rather than identity. However, with privilege path processing, the processing engine will need to consider elements of both the PMI and the PKI in the course of determining the ultimate validity of a privilege asserter's attribute certificate. With respect to PKI, the privilege verifier must verify the identity of every entity in the path using the certification path processing procedure identified in the X.509 standard [Int00]. For example, a referenced public-key must be checked for its validity before the digital signature on an attribute certificate can be verified.

Privilege path processing relies on the elements of PMI to establish a valid delegation path. The central requirement is to ensure that each entity in the path has the authority to delegate privileges to the entity below. The delegation path is distinct from the certificate validation path used to validate the public-key certificates of the entities involved in the delegation process. The attribute certificates within the path must still be digitally signed by the corresponding authority. The delegation path represents a chain of trust between the privilege asserter and the SOA.

Figure 2 provides a general illustration of the privilege processing checks used to establish a chain of trust back to the SOA. The privilege verifier is presented with an AC, EE-AC, belonging to an end-entity, EE. EE-AC might pertain to access to some resource. In order to verify that EE has legitimate possession of EE-AC the verifier must verify the signature on the certificate to ensure it actually was created by the issuer named on the certificate. In this case the issuer is AA1. To ensure that AA1 legitimately holds the relevant privilege the verifier must retrieve the AC that is owned by AA1. AA1-AC is the certificate that allocates privilege to AA1; it is issued by AA2. AA1-AC must also have its signature verified. AA2 may or may not be directly trusted by the privilege verifier for the required attributes. If not, the privilege verifier may have to retrieve another AC (e.g. AA2-AC) until it finds one issued by a directly trusted AC issuer (SOA) for that privilege.

Once a valid chain has been confirmed, the privileges contained in that attribute certificate may be used to make an access control decision. The attributes are compared with the relevant privilege policy and other information associated with the context in which the certificate is being used. It must be determined if the privilege holder actually intended to assert the contained privileges for use with that context. The fact that a chain of certificates to a trusted SOA exists is not enough. The willingness of the privilege holder to use that certificate has to be clearly indicated and verified. The standard does not specify this application-dependent mechanism.

The issue of certificate revocation complicates this process. For the purposes of this paper we will consider certificates to be short lived and the use of certificate revocation lists will not be required. A more complete treatment of this issue and the formats for the attribute certificates can be found in [Gra01].

2.3 Roles

Roles provide a means to indirectly assign privileges to entities. Providing access control based on the entity's functional role as opposed to its personal identity is a powerful concept known as Role-based Access Control

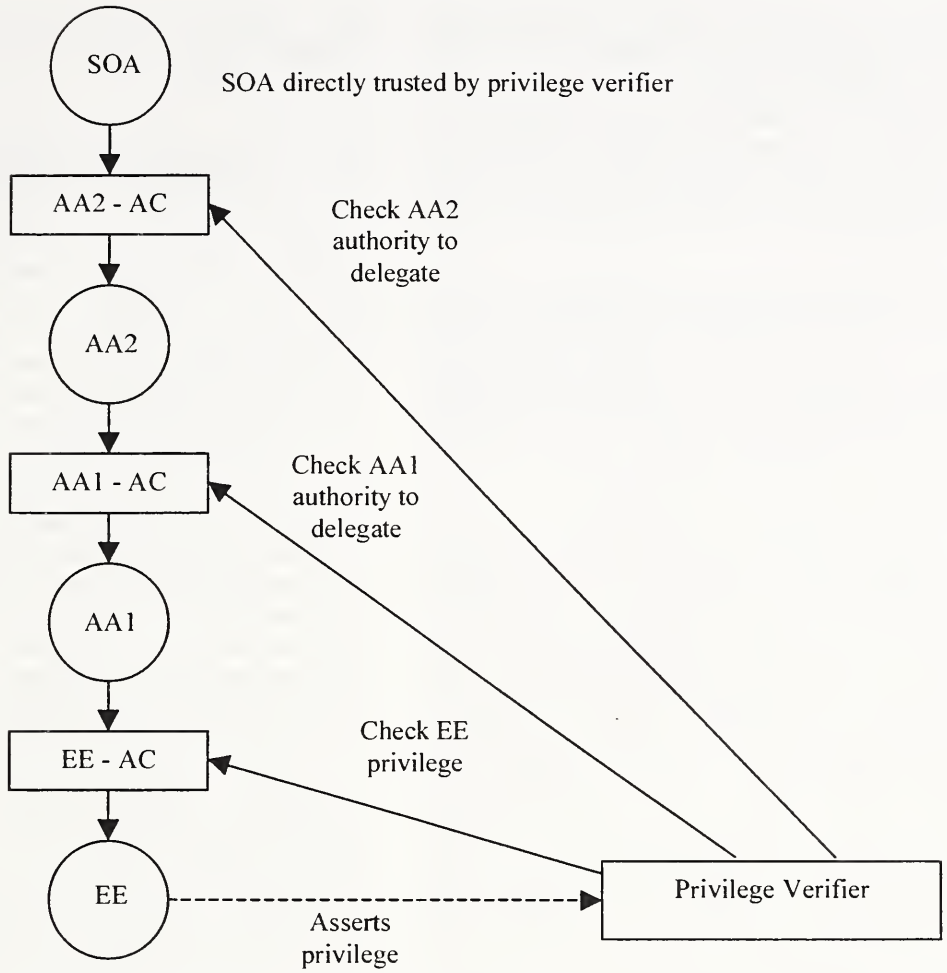


Figure 2 - Chaining attribute certificates

(RBAC). RBAC is a useful approach because it can reflect the authority structures within an enterprise. The basic role model described in the X.509 standard consists of two types of ACs. Specific privileges associated with a particular role are specified within Role Specification Certificates (RSCs). Entities are assigned to the role (specified by the RSC) via another attribute certificate called a Role Assignment Certificate (RAC). The de-coupling of privilege assignment to roles, from the role assignment to individuals allows privileges to be updated without affecting the assignment of the roles.

3 Authorizations in the Problem Domain

3.1 A Procurement Example

Consider a familiar business transaction. Suppose a customer on a Canadian Forces Base needs to procure a personal computer. This computer may be required because of an operational requirement and it will be connected to the Defence Wide Area Network (DWAN). This particular example is chosen for a number of reasons. Many readers will relate to this example. More importantly, the procurement requires the delegation of authority and the cooperation of several different roles.

Specific authorities and responsibilities for the control and spending of funds appropriated by Parliament for DND are conferred on the Minister of National Defence (MND) by the Financial Administration Act (FAA) and the National Defence Act (NDA). Since the MND cannot carry out these responsibilities personally, it is necessary for him to authorize officials to exercise these authorities on his behalf.

The MND is required to ensure that separate organizations or individuals are invested with spending authority and the complimentary, but completely

distinct, payment authority. This is a standard business practice for fraud protection. This requires at least two distinct delegation paths to ensure the proper separation-of-duty. Additionally the computer is required to be connected to the DWAN. This requires the approval of a network technical authority that derives its privilege from a completely separate delegation path.

As an example of delegation, the Responsibility Centre (RC) Manager plays a central role exercising spending authority. An RC Manager is anyone (military or civilian) who manages a distinct unit or organization, prepares and controls a budget, and has spending authority for his/her budget [Dep99].

It is possible to summarize a process model for this procurement process.[1] The customer, acting in the role of RC Manager, will normally recognize the need for the purchase. In this case, the requirement is for a computer. The Base Telecommunications and Information Services Officer (BTISO) role will take responsibility for specifying and describing the technical aspects of this need. The next five steps are usually performed by the section belonging to the Integrated Logistics Officer (ILogO role) based on the input from the customer and the BTISO: determining sourcing options; establishing price and terms; preparing and placing a purchase order; and following up on the order. The vendor receives the order and ships the computer along with an invoice. The customer, in his role as the RC Manager, receives the computer and confirms it matches the requirement. He then approves the invoice and submits the transaction for review to another role, the Financial Officer, who authorizes the release of funds to the vendor.

Other layers of delegation are possible. For example, the BTISO would likely delegate this authority to review and approve technical requirements to a subordinate such as the Network Maintenance Officer (NMO).

The processing of this procurement will require that the individuals filling the various roles have access to the necessary functions of the procurement system software (a legacy system). Their access must be authorized. Their decisions must enable the respective business process function and can not be repudiable. An interesting observation is that the entire transaction can be viewed as series of authentications and authorizations.

3.2 Mapping the Requirement to Attribute Certificates

The interaction of users in the various roles in the previous section suggests that role-based access control can have tremendous relevance in establishing electronic authorization for business process. RBAC takes the approach that authorizations are distributed according to role rather than identity. The process model clearly revealed that roles can be effectively used to conduct a local procurement transaction.

The style of RBAC proposed by the X.509 roles model, and summarized in section 2.3, can be applied to this procurement example. Individuals could be assigned a role assignment certificate matching one of the procurement roles e.g. BTISO, ILogO, NMO. These role assignment certificates could point to a corresponding role specification certificate containing the key authorizations, or privileges.

A complete design in support of this procurement example will not be described here. The intent here is to demonstrate the application of the X.509 standard to this problem, and not to stipulate all the details of a specific design. The portions of the design described in this work are sufficient to support the modeling scheme. Addressing every role in the process is not only time-consuming, but also unnecessarily repetitious. As much insight can be gained about the specification, assignment and delegation of privileges by investigating one role as by examining them all. Therefore, only the BTISO role will be explored in detail. The technique is completely analogous for the other roles, such as the ILogO and the Finance Officer.

3.2.1 Extending the X.509 Roles Model

The BTISO typically requires more privileges than just those needed to participate in a local procurement transaction. He would also likely be the COMSEC Custodian for the Base Crypto Account, the local configuration authority for connections to the DWAN, and, like many other managers (such as the customer in this procurement example), an RC Manager responsible for his own budget. While the details of these privileges are unimportant here, it is likely that the privileges associated with these other duties originate from different sources of authority. Unfortunately, the X.509 standard offers no direct guidance for dealing with complex roles

[1] A complete process model for the procurement was completed and is available at [Gra01].

The design in this paper employs a novel interpretation of the roles model described within the X.509 standard. The standard suggests using the role attribute within a role assignment certificate to point to a single role specification certificate where all the privileges are held. The new interpretation builds upon this idea by proposing that the role specification certificate can itself contain role attributes, each pointing to another role specification certificate.

Convenience was considered important in this design. Otherwise, the attraction of using a certificate-based PMI would fade for those wishing to apply it to complex organizations and roles. The BTISO role in the procurement example is quite common in DND; many of the privileges and responsibilities associated with the role are not unique to a particular Base. The same is true for the other positions. It would be convenient if the same role design could be reused wherever a BTISO position exists. DND is an dynamic organization that demands managers to adapt to unfamiliar work environments in short periods of time. It may be asking too much to expect an infantry Colonel, newly appointed as a Base Commander, to understand PMI and all the privileges required of his BTISO. Sending him on a "shopping trip" for privileges at the various SOAs, besides wasting time, will likely yield incomplete and unsatisfactory results. Convenience, therefore, also suggests that a Base Commander should be able to appoint someone to a position, such as a BTISO, by simply issuing him a single role assignment certificate.

Think of the BTISO role as a super-role encompassing the privileges held by a BTISO. Smaller, more specific roles, such as COMSEC Custodian, DWAN Configuration Control Officer and RC Manager, can be thought of as sub-roles comprising the super-role.

Viewing complex roles in this way offers several advantages. The most obvious convenience is that it allows complex roles, or super-roles, to be quickly and easily constructed by simply combining more elementary roles. Designers of the role specification certificate for the super-role can quickly gather many of the necessary privileges by inserting pointers to role specification certificates for the sub-roles.

Reuse is another observable benefit. The number of attributes that have to be developed exclusively for the role of BTISO can be minimized since many of the necessary attributes already exist within the recognized sub-roles. Of course, this can be a double-edged sword. Each role will have to be carefully inspected to ensure that a super-role does not inherit privileges that are part of the sub-role, such that the super-role acquires privileges it is not entitled to. Nonetheless, a single role specification certificate can be reused by several super-roles. The BTISO needs spending authority, but so does the ILogO, the customer and many others across DND. Somewhere in the hierarchies below these roles the same generic set of spending privileges (identified by the sub-role of RC Manager) could be referenced.

Finally, in keeping with the intent of the X.509 standard, many of the updates to complex super-roles would be made automatically. Every change to a role specification certificate will percolate upwards to modify the capabilities of any role specification certificate above it in the hierarchy. This effect will be most pronounced whenever there are changes at the bottom of the hierarchy. For example, any change in the privileges associated with the role specification certificate for RC Manager will automatically update the capability of any super-role which references it, e.g. the BTISO, the ILogO etc. Although designers of role specification certificates higher in the hierarchy will have to monitor the effects of these changes on the super-roles, the outcome should be to generally increase their currency and relevance since the changes are being effected by the source of authority for a particular privilege.

The bottom of the hierarchy would consist completely of privileges that could not be decomposed any further. These privileges would be contained within atomic role specification certificates, such as RC Manager. These atomic certificates contain privileges that naturally go together; it would make no sense to split them any further. It is likely that a large number of these atomic role specification certificates will be re-used as sub-roles within many other super-roles. These atomic role specification certificates are a natural development since, in all probability, a single source of authority will be responsible for various privileges that are closely related. For instance, all spending privileges, including those associated with the role of RC Manager, are controlled by the same source of authority, the MND. These spending authorities (described earlier) are designed to complement each other. Rather than assign them individually, it would be practical to group these complementary privileges together in role specification certificates, such as for the role of RC Manager.

3.2.2 Delegation Chains for the Validation of Role Specification

The specification and maintenance of these roles, used across DND, would be a centralized function of the National Defence Headquarters (NDHQ). In this way role specifications are produced and maintained by

people and organizations that understand the PMI and the interaction of privilege. The SOA, in this case the MND, would set up the required atomic certificates and delegate the responsibility for the creation and maintenance of complex roles for various parts of the business process to staff officers. They can be thought of as role managers. They produce ready-to-use role specifications (probably complex roles) that can be used by field officers to assign people to roles in their organizations. The ability to access a step of the business process must include verifying the delegation chain from the required attribute/permission on an atomic certificate, through more complex role specification certificates, to the author of the RSCs (an AA that must have the right to delegate the privilege), and through any superior role-specification AAs back to the SOA. The validation of this chain ensures that the privilege is being exercised through an authorized role, and that the creators of that role had the right to delegate the privilege to the role.

3.2.3 Delegation Chains for the Validation of Role Assignment

The delegation of authority to individuals has a separate delegation chain tracing back to the SOA (in this case the MND). The delegation of authority to individuals is made by issuing role assignment certificates.

The MND delegates authority for the Canadian Armed Forces to the Chief of the Defence Staff. The Chief of the Defence Staff delegates authority for large formations of the military to superior commanders who in turn delegate authority for smaller units to commanding officers. These delegations are made by using the ready-to-use roles, which are prepared by the centralized RSC managers in NDHQ. The commanders do not have to, and do not want to, understand the specification and maintenance of the ready-to-use RSCs.

The ability to access a step of the business process must include verifying the delegation chain from the required attribute/permission on an atomic certificate, through more complex role specification certificates, to the commander assigning the role to an individual (an AA that must have the right to delegate the privilege), and through any superior commander AAs back to the SOA. The validation of this chain ensures that the privilege is being exercised through an authorized role, and that the chain of commanders assigning that role to the user both possess the privilege and had the right to delegate the privilege to individuals down the chain of command.

Figure 3 provides a graphical representation of these dual delegation chains. It is assumed in the figure that the AAs have the necessary privileges to delegate; the diagram has been simplified and certificates associated with this are not shown. The role specification validation chain extends from the BTISO RSC back through the manager for the BISTO role to the SOA. The role assignment validation chain extends from the BTISO RSC back through the Base Commander to the SOA.

4 Delegation Path Complexity

When an end entity tries to access a controlled object the privilege verifier protecting that object must ensure the end entity is in valid possession of the privilege/security attribute required by the security policy to allow access. This will require the privilege verifier to walk the certificate chains to ensure the chain of trust is not broken between the SOA and the user of the attribute. For each certificate, the verifier will have to ensure the certificate is properly signed (a public-key operation), and that some required attribute(s) exist in the certificate. The public-key certificate operations dominate the complexity, and attribute checks can be ignored.

The diagram in Figure 3 has been simplified. In the general case there may be a number of RSCs in a chain that describes the role hierarchy from the complex role the end user is using, to less complex roles, and finally to atomic roles. Each of the RSC certificates in the hierarchy would have a role specification validation chain rooted at the SOA (section 3.2.2). Each specification validation chain might include more than one role manager (i.e. the role management might be delegated down the chain). Each chain must be validated.

A superior commander assigns the role to the end entity by issuing a RAC. The role assignment validation chain extending back through commanders to the SOA must also be validated. But at each step back through this chain the commanders' own privileges were assigned to them through their own role RACs. So the certificate chains of each commander's role must also be walked.

Consider the following simplifying assumptions.

a) There is a simple CA; the verifier has access to a trust root certificate for the CA that it can use to verify any PKC. Therefore certificate validation requires two public-key operations: signature verification of the attribute certificate

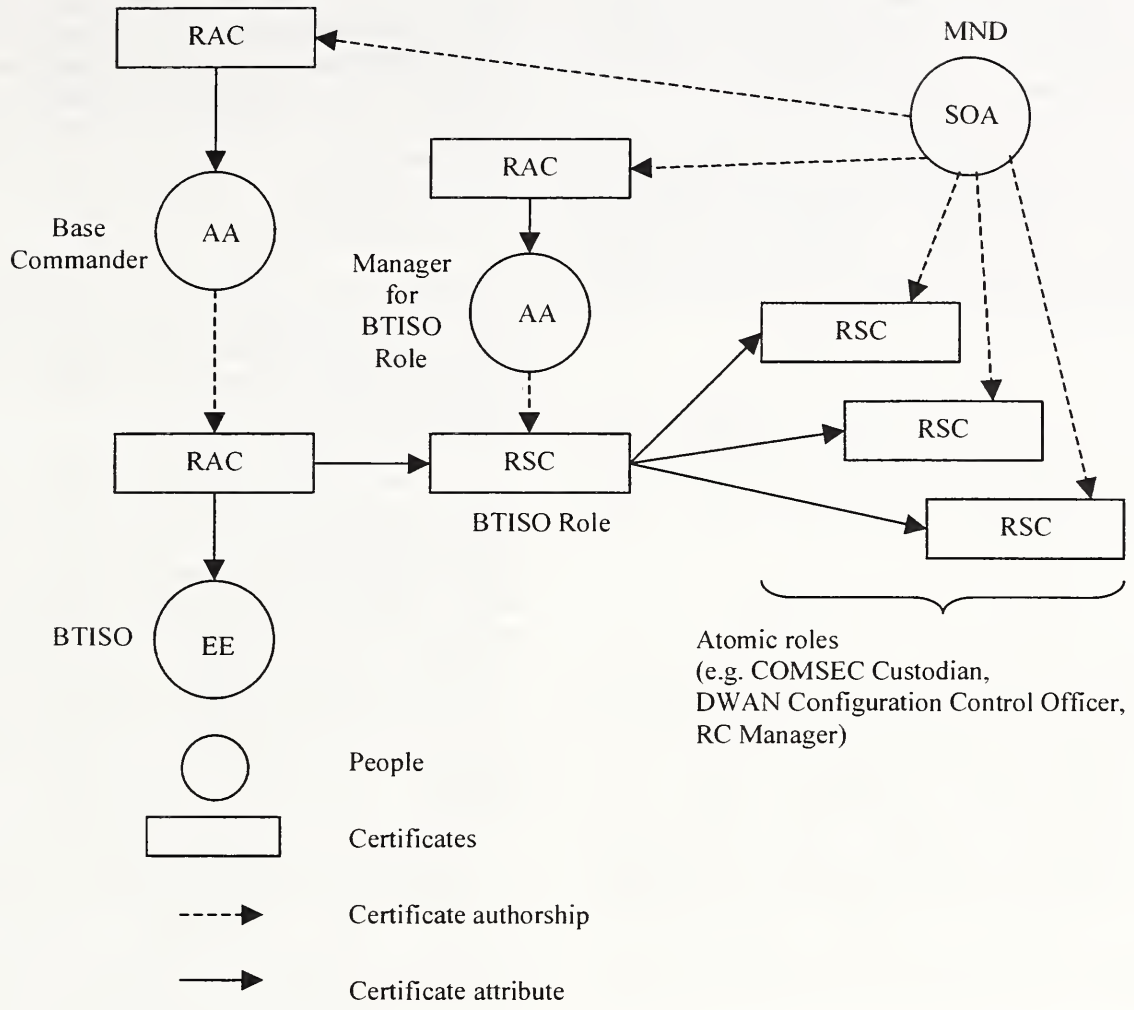


Figure 3 – Delegation Chains

using the issuer's PKC, and PKC verification using the certificate for the CA.

b) The SOA directly issues all atomic RSCs.

c) The RACs issued to the role managers directly reference atomic RSCs and do not reference complex roles.

d) Delegation in the role specification validation chains is uniform. I.e. there are always the same number of role managers in the management delegation hierarchy for each complex RSC.

e) The RSC role hierarchy is uniform. I.e. there is always the same number of complex RSCs in the chain from the end entity's role RSC to the atomic RSCs.

Now, let num_{roles} be the number of complex roles in the RSC hierarchy from the end entity's RSC to the atomic RSCs (including the end entity's role). Let num_{mgr} be the number of role managers in the management delegation hierarchy. Let num_{cdr} be the number of entities in the role assignment validation chain extending back through commanders to the SOA (including the end entity but not including the SOA).

Now, consider the number of certificates that need to be validated in the delegation chains. The atomic RSC containing the required privilege must be validated. Also, each complex RSC in the role hierarchy must be validated. This requires validation of the complex RSC itself and validation of each of the manager's RACs up

the chain to the SOA. Therefore, for a role used by an end entity or a superior commander $1+num_{roles}(1+num_{mgr})$ operations are required to validate its management chain delegation.

The role used by an end entity or a superior commander is assigned using a RAC, which must be verified. The validity of each superior commander's role must also be verified, which means validating its complete management delegation chain too. The complete set of attribute-certificates is then $num_{cdr}(1+(1+num_{roles}(1+num_{mgr}))$.

The number of operations required to validate an access will be twice the number of certificates in the relevant validation chains (from assumption a.). Therefore the overall complexity of making an access control decision for an end entity is:

$$2num_{cdr}(2+num_{roles}(1+num_{mgr})) \quad (1)$$

If a very simple authorization structure is used, where $num_{roles}=1$, $num_{mgr}=1$ and $num_{cdr}=2$, as depicted in Figure 3, then 16 operations are needed to make an access control decision. However, within DND five levels of command delegation would not be unreasonable. E.g. delegation might proceed from the MND, to Chief of the Defence Staff, to the Commander of the Army, to the Base Commander, to the BTISO. Now as a more typical example, consider the case where $num_{roles}=3$, $num_{mgr}=2$ and $num_{cdr}=5$. Complexity for an access control operation is now 110.

Public-key operations are expensive and the complexity of implementing this model seems high. This bears out the complexity warnings in [x509], and in [FH00] where Farrell and Housely do not recommend the use of delegation chains. This complexity results from attempting to mirror the distribution of privilege within a real organization. If the processing complexity can be managed, the flexibility of being able to model the actual privilege delegation paths in an organization is an advantage of this role-based model.

The complexity due to processing paths and retrieving certificates may be mitigated through the use of a cache within the verifier components. This possibility stems from the observation that most of the authorization structure is stable for significant periods of time. The roles assigned to individuals are often stable of a period of months. The privileges associated with roles would also have a similar period of stability. Significant segments of the certificate chains can be pre-validated and cached. Many different end entities require the validation of common chain segments. For example a superior commanders role validation is used in validating access requests for all subordinates. Only chain segments that have changed recently need to be revalidated. The investigation of efficient caching schemes to improve the efficacy of implementation is future work.

5 Conclusion

This work demonstrates how the X.509 standard can be used to support Canadian Department of National Defence authority structure models. It is expected that the results are applicable to large enterprise environments in general.

The roles model in the X.509 standard is compatible with the hierarchy of roles concept within role-based access control (RBAC). An interpretation of the X.509 standard is proposed that allows the construction of complex super-roles from more basic sub-roles. This structure leads to a separation of attribute authorities responsible for the specification of roles, from attribute authorities responsible for the assignment of roles. The combined effect is to produce a PMI model that meets the DND criteria for control over the granting of authority.

The results provide insight into, and quantification of, the complexity of the delegation chains. The use of a roles based model for delegation is seen as being important to the scaling of PMI to service large enterprises with mature, complex authority structures. Using role assignment and role specification certificates in conjunction with delegation paths will be a challenge for designers in complex business transactions. The large number of certificates required in delegation models will complicate implementation. This concern may be mitigated if the verifier can cache certificates and recently calculated delegation paths.

References

[Bra00] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, Mass, 2000.

[Dep99] Department of National Defence. *Delegation of Authorities for Financial Administration for DND and the CF* A-FN-100-002/AG-006, Government of Canada, May 1999.

[FH00] S. Farrell and R. Housley. *An Internet Attribute Certificate Profile for*

Authorizations. Draft – PKIX Working Group. August 2000. (work in progress). http://www.ietf.org/internet-drafts/draft-ietf-pkix-ac509prof-00.txt

[Gov96] Government of Canada. *The Business Case For Electronic Authorization and Authentication (EAA) in The Government of Canada*. January 1996.

[Gra01] Grandy, Chris, *Using A Privilege Management Infrastructure To Support Business Processes Within The Department Of National Defence And The Canadian Forces*, Master's Thesis Royal Military College of Canada, April 2001.

[Int00] International Telecommunications Union. *ITU-T Recommendation X.509|ISO/IEC 9594-8: Information Technology – Open Systems Interconnection – The Directory: Public-Key and Attribute Certificate Frameworks*. ITU-T, 2000.

[WH99] P. Wing, B. O'Higgins. Using Public-Key Infrastructures for Security and Risk Management. In *IEEE Magazine*, pages 71-73, September 1999.

Password-Enabled PKI:

Virtual Smartcards versus Virtual Soft Tokens

Ravi Sandhu
SingleSignOn.Net Inc., and
George Mason University
rsandhu@singlesignon.net

Mihir Bellare
SingleSignOn.Net Inc., and
University of California, San Diego
mihir@cs.ucsd.edu

Ravi Ganesan
SingleSignOn.Net Inc.,
11417 Sunset Hills Road, Reston, VA
ravi@singlesignon.net

Abstract

Recently there has been considerable interest among PKI vendors and researchers in the concept of password-enabled PKI. Several viable proposals and products have emerged. Fundamentally there are two distinct methods for using passwords with private keys. One method is to use the password to retrieve a private key, while the other uses the password as one component of the private key. We motivate the names virtual soft tokens for the former and virtual smartcards for the latter. The major characteristics of these two approaches are identified and contrasted.

1. Introduction and Motivation

The notion of a password-enabled PKI sounds like an oxymoron to those of us who have lived through the last decade of discussion on PKI and its rosy prospects. PKI was supposed to do away with passwords. By all logic and forecast, passwords should be a relic of the stone age of cyberspace and should no longer be in use today. PKI was expected to replace them with private keys securely generated and forever safe in tamper-proof smartcards. In the coming brave new world, these private keys would be activated by appropriate biometrics securely embedded and captured by the smartcard. The reality of 2002, however, is that passwords are used in cyberspace on a scale scarcely imagined a decade ago. There are hundreds of millions, perhaps even billions, of instances of password usage in cyberspace every day. Conservatively, consider tens of millions of users each invoking ten instances of password usages per day. In contrast one has to look high and low to find actual uses of smartcards, even in laboratory or pilot situations.

Simply stated, it is an indisputable empirical fact that smartcards have not happened.[1] If the original vision of smartcards with ubiquitous readers had become reality there would be no need to talk about password-enabled PKI. All the same, it is worth mentioning that the vision of smartcards has not faded completely, and they may still happen some day. At this moment the DoD is engaged in a major rollout of smartcards in numbers that make sense in the scale of today's Internet.[2] Not a few thousand or even a few hundred thousand but in the scale of 2 to 5 million. The discipline and resources of the DoD have few parallels in the world. This is a fascinating experiment to watch. It may finally prove the feasibility of a large-scale deployment of smartcards. Nonetheless it will be hard for Federal Government agencies, corporations, educational institutions, etc. to emulate this scale of deployment of smartcards. Note that the difficulty is not so much in the process and cost of issuing the smartcards per se, but much more in deploying smartcard readers on each and every computer in use by the user population. Proliferation of devices such as PDAs and wireless phones further compounds the problem. Moreover, we have needs at larger scale than the DoD experiment. The Federal Government often deals with 100's of millions of users. It is not unusual for large corporations to be in touch with 10's of millions of users. It is certainly within their vision to be in touch with 100's of millions and even billions of users in the future. Given the multi-year deployment of DoD smartcards, one wonders how the truly large scale will ever be realized in this mode. It seems rather unlikely that we will have a national scale deployment of smartcards in the near future, let alone a global scale deployment. Organizations whose PKI strategy depends entirely on smartcards happening very soon are making a rather risky bet.

[1] This statement should be understood in context of the use of smartcards for PKI on the Internet via widely available desktops and laptops. The use of smartcards in specialized applications has seen considerable success, more so in Europe and Asia than in North America. In these applications the smartcard is often embedded in a device such as a wireless telephone or a television-set top box, or in a credit-card with specialized readers.

[2] http://www.defenselink.mil/news/Oct2000/n10102000_20001010 7.html

If smartcards are not available where do we store the user's private key and how do we make it portable? Most systems today store the private key encrypted with a user-selected password on the hard disk. Portability is achieved by transporting the encrypted private key on removable media, such as a floppy disk. Currently one cannot guarantee availability of floppy disk readers, or other media-specific devices, on every computer. In general the portability of any media-specific transport is questionable in a truly open environment. All the same the notion of a "soft token", that is a private-key encrypted with a password, in contrast to a "hard token", that is a private-key which never leaves a smartcard, has been around for over a decade and has been deployed in several systems. From the user's perspective it is a natural progression to store the soft token on a network server rather than having to carry it around. This is very easily achieved by copying the contents of the soft token onto a server.

Password-enabled PKI relies on passwords to enable the use of private keys. Passwords are extremely easy to use and are easily usable from multiple computers. Users continue to express frustration with passwords, mainly because they have too many of them and are often required to change them too often. Password-enabled PKI alleviates both of these problems. PKI facilitates use of the same digital identity at multiple relying parties, including those with whom the user has had no prior contact. Thus a user need not be burdened with a separate password for every relying party. With a dramatically reduced number of passwords to remember, users can be reasonably persuaded (or gently enforced) to choose passwords of adequate complexity without having to write them on paper as a memory aid. Gentle enforcement of password complexity rules is more user-friendly than the current conventional wisdom of constantly chasing users to change passwords as a countermeasure to selection of weak passwords (or the writing on paper of many complex passwords).

To a large extent password-enabled PKI has happened in spite of PKI orthodoxy which calls for smartcard-enabled PKI wherein the private key never leaves the smartcard. As such the concept of password-enabled PKI has not really been studied systematically. Instead a number of proposals have been published and implemented, each one motivated by its own principal considerations. One of the goals of this paper is to provide a systematic analysis from security, functionality and operational perspectives. We specifically assume that the underlying cryptographic protocols are secure. This is a reasonable assumption since in many cases proofs of security or at least strong informal arguments have been provided. Empirically, we can say that it is quite feasible to get the cryptography correct. Our goal is to understand the overall security that is achieved and the functional and operational implications of specific approaches.

2. Password Vulnerabilities

It is generally agreed that password-enabled PKI will not provide the same level of security as smartcard-based[3] or biometric-based PKI.[4] All the same there is considerable confusion about the actual security vulnerabilities of passwords. So we begin with a brief discussion of password vulnerabilities before turning to the main topic of the paper.

There are some inherent vulnerabilities of password-based systems. A password can be compromised without knowledge of the legitimate owner. There is no physical evidence of theft. The possibility for undetected compromise is further enhanced if users reuse the same password at poorly protected sites, who may do something silly like storing passwords in the clear (or even less extreme). This is almost as bad as writing the password on paper and displaying it in a public place. Conversely a password can be easily shared. A common example is a corporate executive who shares her password with her secretary. In absence of other convenient mechanisms for this purpose, sharing of a password is a simple means to provide the secretary access to the executive's email. These inherent vulnerabilities cannot be completely addressed without cooperation and education of users. However, technology to mitigate these problems does exist. Misuse detection systems can help in identifying occurrences of misuse due to compromise or sharing. The concept of a trusted path can be used to ensure that passwords are revealed to trusted entities and not to software that spoofs the look and appearance of trusted entities. Even more effective is the use of protocols that do not reveal passwords but instead prove knowledge of the password for authentication.

Passwords are also susceptible to guessing attacks. On-line guessing requires the attacker to try

[3] It should be noted that not all smartcards are equal. It is possible to do smartcards very badly so they are not tamperproof. For purpose of this paper we assume that smartcards can be made tamperproof. In practice this is a difficult goal.

[4] Currently there is considerable interest in biometrics for authentication, especially following the events of September 11, 2001. Biometric-enabled PKI, with or without the use of smartcards, is a fascinating possibility for higher assurance than achieved by password-enabled PKI or even smartcard-enabled PKI. Consideration of biometric-enabled PKI is beyond the scope of this paper.

password guesses directly against the protected system and see if the guessed password works successfully. Enforcement of password complexity rules makes these attacks harder. The threat is further mitigated by throttling schemes which slow down the rate at which such attacks can be pursued. With a simple "three guesses and out" rule it is possible to introduce denial of service vulnerabilities but more sophisticated approaches are possible. For our purpose we assume that on-line guessing is taken care of in some such manner.

The most serious threat to existing password-based systems is the possibility of off-line dictionary attacks. In these attacks the attacker has knowledge of the outcome of some cryptographic operation which uses the password as a "key". The precise knowledge available and attendant attack varies from system to system. We will generically call this information as known plaintext.[5] We will also use the shorter term dictionary attack to specifically mean off-line dictionary attack. Known plaintext is sufficient to allow an attacker to verify if a password guess is correct or not. The crucial aspect is that the guesses can be verified off-line. By trying large numbers (tens or hundreds of thousands) of commonly used passwords from a so-called dictionary the attacker can succeed without searching the entire key space. This problem has been well known since at least 1979 [MT79] but it continues to be a major vulnerability of existing password-based systems [WU99]. We can distinguish between network-based and server-based offline dictionary attacks. In the former case the required known plaintext is obtained from the network protocol, possibly by network sniffing or more directly by simply running the protocol. Server-based attacks require capture of this information by server penetration in some way. In particular system administrators of the server will typically have easy access to the requisite known plaintext.

To complicate dictionary attacks a password is typically salted before it is used as a "key". The salt is a random number which is usually not kept secret. Different users with different salts will generate different known plaintext making the dictionary attack more time consuming. In particular the attacker cannot precompute known plaintext values from the dictionary passwords alone, but must do so separately for each value of the salt. This makes precomputation of the dictionary attack infeasible since the space of possible salts is very large.

[5] It should be noted that known plaintext can be known structure rather than known content.

3. Password-based Cryptographic Protocols

The Kerberos system [KN93, NT94] was one of the first to use passwords as a basis for cryptographic protocols. Susceptibility of Kerberos to network-based dictionary attacks is well-known [BM91, WU99].[6] A number of password-based cryptographic protocols immune to network-based dictionary attacks have since been published. Notable amongst these are the EKE [BM92], SPEKE [JAB96] and SRP [WU98] protocols, but there are many others. All these protocols use public-key cryptography in some way, a requirement that has been shown to be theoretically necessary [HK99b]. We can reasonably claim that, since about 1992, we know how to construct password-based cryptographic protocols immune to network-based dictionary attacks.[7]

In the above protocols both the client and the server store the password. Server compromise is however a real threat, and in this case it immediately yields the password to the attacker. In the augmented EKE [BM93] and SNAPI-X [MPS00] protocols, the server holds a hash of the password rather than the password, so server-compromise does not immediately yield the password to the adversary, but the attacker, having compromised the server, can still mount a dictionary attack based on the password hash. Immunity to server-based dictionary attack is not so easy to achieve. An approach based on multiple servers has recently emerged. The user's password is used to retrieve shares of a secret from multiple servers without exposure to network-based dictionary attacks. The secret is then assembled at the client computer from its shares. This long random secret can then be used for a variety of cryptographic purposes. Ford and Kaliski [FK00] present an elegant n-of-n scheme for this purpose, and suggest using 2-of-2 in practice. In general all n servers need to be penetrated by an attacker. Compromise of (n-1) is not sufficient. Jablon [JAB01] proposes schemes with additional desirable properties.

In practice schemes with multiple servers impose operational requirements to keep additional servers online and available. Moreover these servers may be subject to common-mode security failures.

[6] Kerberos failure to server attacks is complete and absolute obviating the need to do a server-based dictionary attack. It is interesting to note that Kerberos employs the user name and realm name as salt in its string_to_key function [KN93].

[7] Security analysis of such protocols is however subtle, and definitions of security goals, together with proven secure protocols, including a proof for the core of EKE, have emerged only more recently [BPR00, BMP00].

Once an attacker knows how to break one server, likelihood of success on the other is quite significant in practice. Possibility of insider attacks could be reduced due to requirement of insider collusion across multiple servers, but outsider attacks may not be significantly mitigated. At the same time operational quality may be degraded. Security infrastructure is expected to be more robust than the infrastructure it protects. Each security server would generally be replicated for reliability purposes. Each additional server therefore counts as two. Appropriate hardening of a single server with suitable separation of duties and least privilege could present a more viable approach to outsider and insider attacks.

4. Password-enabled PKI

With this background and motivation we now address the main topic of this paper. There are fundamentally two distinct ways to implement password-enabled PKI.

1. Employ the user's password as a means to securely retrieve the user's private key on to any computer from where it can then be used without further online interaction.
2. Employ the user's password as a component of the user's private key which can be used only in conjunction with another component which, in turn, can only be used on an online server.

The principal distinction between these two approaches is whether or not the user's private key is completely resident on the user's computer in a usable form. In the first case the user's key is available in the clear on the user's computer and can be used independent of any further interaction with the online server. Network-based storage of a user's private key in this manner is analogous to storage of an encrypted private key on a soft token. Once this private key is decrypted on a computer it can be used indefinitely without continued need for the soft token. Because of this analogy we call this approach a **virtual soft token** (or network-based soft token).

In the second approach the password only enables usage of the private key without bringing the entire private key together in one place. The overall private key is split into two components. One component is computed from the user's password. The other is resident on a secure online server. Let us call the former component the password component and the latter component the server component. Both components are required whenever the user wishes to use her overall private key but they are never brought together in one place. Instead an interactive protocol is carried out to achieve that result. Network-based usage of a component of the user's private key in this manner is analogous to usage of a private key in a smartcard. Just as the private key never leaves the smartcard, the server component of a user's overall private key never leaves the network server. Because of this analogy we call the second approach a **virtual smartcard** (or network-based smartcard).[8]

In the remainder of this paper we identify major characteristics of these two approaches to password-enabled PKI and compare them.

4.1. Virtual Soft Tokens

Virtual soft tokens enable retrieval of a user's private key onto any computer of the user's choice. A simplistic approach to this task would be to store each user's key encrypted with the user's password on an online server. Anyone could retrieve any of these encrypted keys, but without knowledge of the correct password would not be able to directly decrypt them. The virtual soft token is simply a substitute for the physical soft token.[9] Unfortunately this scheme is susceptible to dictionary attacks. An attacker who has access to the encrypted private key can verify guesses for the password by decrypting the private key with the guess and verifying success or failure with respect to the known public key.[10]

A virtual soft token therefore cannot be freely accessible for download. Instead it must be protected from network-based dictionary attacks by one of the password-based protocols such as EKE, SPEKE or equivalent.

Virtual soft tokens were first proposed in the SPX system [TA91]. The designers of SPX did not feel comfortable downloading the user's long-term private key to the client machine. Instead they proposed creation of a short-term private key whose public-key certificate was signed by the user's long-term private key. Only the short term key would be downloaded to the client machine for unrestricted use within its short life. In a sense this proposal is stronger than a physical soft token since compromise

[8] It should be noted that the term virtual smartcard has been used for schemes that are virtual soft tokens in our terminology. This is inappropriate since an essential characteristic of a smartcard is that the private key never leaves the smartcard.

[9] People who use soft tokens can trivially virtualize them in this manner by simply copying the soft token to some server from where it is accessible.

[10] Hoover and Kausik [HK99a] suggest that this dictionary attack can be avoided by protecting disclosure of the public key. Unfortunately their approach of "cryptographic camouflage" negates the main advantage of PKI where the public key does not need to be confidential. Technically, Hoover and Kausik also require elimination of redundancy in encryption of the private key so "known structure" attacks are not possible.

of the client leads to compromise of a short-term key with a life of say 8 hours. Compromise of a client with a long-term key with a life of say 1 year is more devastating. In another sense the SPX soft token is weaker with respect to non-repudiation. The user's long-term key is exposed on the SPX server where it is needed to construct the certificate for the short-term key. The SPX server can therefore impersonate the user via knowledge of the long-term key. Novell's NetWare v4 deployed a similar process for downloading a temporary private key [KPC95] (although it used a different set of underlying cryptographic algorithms).

Recent proposals for virtual soft tokens have returned to the idea of retrieving the long-term private key on to the client. As we have seen we know how to prevent network-based dictionary attacks in this context. A number of protocols for this purpose were recently presented by Perlman and Kaufman [PK99]. There are some significant differences in detailed properties of these protocols. Nonetheless from our vantage they all share a common core of security properties: exposure of long-term private keys on the client and immunity to network-based dictionary attacks.

Ford and Kaliski [FK00], and later Jablon [JAB01], propose solutions to server-based dictionary attacks. As discussed earlier these solutions require additional servers which may degrade operational quality while the gain in security may be diminished due to common-mode failures.

4.2. Virtual Smartcards

Virtual Smartcards are based on split private keys. In classical 2-key RSA the public and private keys for given n are related by the following equation.

$$e*d = 1 \bmod \varphi(n)$$

The splitting of d into d1 and d2 is computed as follows.

$$d1*d2 = d \bmod \varphi(n)$$

The fundamental operation of exponentiation in RSA then gives us the following equations.

$$(M^{d1})^{d2} \bmod n =$$
$$(M^{d2})^{d1} \bmod n =$$
$$M^{d1*d2} \bmod n =$$
$$M^{d} \bmod n$$

This idea can be extended to more than two splits of the original private key d if so desired. It can also be applied to an additive rather than multiplicative split. These ideas were first published by Colin Boyd [BOY89]. Their first application to virtual smartcards is due to Ganesan [GAN95, GAN96]. Ganesan's innovation was to realize that one of the split keys, say d1, can come from a password and therefore easily remembered and carried around mentally by a user. Nonetheless security of d2 is equivalent to security of a traditional RSA private key.[11]

To summarize, in a 3-key RSA system there are 2 private keys whose multiplication mod $\varphi(n)$ is equivalent to a single overall private key. One of these keys d1 is computed from the user's password and known only to the user. It is the password component of the overall private key. The server component of the overall private key is d2 which is stored and used only on a secure online server. The server component constitutes a virtual smartcard which can be used only if knowledge of d1 is demonstrated. The overall private key d is never reconstructed on the client or the server. Every use of d involves an online interaction between the client and server.[12]

An immediate benefit of virtual smartcards is the ability to do instant revocation. The server resident d2 can be revoked at any time rendering the password component d1 completely useless. From here on d1 cannot be used to generate a signature even if the certificate for (e,n) continues to be valid. The network-based virtual smartcard will refuse to participate in the signing protocol. This is a tremendous benefit relative not only to virtual soft tokens but also to local smartcards. Another benefit is potential for misuse detection by monitoring usage of the virtual smartcard. Note that these benefits continue to accrue even if d1 is stored on a local smartcard rather than computed from a password. As such virtual smartcards provide valuable additional services even when we reach the age of ubiquitous smartcards (and smartcard readers).

MacKenzie and Reiter [MK01] have an interesting variation on the use of split-key RSA. They show how to make the loss of a local smartcard safe in that there is no private key within the smartcard that can be extracted. Also the smartcard is useless without knowledge of the user's password. In a nutshell the password component of a user's password is much the same as in Ganesan's scheme.

[11] This notion is formally proved in Appendix A.

[12] Contrast this with SPX discussed above where the entire private key is resident on the server. SPX thereby fails to provide non-repudiation.

The server component, however, is stored encrypted with the server's public on the smartcard, i.e., d2 encrypted with the server's public key. Cooperation of the server is therefore required whenever the smartcard is used. This is much like the virtual smartcard scheme. However, revocation is done out of band and requires the servers to maintain the equivalent of revocation lists. Mobility in this scheme is achieved by moving the device from computer to computer which requires a suitable reader or interface. This is a characteristic of conventional local smartcards.

5. Conclusion

In this paper we have identified two approaches to password-enabled PKI. We have motivated the reasons for calling these virtual soft tokens versus virtual smartcards. Virtual smartcards remove exposure of the user's private key on a client computer while allowing for misuse detection and instant revocation. Conversely, virtual soft tokens expose the user's private key on client computers and cannot support misuse detection or instant revocation. These are substantial differences.

As we look to the future, PKI thinking must depart from its conventional reliance on smartcards as the technology which will make PKI real. With hundreds of millions of computers deployed all over the world today retrofitting smartcard readers on each one is a formidable task. A variety of wireless and personal computing devices are also proliferating. Uniform availability of smartcard readers across all these devices is extremely unlikely. Instead we should look to an environment where virtual smartcards are pervasive with local smartcards and biometrics being used for higher assurance situations.

The recent big push for identity services on the Internet has veered away from PKI to proposals that are entirely password based and make extensive use of symmetric cryptography. In the past year we have seen a number of such initiatives from big players in the Information Technology arena. PKI still offers considerable advantages over symmetric technology. But if the PKI community is not alert and adaptive to industry trends we may find the baby is thrown out with the bath water.

References

[BM91] Bellovin, S and Merritt, M. "Limitations of the Kerberos authentication system." Proceedings of the Winter USENIX Conference, 1991, pp 253-267.

[BM92] Bellovin, S and Merritt, M. "Encrypted key exchange: password-based protocols secure against dictionary attacks." Proceedings of the IEEE Symposium on Security and Privacy, 1992, pp. 72 –84.

[BM93] Bellovin, S and Merritt, M. "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise." Proceedings of the ACM Conference on Computer and Communications Security, 1993, pp. 244 – 250.

[BPR00] M. Bellare, D. Pointcheval and P. Rogaway. "Authenticated Key Exchange Secure Against Dictionary Attacks." Advances in Cryptology - Eurocrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1807, B. Preneel ed, Springer-Verlag,2000.

[BMP00] V. Boyko, P. MacKenzie and S. Patel. "Provably Secure Password Authenticated Key Exchange Using Diffie-Hellman." Advances in Cryptology - Eurocrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1807, B. Preneel ed, Springer-Verlag, 2000.

[BOY89] C. Boyd. Digital multisignatures. In Cryptography and Coding, H. Beker and F. Piper eds., Oxford University Press, 1989, pp. 241-246.

[BS01] M. Bellare and R. Sandhu. "The security of practical two-party RSA signature schemes." Manuscript, November 2001. Manuscript available via http://www.cse.ucsd.edu/users/mihir.

[FK00] Ford, W. and Kaliski, B. "Server-assisted generation of a strong secret from a password." Proceedings 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000, pp 176 – 180.

[GY94] R. Ganesan and Y. Yacobi. A Secure Joint Signature and Key Exchange System. Bellcore Technical Report TM-24531, October 1994.

[GAN95] Ravi Ganesan. Yaksha: Augmenting Kerberos with public-key cryptography.

Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1995.

[GAN96] R. Ganesan. Yaksha: Towards Reusable Security Infrastructures. PhD Thesis. Johns Hopkins University, Baltimore, MD, 1996.

[HK99a] D. Hoover and B. Kausik, "Software smart cards via cryptographic camouflage." Proceedings of the IEEE Symposium on Security and Privacy, 1999.

[HK99b] S. Halevi and H. Krawcyzk. "Public-key cryptography and password protocols." ACM Transactions on Information and System Security (TISSEC) Volume 2 , Issue 3 (August 1999), Pages: 230 – 268.

[JAB96] D. Jablon, "Strong password-only authenticated key exchange." ACM Computer Communications Review, October 1996.

[JAB01] D. Jablon, "Password authentication using multiple servers." Proceedings RSA Conference: Cryptographers' Track, 2001 San Francisco, CA, April 8-12, 2001, Springer LNCS 2020.

[KPC95] C. Kaufman, R. Perlman and M. Speciner, "Network Security: Private Communication in a Public World." Prentice-Hall, 1995.

[KN93] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). RFC 1510, September 1993.

[MPS00] P. MacKenzie, S. Patel and R. Swaminathan. "Password Authenticated Key Exchange based on RSA." Advances in Cryptology - Asiacrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed, Springer-Verlag, 2000.

[MR01] P. MacKenzie and M. Reiter. "Networked cryptographic devices resilient to capture." Proceedings of the IEEE Symposium on Security and Privacy, 2001.

[MT79] Robert Morris and Ken Thompson, "Password Security." Communications of the ACM, Volume 22 Issue 11, November 1979.

[NT94] C. Neuman and T. Ts'o. "Kerberos: An Authentication Service for Computer Networks." IEEE Communications, 32(9):33-38. September 1994.

[PK99] R. Perlman and C. Kaufman, "Secure password-based protocols for downloading a private key." Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1999.

[TA91] J. Tardo and K. Alagappan, "SPX: global authentication using public key certificates" Proceedings of the IEEE Symposium on Security and Privacy, 1991, pp. 232-244.

[WU98] T. Wu, "The Secure Remote Access Protocol." Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1998.

[WU99] Thomas Wu, "A Real-World Analysis of Kerberos Password Security." Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1999.

Appendix A: Equivalence of 3-key RSA To 2-key RSA

We show that the security of 2-key RSA is equivalent to the security of 3-key RSA, following Ganesan and Yacobi [GY94] who first established this conjecture of Boyd [BOY89].[13]

A traditional 2-key RSA pair is generated as follows.

1. Generate two large, distinct primes p, q of roughly equal bit-length
2. Compute $n=p*q$
3. Select e such that $\gcd(e,\varphi(n))=1$ and $1<e<\varphi(n)$, where $\varphi(n)=(p-1)*(q-1)$
4. Compute d, such that $1<d<\varphi(n)$ and $e*d \equiv 1 \bmod \varphi(n)$

[13] We note that this argument only reflects key-recovery attacks. Security arguments for our schemes that consider forgery attacks are more involved, and provided in [BS01].

5. Destroy p, q
6. Public key is e, n and private key is d

In the password-based 3-key system steps 1-4 are as above, followed by the steps given below.

5. Ask user to select a password Pwd that meets password selection rules
6. Pick an iteration count IC
   Repeat
   6.1 Pick a random SALT
   6.2 Compute d1 = Expand(Pwd,SALT,IC)
   Until (gcd(d1, φ(n))=1 and 1<d1<φ(n))
   [The function Expand is specified via PKCS5. The IC value and the final SALT value are accessible for subsequent use by the user.]
7. Compute d2 such that 1<d2<φ(n) and d1*d2 = d mod φ(n).
8. Destroy p, q, d
9. Public key is e, n; user's private key component is d1 (user remembers password Pwd from which d1 is computed) and appliance private key component for that user is d2.

We claim that the expected number of iterations of the repeat loop in Step 6 is around 2, so that the loop terminates quite fast. (Assume the Expand function is random and has range $\{0,1\}^k$ where $2^{k-1} \leq n < 2^k$. Then the expected number of iterations is at most (2*n)/φ(n) which is very close to 2.)

The strength of the split-key setting is that it provides as much security as RSA even if the user password is compromised, in the following sense: the problem of computing d2 given n, e, d1 is as hard as the traditional RSA problem of computing the secret exponent given the public key in the standard setting.

To detail this claim, we recall that the traditional RSA problem is defined as follows:

Given: n, e
Compute: d such that e*d=1 mod φ(n) and 1<d<φ(n)

We define the split-key RSA problem as follows:

Given: n, e, d1
Compute: d2 such that e*d1*d2 =1 mod φ(n) and 1<d2<φ(n)

We claim that if the split-key RSA problem is tractable, then so is the traditional RSA problem. To justify this claim, we assume we are given a method of solving the split-key RSA problem relative to a password generation process (formally, randomized algorithm) P that models the client's choice of password. The following code shows how we can then solve the traditional RSA problem. Explanations follow the code.

Given n, e,

1. Run P to obtain a password Pwd
2. Pick random SALT, and IC, and compute d1 = Expand(Pwd,SALT,IC)
3. Run the given split-key RSA solving method on input n, e, d1 to obtain d2
4. Let m = e*d1*d2 - 1
5. Use m, e to factor n [see later text for why this is possible]
6. Use the factorization of n to compute φ(n)
7. Let d be the inverse of e modulo φ(n)
8. Output d

Note that the value d1 chosen in Step 2 may not be relatively prime to φ(n) and in that case the algorithm will probably not succeed. However, d1 as chosen in step 2 has probability around 1/2 of being relatively prime to φ(n) and hence the success probability of the algorithm above is about one-half that of the given method for solving the split-key RSA problem.

The value m computed in Step 4 is a multiple of φ(n), because, modulo φ(n) we have:

$$e*d1*d2 - 1 = e*d1*d2 - e*d = e*[d1*d2-d] = 0.$$

Step 5 uses a well-known fact, namely that given a multiple of φ(n) it is possible to factor n.

One might ask why the algorithm does not, after step 3, simply compute d = d1*d2, output d and halt, since this d satisfies e*d mod φ (n) = 1. However this d may not satisfy 1<d<φ(n).

Delegated Cryptography, Online Trusted Third Parties, and PKI

Trevor Perrin, Logan Bruns, Jahan Moreh, Terry Olkin
{tperrin,logan,jmoreh,tolkin}@sigaba.com, Sigaba Corporation

Abstract

We propose that enterprise PKI users should delegate asymmetric cryptography operations to an online trusted third party maintained by their enterprise, thus freeing themselves from the burdens of owning key pairs or interfacing with PKI. Users would authenticate to this third party (which we'll call a delegate server) and then request it to sign and decrypt data on their behalf with its own private key and encrypt and verify data with the public keys of other users or other delegate servers. A delegate server would thus be like a CA in that it represents a group of users but like an end-entity in that it signs and decrypts using its own private key and encrypts and verifies using public keys which it has calculated certificate paths to. To bind encryptions and signatures performed with delegate server keys to particular users we suggest two approaches, one using XML security standards, and one using what we call signature operation certificates which are signed by a delegate server and bind a hash value to a signing user, and encryption operation certificates which are encrypted to a delegate server and bind a symmetric key to an intended decrypting user. These operation certificates have several benefits, and so we propose that conventional PKI end-entities as well as delegate servers could use them to encapsulate signatures and encryptions, and that current PKI protocols could be modified to support them. If this was done, enterprises could individually choose whether to utilize delegate servers or conventional PKI. In many situations a delegate server infrastructure would be easier to deploy, easier to use, and easier to integrate with applications, and would offer advantages in security, extensibility, and efficiency.

1 Introduction

The deployment of cryptography on modern computer networks is proceeding on two fronts. The first is the use of cryptography to achieve authentication. Users log-on to workstations, networks, and networked applications using credentials such as passwords, one-time password devices[1,2], smartcards containing private keys, or biometrics[3]. Cryptography either provides encryption to protect the transmission of the user's credentials (e.g. SSL[4] for website passwords), or provides credentials-presentment protocols with various intrinsic degrees of security (e.g. SRP[5,6] for passwords, or SSL with client authentication for private keys).

These authentication and session-establishment protocols are easy to use for both the credentials-presenting and the credentials-verifying parties, and thus are widely deployed. Such methods are not, however, sufficient to fulfill the promise of cryptography. Ideally a cryptographic infrastructure would be capable of providing confidentiality and authentication to both interactive and noninteractive communications amongst large groups of users[1]. Compared to this, authentication methods (with the exception of asymmetric key pairs) are limited in both scale and scope: scale in that a single credential should only be used between a single pair of users, and scope in that these methods can only secure interactive traffic, and thus cannot be used to encrypt or sign data such as emails or files. Attempts to solve the scalability problem involve clients authenticating once to a "single sign-on" service with their primary credentials and then receiving secondary credentials that they can use to access other services. This approach represents the current cutting-edge of authentication techniques, particularly on the web where it is being pursued by Microsoft Passport[7], the Liberty Alliance Project[8], and the SAML XML specification[9].

The second front on which cryptography is being deployed is known as public key infrastructure (PKI). This technology assumes that every client should possess a long-lived asymmetric key pair[10]. The public key can be shared with different parties, which can then authenticate the private key owner, verify the owner's signature on a piece of data, or encrypt data that only the owner can decrypt. A single key pair thus allows point-to-many instead of just point-to-point security, and can be used to secure both interactive and noninteractive communications. These characteristics enable certificates[11]: a user Alice could sign Bob's public key along with Bob's name, thus producing a certificate which could be published in a public direc-

tory or carried around by Bob, and which would convince anyone who trusts Alice that Bob's public key really belongs to him. Bob can issue certificates to other users as well; the certificates issued amongst a group of users comprise a directed graph, and if one user can compute a path from himself to another then he can determine that other user's public key and use it to secure communications between them. Often a specialized entity known as a certificate authority (CA) will assume responsibility for issuing, revoking, and publishing the certificates for some group of users. A system of cooperating CAs, directories, and other supporting services are what we collectively refer to as PKI.

PKI appears to meet our requirements for cryptographic infrastructure, yet attempts to deploy it over the last decade have met with strikingly little success. Even within a single enterprise, PKI rollouts are often expensive and time-consuming, and result in stovepipe systems unable to interoperate with each other, integrate with new applications, or evolve to meet new demands and incorporate new technologies[12,13,14,15]. Some feel these are growing pains that will disappear as the technology matures, but we will argue that they instead reflect several systemic flaws in the PKI vision:[2]

First, private keys are difficult for people to use and easy for attackers to abuse. Private keys are not memorable like passwords, derivable from the person like biometrics, or enterable from any keyboard like one-time passwords. They are instead typically stored in a file on the user's computer, stored on smartcards, or stored at a server that delivers them to the user on request[16]. These approaches have various deficiencies in terms of portability, universality, and security. Security concerns are aggravated because private keys can be stolen and abused offline (i.e. without generating an audit trail).

Second, trust relationships are difficult for people to manage. If a global PKI had materialized, with a single CA at the root of a certificate hierarchy, then this would be a non-issue: each user would simply configure his software to fully trust the CA's public key. Instead many different CAs exist, some public and some private, and in some systems (such as PGP[17,18]) users can issue certificates to each other as well. Faced with such a fragmented trust environment, users must configure their software's trust list with the "trust anchors" from which to begin computing certificate paths, by first importing and verifying these anchors' public keys and then indicating to what extent the user trusts each anchor or over which names the user considers the anchor authoritative. These procedures are complicated yet security-critical, and users rarely understand or perform them well[19].

Third, the interface between end-user software and PKI systems is complex and difficult to standardize. End-user software must interact with the PKI to perform management operations such as obtaining, revoking, renewing, archiving, and recovering the user's certificate and key pair, and also to construct and validate certificate paths to other users' public keys. These operations require knowledge of the PKI's management protocols, directory architecture, trust topology, certificate formats and profiles, certification policies, and revocation/validity-checking methods, among other things. Each of these provides an axis along which PKIs can and do vary. As a result, PKI end-user software tends either to provide lowest-common-denominator support for PKI or to be tightly coupled to a particular vendor's products or even a particular deployment, yielding systems that are either underfunctional or overly rigid and brittle.

Fourth, end-to-end path construction is inefficient: every user's certificates and revocation data must be made accessible to every other user in a timely fashion, and every user must compute the paths between himself and every other user with whom he wants to communicate. The first requirement necessitates a high-performance and high-availability distributed directory that will be difficult to scale to large communities of geographically dispersed users. The second requirement results in redundant computations of potentially lengthy and complex paths.

A few observations about the enterprise environment will suggest a way to remedy these flaws. PKIs are generally deployed to support users who communicate under the aegis of enterprises (meaning businesses or similar organizations). In these enterprises there are authentication methods already deployed to control access to networks and workstations; there is trust between members and their enterprise, and between enterprises and each other; there are administrators capable of configuring and maintaining networked services for enterprise members; and there are private networks offering reasonably high performance and reliability, and some measure of protection against outsider attacks.

In such environments, we believe the authentication and PKI uses of cryptography should be hybridized in the form of a networked service, hosted by an enterprise for its members, which users could authenticate to and request to perform asymmetric cryptography on their behalf. More precisely: users within an enterprise would authenticate to a locally-provided delegate server (DS) which would possess a key pair and would interface with a PKI system whose end-entities would include both individuals and other DSs representing other enterprises. To produce a signature on a document Alice would send her DS a cryptographic hash of the document, and the server would sign this hash along with an attached statement that says "this was presented

by Alice", and return this signed message to Alice who would embed it in the document. To encrypt a document to Bob, Alice would send her DS a symmetric document encryption key and the name Bob, and the DS would encrypt the symmetric key along with an attached statement "this is intended for Bob" using Bob's public key or the public key corresponding to Bob's DS. When Bob received these secured documents from Alice he would extract the messages that came from Alice's DS and either process them using his own private key or forward them to his DS. If the latter, his DS would verify or decrypt these messages and then examine the attached statements, and either confirm that it was Alice on whose behalf the signature was produced, or release the document encryption key after verifying that it was indeed intended for Bob.

This approach addresses the first flaw in PKI by using convenient authentication methods, instead of private keys, to access a server which can monitor all events for intrusion detection and response purposes. It addresses the second and third flaws by centralizing trust relationships and PKI software at the organizational instead of individual level, where they can be managed at a single point by qualified staff. It addresses the fourth flaw by associating certificates and key pairs with enterprises instead of individual users, thus reducing the volume of certificates and revocation data that must be distributed and the length of paths that must be computed. It also centralizes path computation at servers where its cost can be amortized across large groups of users.

One drawback of this approach is its reliance on communication between users and DSs. This raises issues of performance and availability, and also raises the spectre of denial-of-service and traffic analysis attacks[20], but the characteristics of enterprise networks we mentioned above should mitigate these concerns. Another drawback is the potential security risk and performance bottleneck of performing all asymmetric operations for an enterprise at a single point. We believe that adequate security can be achieved by choosing an appropriate lifetime and strength for DS key pairs and by confining sensitive data at the DS to a secure coprocessor[21,22], and that adequate performance can be achieved with appropriate hardware or techniques such as caching Diffie-Hellman key agreement values. A third drawback is that having DSs involved in all cryptographic operations may raise privacy concerns, but it provides compensating benefits such as centralized auditing, fine-grained access control, and instantaneous user revocation. A final drawback is that current cryptographic protocols and data formats were not designed with DSs in mind, but we believe that an elegant extension to the notion of certificates will make it easy to retrofit DS support into current PKI systems.

In what follows we will expand on these points to argue that DSs make large-scale cryptographic infrastructure feasible. In the next section we will take a step back and develop a more abstract understanding of the problems and methods of cryptographic infrastructure. In section three we will apply this understanding to the real world to show why delegated cryptography makes sense. And in the final section we will consider how current cryptographic protocols and data formats could be retrofitted to support this technique.

2 Cryptographic Infrastructure

For our purposes, the basic situation of cryptographic infrastructure is this: there is a graph consisting of nodes linked by communication channels, where a node could be a machine or a person, and a channel could be such things as a trusted courier, a computer network, or even just a stretch of time. Some of these channels are physically secure. Others may be subject to passive attacks, where an adversary eavesdrops on the messages going back and forth; or active attacks, where an adversary alters, deletes, and adds messages. It is desired that certain communications between nodes be confidential and/or authenticated. By confidential we mean that an adversary cannot determine their contents. By authenticated we mean that an adversary cannot delude one party to a communication as to the other party's identity.

Physically secure channels are not subject to attacks and are thus both confidential and authenticated. Otherwise passive attacks can violate confidentiality and active attacks can violate authentication. To ensure these properties on channels subject to these attacks the nodes must code their communications using cryptographic algorithms. These algorithms can be used to protect either noninteractive messages (i.e. self-contained units of data sent from one node to another) or interactive sessions. Interactive sessions allow the use of cryptographic protocols which make certain security properties easier to obtain; in particular, Diffie-Hellman key exchange[10] can establish confidential sessions between any pair of nodes, thus making authentication the only difficulty in the interactive case.

2.1 Cryptographic Data

With the exception of Diffie-Hellman key exchange, the algorithms and protocols used by two nodes to provide confidentiality and authentication require that certain related cryptographic data be used as inputs by both sides. We can classify these data as credentials, symmetric keys, or asymmetric keys. Credentials involve a data source possessed by one node (such as a password, one-time password device, eyeball, etc.) and credentials-verifying data possessed by another node

(such as the password itself, a hash of the password, an SRP verifier[5,6] of the password, an iris code[23], etc.). Credentials either possess low entropy (passwords), imprecision (biometrics), or time-variance (one-time password devices), and thus can only be used to authenticate interactive sessions (i.e. they can't provide message security). Certain credentials, such as passwords, can be used in conjunction with zero-knowledge password protocols[5,6,24] that provide mutual authentication between nodes. Otherwise, the credentials need to be presented from one node to the other, which can only be done securely if the credentials-presenting node has already authenticated the credentials-verifier.

Symmetric keys possess high entropy, and thus a pair of nodes sharing a symmetric key can exchange confidential and/or authenticated messages and establish confidential and/or authenticated sessions. Asymmetric key pairs consist of both a private and public key. The private key should be kept secret by its owner, but the public key could be shared with many other nodes, which makes this type of cryptographic data intrinsically different from both credentials and symmetric keys, which can only provide security between a single pair of nodes (if these data are shared with more than two nodes, the excluded parties to any communication relying on these data could launch passive or active attacks on the communication). Source authentication can be provided to messages travelling from the private key owner to a public key possessor, and confidentiality can be provided to messages travelling in the opposite direction. The private key owner can also authenticate himself interactively to a node possessing the public key.

Pulling this together, nodes can generate cryptographic data and then exchange them with other nodes over secure channels, then leverage these data to add security to vulnerable channels. Credentials or symmetric keys must be exchanged over confidential and authenticated channels; otherwise the adversary could intercept or forge the data, and later on launch attacks. Public keys can be exchanged over authenticated but nonconfidential channels, since secure use of asymmetric cryptography does not depend on the secrecy of public keys. The value of all this is that a secure channel which may be too transient, performance-limited, or costly to use for regular communications can be used to bootstrap security on a more convenient but vulnerable channel.

2.2 Trust

Now it may happen that nodes desire a secure channel, but do not have any physically secure channel with which to bootstrap a cryptographically secure one. In this case, they may have to trust a third party. For example, if Alice wants to send a secure message to Charlie but can't do so directly, she may have to entrust this message to Bob. But just how far do Alice and Charlie trust Bob? If Alice sends the message directly through Bob, and Charlie is willing to believe Bob when Bob says "Alice sent this", then Bob is clearly in a position to read, alter, and forge messages from Alice to Charlie. Alternatively, Alice could give Bob a symmetric key to give to Charlie, then communicate with Charlie herself on a separate channel, but since Bob knows the symmetric key, he could still launch passive or active attacks on this channel, so we won't consider this method as requiring significantly less trust in him. A third alternative is for Alice and Charlie to exchange public keys through Bob. Bob could perform a man-in-the-middle attack here, giving Alice and Charlie false public keys so that he could read and forge their messages, but if he doesn't want to be found out when they try to communicate he will have to launch an active attack where he makes it appear as if the messages were actually processed using the appropriate public keys. If Bob's diligence in these attacks flags, or if Alice and Charlie acquire a communication channel that Bob can't attack, then it is likely that they will discover his perfidy.

So we can roughly say that Bob is either completely trusted (meaning Alice and Charlie accept that he can read or alter their communications without detection), or partially trusted (meaning Alice and Charlie accept that he can only read or alter their communications if he falsifies the key exchange, and that he can only keep this from being detected by launching continuous active attacks on their communications). To clarify the overall picture, we can imagine that each node maintains a table of trust relationships, each of which consists of some cryptographic data and a list of names[3] with associated trust values. For example, Alice might have a trust relationship indicating that Bob's symmetric key is completely trusted for Bob but only partially trusted for Charlie. The trust table contains both primary trust relationships, which the node axiomatically trusts, and derived relationships, which are a consequence of communications that were secured under some other relationship.

For example, when Bob sends Charlie's public key to Alice, Alice will add it to her trust table as a new trust relationship, but if Alice laters discovers that Bob's key was compromised, or decides that she no longer trusts Bob to vouch for Charlie, then the relationship containing Charlie's public key and any relationships derived from it will be invalidated. As another example, asymmetric cryptography is expensive. When Alice establishes a secure session with Charlie, she will likely not encrypt all her messages to him using his public key but might instead verify his signature on a Diffie-Hellman public value as a prelude to establish-

ing a session key, which she will consider as a trust relationship with Charlie for the duration of the session.

2.3 Trust Derivation

The point of the above is that a cryptographic infrastructure can be analyzed in terms of both its static structure of primary trust relationships and the procedures it uses to build and utilize a dynamic structure of derived trust relationships. To make this more concrete we'll examine PKI:

A classic, X.509-style PKI[25,26,27] consists of end-entity nodes and CA nodes. End-entities and CAs have key pairs, and end-entities have trust relationships with certain CAs wherein they partially trust the CAs to vouch for certain other end-entities. CAs express their trust relationships with each other and with end-entities in the form of certificates, which they make available in a directory (which is equivalent to broadcasting them to all end-entities across an unsecure channel). A certificate is an authenticated message from a CA which asserts a trust relationship. This trust relationship contains a public key and expresses complete trust in an end-entity name if it is an end-entity certificate, or expresses partial trust in a set of end-entity names if it is a CA certificate. When an end-entity examines a certificate which comes from a CA whom the end-entity trusts, and references some names which the end-entity trusts the CA to vouch for, the end-entity will add a new, derived trust relationship to its table consisting of the certificate public key and those names. When an end-entity wants to communicate with another, it will search in its trust table for a completely trusted relationship with the other end-entity and use the corresponding public key.

Now since CAs don't participate directly in end-entity communications, all derivation of trust relationships is performed by end-entities themselves. Furthermore, since complete trust is only granted to end-entity public keys whose corresponding private key is assumed to be held by the end-entity himself, Alice has only one choice when she wants to send a confidential message to Bob: she must derive trust in his public key and encrypt the message to it. One theme of this paper is that there are alternative infrastructure choices worth exploring. In particular, imagine replacing the CAs on the trust path between Alice and Bob with online nodes and changing the trust relationships along this path to involve complete rather than partial trust. Alice could still derive a trust relationship with Bob and encrypt to him if these nodes published certificates, but she could just as well encrypt and send the message to the next node along the trust path, with an attached statement that says "please forward this to Bob". If each node did the same the message would get to Bob securely without any derivation of trust relationships at all.

More abstractly: given any network of trust relationships, nodes could always securely communicate with other nodes by relying on the involvement of intermediaries. Derived trust relationships are a way for endnodes to improve this process by relying on intermediaries to securely communicate a trust relationship once which the endnodes can thereafter use directly. This enables partial instead of complete trust in intermediaries when the derived relationship involves public keys, and more generally improves the performance, security, and availability characteristics of an infrastructure since the overhead of constantly contacting the intermediaries is removed, and they are also (at least to some extent) removed as potential points of failure or attack.

From this perspective, the PKI decision to minimize interaction with intermediate nodes seems well-founded. But there are costs associated with end-to-end derivation of trust relationships as well: for one, if multiple end-to-end paths share an intermediate path segment, having each endnode calculate a trust relationship across that segment is redundant. For another, end-to-end derivation requires each endnode to be capable of communicating with every other intermediary in the infrastructure; this level of compatibility might be difficult to achieve in a heterogenous infrastructure, and the amount of communication required could be expensive in a large or geographically dispersed system. Finally, an intermediary node might desire to control an endnode's ability to communicate securely, so as to be able to monitor the endnode's cryptographic activities, and instantly revoke the endnode's access if necessary. This is not possible if trust paths can be derived between the endnode and other nodes that exclude the intermediary. Clearly, weighing all these factors to select and locate trust derivation mechanisms is a challenging job, and much of our argument centers on the claim that conventional PKI has done this poorly.

2.4 Design Methodology

At this point we can suggest a methodology for designing cryptographic infrastructures. First, nodes and channels should be identified, and channels should be classified as either secure enough to use for establishing primary trust relationships or as vulnerable and in need of cryptographic protection. Nodes should then be assigned cryptographic data sufficient to their capabilities: asymmetric key pairs if they are capable of managing these and shouldering the computational burden, credentials or symmetric keys otherwise. We should assume nodes will distribute these data to each other and assign trust in them, thus establishing their primary trust relationships. Finally, mechanisms should be deployed to support the derivation of trust relationships. These should be placed so as to maximize the

gains acquired by removing the involvement of intermediaries while trying to avoid requiring derivations be performed from nodes where this would be difficult or expensive, or where it would remove nodes that we would prefer to remain involved. We will apply this methodology in the next section to analyze the application of PKI to enterprise scenarios and to develop our suggestion for a more effective approach.

3 Delegate Servers

We are using the term 'enterprise' for any organization that has the following characteristics: the organization has a number of members and an administrative entity (or perhaps multiple entities, arranged in a hierarchy); the members would like to have secure communications with each other and with people outside the organization across vulnerable computer networks; the members trust the administrative entity to vouch for the identity of everyone with whom they communicate, and they also trust that it will not read, modify, or forge their communications illicitly; each member has some sort of secure channel with the administrative entity, even if this is only the ability to walk into the administrator's office and talk to him; and the administrative entity has the resources to operate a networked service for the benefit of its members.

In the previous section we presented a methodology for designing cryptographic infrastructures. In the first section we presented a number of criticisms of PKI, and presented an alternative approach using what we called delegate servers. We will now apply our methodology to the enterprise scenario, showing step-by-step the construction of a DS-based infrastructure, and highlighting how it differs from conventional PKI.

3.1 Credentials vs. Private Keys

First, we must identify nodes and channels. Clearly each member of an enterprise is a node, and we will make each enterprise's administrative entity a node as well (or perhaps a hierarchy of nodes, if this reflects administrative relationships more adequately). Computer networks will provide channels between all nodes which are subject to both active and passive attacks, but we will assume that there are low-performance but physically secure channels between enterprise members and administrative nodes, and that there are low-performance channels protected against at least active attacks between certain administrative nodes, whether belonging to the same or different enterprises.

The next step is to assign cryptographic data to nodes. The PKI approach is to give each administrative node a key pair and call it a CA, and give each person a key pair and call him or her an end-entity. Since key pairs are the most powerful form of cryptographic data, and since administrative entities can presumably install their private key once on a high-performance system and be done with it, we concur in giving key pairs to administrative nodes. We will call those administrative nodes that have direct trust relationships with enterprise members DSs, and those which only have trust relationships with other administrative nodes (in a bit of foreshadowing) CAs. As for enterprise members, we reject PKI's assertion that key pairs are the best form of cryptographic data for them. Instead, we feel that authentication credentials such as reusable passwords, one-time passwords, and biometrics are generally easier for the enterprise to deploy and easier for people to use. Moreover, authentication credentials are widely deployed: password authentication systems such as Kerberos[28,29], RADIUS[30], LDAP[31,32], or various other Unix and Windows logon systems exist on most corporate networks, and a vendor recently shipped more than ten million of their one-time password devices[33]. The DS infrastructure will thus assume only that people have some way of authenticating themselves to a service; we will let each enterprise, or perhaps even each person, decide precisely which authentication method they prefer. Since authentication credentials can only provide point-to-point secure sessions instead of point-to-many secure sessions or messages like asymmetric key pairs can, this decision will have substantial ramifications. Before moving on, let's examine it carefully.

To compare credentials against private keys we need to consider both ease of use and security characteristics, since these tend to trade off against each other. For example, a general difference between authentication credentials and private keys is that the former can only be used online and the latter can be used offline as well. Offline support is an occasional convenience for the user, particularly when travelling, but it's an even greater convenience for attackers, since they can steal a private key and use it without generating an audit trail or monitorable events. Since there are ways the DS approach could allow a limited degree of offline operation, we consider the offline exploitability of private keys a serious deficiency. To compare credentials and private keys more closely we will examine three dimensions: portability, universality, and vulnerability. By portability we mean the ease with which a person can carry the data with him. By universality we mean whether or not the data can be used with any computer or whether the computer requires special hardware. By vulnerability we mean how easily the data can be stolen, taking into particular account active attacks on authentication protocols and local attacks by software or hardware on the user's machine. We will consider three authentication methods versus three private key storage methods: memorized passwords, one-time password devices, and biometrics, versus private

keys stored in files, smart cards, and servers.

Memorized passwords are highly portable, universal since their entry only requires a keyboard, relatively invulnerable to active attacks since they can be used with zero-knowledge password protocols[5,6,24] but might be vulnerable to online guessing, and vulnerable to local attacks since the password must be entered and stored in memory somewhere.

One-time password devices[1,2] are reasonably portable, universal since entry of the password only requires a keyboard, invulnerable to active attacks since they can be used with zero-knowledge password protocols and have enough entropy to resist online guessing, and invulnerable to local attacks since the device secret is not exposed to the local computer.

Biometrics[3] are extremely portable, not universal since their entry requires special hardware, vulnerable to active attacks since they cannot be used with zero-knowledge password protocols, and vulnerable to local attacks since the biometric value is typically exposed to the local computer; also, attacks are unusually damaging since once compromised a biometric cannot be changed.

Private keys stored in files are difficult to transport, universal since the file could be copied to any computer, and highly vulnerable to local attack since the private key file can be stolen at any time, not just when the user is performing an operation.

Private keys stored in smart cards are reasonably portable, not universal since not all computers have smart card readers, and reasonably invulnerable to local attack since the private key is contained within the card and probably only vulnerable to hardware attacks such as timing or power analysis[34,35], or glitching[36]).

Private keys stored on servers and delivered to authenticated users are as portable and universal as the underlying authentication technique, and are vulnerable to local attacks.

In sum: storing private keys on the file system is inferior in our metrics to downloading private keys from a server, assuming a universal authentication technique is used. So with private keys, we essentially must choose between a nonuniversal solution reasonably secure against local attack (smart cards) and a universal solution vulnerable to it (server delivery). With authentication techniques, we can choose a solution that is both universal and secure against local attack (one-time password devices). Since there is no single metric or combination of our metrics in which private keys are superior to authentication credentials, since private keys can be exploited offline, and since authentication credentials are already widely deployed, we believe that enterprise cryptographic infrastructures should be designed for people with authentication credentials (including private keys) instead of solely for people who have private keys.

3.2 Delegated vs. End-to-End Derivation

Now that we've assigned cryptographic data we can assume nodes distribute these data to each other along secure channels. In the PKI case that means all nodes share their public keys with those whom they want to be trusted by and then construct primary trust relationships: CAs will trust end-entity keys completely for the end-entity's name, and all nodes will trust CA keys partially for the set of end-entity names over which they consider each CA authoritative. In the DS case the only differences are that people share their credentials-verifying data with their DS instead of sharing their public keys; that if the authentication protocol for a given credential provides mutual authentication, then the person constructs their trust relationship with the DS in terms of that credential instead of the DS's public key; and that all trust extended to DSs is complete instead of partial.

Now that we have a structure of primary trust relationships, we must specify the mechanisms whereby this structure is used to secure communications. In a PKI system CAs publish their primary trust relationships as certificates, then do nothing further. End-entities retrieve these certificates and use them to derive trust relationships with other end-entities' public keys which are then used as inputs to cryptographic algorithms and protocols. We earlier criticized this approach because it exposes end-entity software to the specifics of all the PKI systems and data formats in a potentially large and heterogenous infrastructure, and because it requires the distribution of significant amounts of certificates and revocation data, and redundant path computations upon these; below we elaborate on these points:

To address the first point: a network of primary trust relationships is similar to a routing network, in that each link (i.e. cryptographic datum) is labelled with the addresses (i.e. names) which can be reached through it. The requirement that end-user software must construct a complete trust path is analogous to a requirement that each host network card not only compute the entire route for its packets, but understand each link-layer protocol along that route. This violates modularity by making every point on the edge of a network dependent upon every link in the network. The result is either going to be that new PKIs are prevented from joining the system and changes are not made in an effort to keep the system homogenous so that end-user software will continue to function, or alternatively every modification will force painful software upgrades on the user population; we earlier called this situation one of rigidity and brittleness. We will enumerate the dependencies that cause this brittleness below:

To construct a certificate path requires retrieving certificates from one or various directories, com-

prehending the syntax and semantics of each certificate, applying a search algorithm appropriate to the PKI's trust topology to these certificates to determine candidate or partial paths, applying a revocation or validity-checking method to each certificate along such a path to determine its validity, and iterating the above processes until a complete and valid path has been built. This process is thus dependent upon knowledge of directories and directory protocols (such as X.500[37], LDAPv2[31], LDAPv3[32], or various proprietary protocols); certificate formats (such as X.509[25], OpenPGP[18], or SPKI[38]) and their semantics (how they express identities, attributes, and authorizations) and their many different versions, profiles, policies, extensions, unique identifiers, signature algorithms, encoding quirks, etc.[39]; trust topologies (such as bidirectional hierarchies, top-down hierarchies, hybrids such as bridge structures, meshes, etc.); and revocation/validity-checking methods (such as CRLs[25,26], segmented CRLs[25,26], delta-CRLs[40], sliding-window delta CRLs[41], OCSP[42], DPV/DPD[43], etc.).

And these are only the complications that pertain to path construction; if we consider the management protocols between an end-entity and CA we find a similar raft of competing, complex, and variably-implemented protocols[44] such as PKCS #10[45] with SSL[4], PKCS #10 with PKCS #7[46], CMP[47], CMC[48], and SCEP[49]. In some of the above cases we can expect convergence and stability as winners are chosen and interoperability is pursued, but in other cases, particularly those involving certificate semantics and revocation strategies, there are substantial unresolved theoretical issues. In an environment of such turmoil and complexity we can't expect end-entity software that supports anything but the simplest or most homogenous PKI deployments to exist for some time.

Even disregarding this complaint, end-to-end path construction suffers from being inefficient. Many users, particularly within an enterprise, will share the same trust anchor points and validation policies, and thus the certificate paths they construct will be identical. Path construction is expensive because it requires retrieving certificates and revocation information from a potentially wide variety of sources and performing expensive signature verifications and revocation look-ups (or online status checks) upon these. Performing revocation checking on end-user certificates is particularly taxing[50]: end-users generally have transient relationships with their certificate issuer (people get hired, fired, demoted, etc.), and their private keys are highly vulnerable to compromise or loss (people lose their smart cards, forget their PINs, store their private keys on unprotected computers, etc.). A large volume of revocation information will thus be generated, and since the costs of relying on a compromised private key are high, end-entity software must frequently download the latest revocation lists. In the case of a complex trust topology, these costs are magnified: path construction is not deterministic but requires sophisticated graph exploration algorithms that can ignore dead-ends and detect cycles; even if these algorithms work flawlessly they may have to retrieve and consider a large number of certificates before determining a path.

So for reasons of overcoupling and inefficiency, we feel that PKI's reliance on end-user software which processes messages from each CA to derive end-to-end trust relationships is a poor design choice. A better approach would be to have administrative nodes derive trust relationships. For one thing, an administrative node could derive these relationships once and then reuse them. For another thing, administrative nodes are assumed to be under the control of a competent staff who can configure them and upgrade their software quickly, thus managing the complex coupling with the infrastructure that trust derivation requires. Also, there will be many fewer administrative nodes in an infrastructure than end-user nodes, so having to configure and maintain them is less of a burden. The administrative nodes that end-users have direct access to we have called DSs. Since end-users must have access to these trust-derivations to make use of them, it follows that DSs are where trust derivations should be performed, or at least made accessible.

Now it should be noted that end-to-end derivation of trust relationships without online interaction with intermediaries is impossible in a DS infrastructure, because of the structure of primary trust relationships: end-users do not have public keys which can be communicated widely, instead they have authentication methods which can only be used to provide secure sessions with their DS. Thus, trust derivation will not only begin at DSs, it must terminate at them as well, since it is impossible to construct a path all the way to an enterprise member.

So we have located trust derivation; now the question is how to perform it. Obviously we should use PKI! Our criticisms of PKI are that it requires private keys, that it requires a tight coupling between end-entities and infrastructure, and that it is inefficient when deployed to a large community of users with a high revocation rate. But these criticisms don't apply to DSs: for one, DSs already have key pairs, and they are maintained by a staff which can manage the challenges involved in software that is tightly coupled to an infrastructure. Also, there are many fewer DSs then there are end-users, and DSs are much less likely to be compromised than end-users, or to have a volatile relationship with the enterprise, so revocation rates will be lower. Path construction will thus occur in a much smaller and more stable environment, and in this environment the advantages of PKI come to the fore:

namely, communication with and trust in intermediaries is minimized. Given that cross-enterprise trust relationships will span the public Internet, where traffic analyzers and denial of service attackers may be lurking and where occasional traffic outages and performance lags occur; and given that cross-enterprise trust paths might involve any number of intermediaries (governments, industry consortiums, public CA maintainers, etc.), there is clearly value in reducing communication with these nodes and trust exposure to them.

Within the enterprise, it's a completely different story. Private keys are burdensome for people, desktop software is so widely deployed that complex configurations and upgrades are difficult, enterprise networks are reliable, high speed, and sheltered from grosser forms of abuse, and though DS involvement adds a new point of attack and failure to the infrastructure, it also adds a point of monitoring for the first point of attack (the user's credentials). Under the reasonable assumptions that DSs are highly trusted, that credentials are highly vulnerable, and that monitoring substantially increases the ability to detect compromises, determine their extent, and trace their source, this results in a net increase in security.

3.3 Delegate Server Infrastructure

Pulling this all together: we are proposing that DSs should possess key pairs and function as end-entities in a PKI, and that enterprise members should authenticate to DSs to secure their communications. An efficient way to do this is for enterprise members to perform bulk cryptographic operations involving symmetric keys and hash values, then call out to DSs only to perform asymmetric cryptography upon these. We will discuss protocol details in the next section. For now, we should view the result as simply PKI applied at the granularity of enterprises instead of individuals: since functioning as a PKI end-entity is difficult for both the user and his software, and since PKIs with large numbers of volatile and vulnerable end-entities have performance problems, we are applying PKI only at a high level, and extending security down to the user level with simpler and more user-friendly authentication mechanisms. Alternatively, we can view DSs as simply an aggregation technique: a DS allows us to treat a group of user nodes as a higher-level enterprise node from the perspective of a higher-level cryptographic infrastructure. Though we've claimed this higher-level infrastructure should be PKI, this is by no

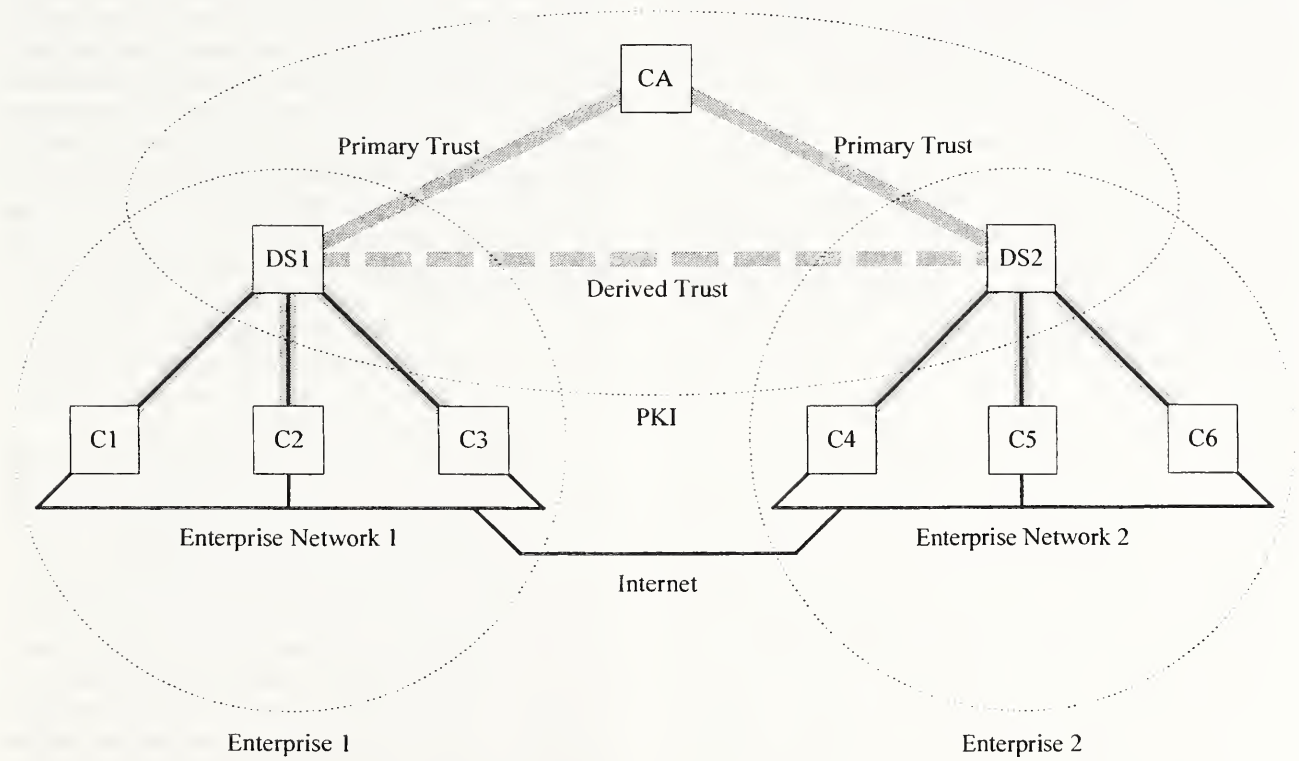


Diagram. An example infrastructure showing two enterprises, each with a single DS and three clients. The thick gray lines represent trust relationships: primary trust relationships link DSs and clients, and DSs and CAs. The DSs construct paths to each other to establish a derived trust relationship, shown as a broken gray line. The thin black lines represent network communication channels. Clients communicate with their DS and with other clients in the same enterprise using the enterprise network. Clients communicate across enterprise boundaries using the Internet.

means a foregone conclusion, and by no means the only way we could knit DSs together. If end-user software treats DS messages as opaque blobs of data, then we can retain some flexibility for DSs to support whichever algorithms, formats, and alternative infrastructure strategies they desire. For example, when asked to encrypt a symmetric key to another DS, a DS could instead add a unique identifier to the key and forward it over a secure channel to the other DS, then return this identifier to the client as if it was the encrypted key.

From a user's point of view, a DS infrastructure would be simple yet effective. Alice could sit down at any computer that was inside her enterprise's network (or perhaps even outside it), configure the computer with the address of her DS, and then, with nothing more than her password, achieve confidential and mutually authenticated communications using email, file transfers, instant messaging, remote login, web browsing, videoconferencing, etc.. From an administrator's perspective a DS would be reasonably easy to manage.. The chief task would be to set up primary trust relationships with users, CAs, and other DSs, and to configure the DS with whatever knowledge of PKI systems was necessary to compute paths to other DSs and end-entities (perhaps knowledge of remote directories, certification policies, etc.). The chief ongoing maintenance would be the periodic replacement of DS key pairs and user credentials, and detecting and responding to credentials compromises.

3.4 Security Criticisms

Several criticisms could be directed against DSs. For one, authentication methods like passwords are generally considered much weaker than private keys[51,52], but DSs would use strong authentication protocols[5,6,24], forcing attackers to contact the DS to verify each guess. A DS could detect and rebuff guesses by slowdown or lockout mechanisms,. and could deter them by attempting to trace the attacker and respond. Using randomly assigned passwords or enforcing a password-strength requirement should give adequate security for many scenarios. For high security, one-time password devices can be used. Since authentication credentials are easier to use than private keys and thus easier to deploy widely and since abuse can be detected and monitored, we believe that in most scenarios our approach is more secure than giving users private keys.

We have made much of the ability of DSs to detect and manage compromises. We will give an example of how this might work: Alice would receive a weekly usage statement from her DS. For each private key operation Alice had performed, the statement would list the date, time, IP address of the requestor, and perhaps a plaintext string generated by Alice's end-user software saying things like "Signing document ContractForBob" or "Decrypting message from Charlie with subject 'Meeting notes'". Alice would review this usage statement and compare it against a local log of her activities; if she noticed illicit uses she would contact the DS and repudiate them, then cancel her credential and begin the process of getting a new one. Alice's DS could proactively notify the DSs that were on the other end of any forged message authentication codes, faked sessions, or illicitly decrypted messages, or perhaps could publish a revocation list containing these which other DSs could check, and if one of them noticed that one of its users' communications was affected, then this DS would notify the relevant user by email or phone that certain of his communications had been compromised. A DS could also monitor requests in real time to detect suspicious patterns of activity such as users performing operations during nonbusiness hours or when on vacation, or the decryption of a tripwire message or authentication with a duress code; when suspicious activity was detected the DS could prompt the user for a backup password, lock him out, pretend to be unreachable, trigger an alarm, generate a warning message to the user's email address, etc..

This should be contrasted to how a PKI handles compromise or loss of a private key. If Alice's private key is stolen covertly, the attacker could decrypt all messages encrypted to her and could authenticate or forge her signature without her receiving any indication that this is occurring. If she does begin to suspect something, she will have no way of knowing when the attack occurred and how much data was compromised. To be safe, she might have to repudiate everything ever signed under the private key, and consider all messages encrypted to it as compromised. Since PKI users typically have long-lived key pairs, this compromise could affect a year or two's worth of data. In sum, compromise of a private key is catastrophic in PKI; when using DSs, compromise of a credential can be quickly detected, limited, and recovered from.

Another security criticism is that DSs provide a high-value target for attacks. This is true: DSs are completely trusted, not partially trusted like CAs. If a DS was compromised, its private key could be used to decrypt any messages encrypted to its users, forge any of their signatures, or fake any of their authentications. To reduce the risk of cryptanalytic attack, DSs should be given large key pairs[53], and should change them frequently. DSs should also destroy old private keys once these reach a certain age, so a compromise of the entire DS will only expose a limited amount of traffic; this policy could impact the availability to users of their own data; we will look at ways to mitigate this shortly. The impact of compromise would also be lessened if users availed themselves of protocols with perfect forward secrecy[54,55] when possible.

As a matter of prudence DSs should be kept in a physically secure location, and their private keys and credentials-verifying data should only be handled within a tamper-resistant secure coprocessor[21]. This may not keep them safe from someone willing to take the DS offline and who has the technical resources and time to dissect the module and defeat its safeguards, but it should prevent covert theft of cryptographic data by attackers with intermittent physical access. When all these steps are taken (large keys with short lifetimes that are deleted at a certain age, physical protection of DSs, hardware protection of cryptographic data), we believe the security advantages offered by DSs outweigh their disadvantages. We also believe that it is not possible to significantly reduce trust in a DS while preserving its advantages; some private key delivery servers store private keys encrypted by user passwords so as to hide these keys from the server itself, but the server could always launch an offline attack and probably recover any user's password and private key.

3.5 Performance and Privacy Criticisms

A third criticism is that DSs are a potential performance bottleneck because of the computational cost of asymmetric cryptography. Given that computers are fast and getting faster, and that accelerator cards costing less than two-thousand dollars currently measure their speed in thousands of 1024-bit operations per second[56], we think DSs could offer adequate performance for a reasonable price. Moreover, centralizing computation has the advantage that weak client devices such as PDAs, cellphones, chat devices, web appliances, etc., would not have to perform the calculations themselves; they could authenticate once to the DS using lower bit length or elliptic curve calculations[57,58], then request multiple different operations in a single DS session (decrypting dozens of messages, encrypting and signing others, etc.).

Regardless, if the cost of constantly performing asymmetric operations was deemed too high, a community of DSs could use Diffie-Hellman key pairs. These key pairs would allow each DS in every pair to perform a single asymmetric calculation to determine the shared symmetric key it has with its partner, which it could then cache and reuse. With the appropriate protocols and data formats these symmetric keys could be used for point-to-point security; they could not be used for signatures (i.e. point-to-many message authentication) or anonymous encryption, but in a community not requiring these operations the use of Diffie-Hellman key pairs could eliminate asymmetric operations except for periodic key agreements.

A fourth criticism is privacy. A DS can monitor all operations performed by its users, and thus can harvest data about whom users communicate with and when, what their work habits are, etc.. This is unavoidable; we can only hope the security benefits of monitoring and the ease of use benefits of DSs overwhelm this deficiency. Theoretically, users could contact DSs using pseudonyms and untraceable channels, and could distribute their operations under multiple different names and DSs, and perhaps even blind[59,60] the data they send to a DS so that it could not correlate user operations with intercepted communications, but users capable of all this probably don't need DSs in the first place. A more feasible approach to privacy might be for a secure coprocessor at the DS to encrypt audit trail records so as to keep them hidden even from the DS administrator, and only make the relevant records available to authenticated users[22].

3.6 Communications Criticisms

The most serious criticisms involve the online communications required between users and DSs. The channel between a user and DS may be subject to denial of service attacks or traffic analysis[20], and may suffer performance or availability problems. Inside an enterprise network we believe these are minor issues. On the Internet, DSs should not be used for important time-critical communications, since an attacker or transient network failure could render the DS unreachable for a period of time. Internet DSs should also not be used when extremely high performance is required, or when the mere fact that a communication is occurring needs to be hidden. Anyone designing DSs for Internet use should take extreme care that the protocols are not exploitable and that they offer a measure of resistance to denial of service attempts (by using stateless cookies, for example[61]). Anyone deploying DSs on the Internet should take measures such as deploying redundant DSs in separate locations with high-speed network access and ensuring a competent staff is on hand to ward off crises.

Another concern with online communications is that users cannot secure new communications or read encrypted old ones when they are in environments without network access. One partial solution to this, and to the latency DSs introduce into asymmetric operations, is a persistent cache of symmetric keys and hash values stored on the user's machine. Whenever a DS is contacted to perform a decryption or verification, the result would be stored in this cache, and from then on every time the user opened the same file or read the same email these values would be fetched locally. The cache would be encrypted, so that whenever a user logged onto his computer he would have to contact the DS to decrypt it. This would keep data in a stolen computer secure and help detect illicit access (for example, by a coworker who knew your password and read your emails while you were at lunch).

If you knew you were going to be offline for a period of time, you could leave the cache in a decrypted state so that you could continue to access secure emails and files without DS contact. Another benefit of this cache is that it would keep alive messages that were encrypted to private keys that the DS had expired. To ensure that the cache itself does not expire, it could be encrypted at the DS under a special long-term key, or alternatively, every time the user contacted the DS the cache could be re-encrypted under the DS's current private key. To reduce the dangers of cache compromise, users should be able to review and purge the contents of their cache. This cache would not allow users to perform new operations when offline, but since the user is offline and thus presumably not in a position to communicate anyways, we consider this acceptable.

3.7 Delegate Server Interoperability

Despite all our arguments, DSs are inappropriate for some environments. If a user does not completely trust anyone but himself, if offline operation is important or constant contact with a DS is too vulnerable, inefficient, or unreliable, or if no administrators are willing to maintain a DS for this user, then he will have to possess his own key pair and interact with PKI on his own. Interoperability between non-DS and DS users is assured because a non-DS user can be viewed as merely a special-case DS: a DS's key pair is used on behalf of many people; a non-DS user's key pair is used on behalf of only one (himself).

In some environments, users may wish to use DSs for some operations but not all. For example, a user may wish to manage his own key pair but use a DS for public key operations (i.e. encrypting and verifying), thus freeing himself from the burden of path computation. Alternatively, he might feel more secure using the DS's key pair, but prefer to establish trust relationships with others himself.

3.8 Alternatives

Our criticisms of PKI are not novel. Various proposals have attempted to address the vulnerability and inconvenience of private key transport and the difficulty and expense of path construction. One approach is not to use PKI at all, but to use an infrastructure like Kerberos[28,29], which is entirely based around symmetric key trust relationships. We feel asymmetric cryptography has significant advantages in minimizing the trust and availability requirements placed on infrastructure nodes. However, there is a proposal to use asymmetric cryptography for cross-realm authentication[62] in Kerberos which would realize these advantages but still allow users to authenticate to their local server using passwords. The resulting hybrid infrastructure is quite similar to what we are proposing. The difference is that Kerberos only supports the establishment of symmetric keys between clients, whereas DSs allow clients to perform asymmetric operations such as signatures or anonymous encryptions. Also, whereas Kerberos requires users to operate under an online server, DSs are optional, and DS clients could seamlessly interoperate with conventional PKI users.

Turning now to proposals for improving PKI, one approach to private key transport that we have already mentioned is the use of private key delivery servers[16], which make the private key more transportable but leave it vulnerable to theft and offline abuse. Another approach is to use proxy certificates[63], which are issued under a user's regular certificate or under another proxy certificate but are only valid for a limited period of time and for a restricted set of uses. The generation of proxy certificates could be performed by the user on his local machine or by an online service that issues proxy certificates to users under proxy certificates that users had previously issued to it[64]. The advantage of the proxy certificate approach is that long-lived end-entity private keys can be kept in a highly secure environment while the more exposed proxy certificate private keys are given limited validity periods and privileges so as to minimize the damage done by a compromise. Nonetheless, like any approach that gives users control of private keys, the security benefits of auditing and instant revocation are not available; in addition, the transient nature of proxy certificates makes them unsuitable for message confidentiality. Short-lived certificates issued by online CAs[65] have the same disadvantages, but eliminate the need for long-lived end-entity certificates, while requiring greater trust in the online service since it possesses a long-lived CA private key instead of short-lived proxy keys issued to it by various users.

An approach more like ours is the use of virtual smart card servers[66], where each user's private key is stored at a server which the user authenticates to and requests operations from. These servers provide the same portability, auditing, and message confidentiality benefits as DSs. As for path construction, protocols like DPV/DPD[43] and XKMS[67] have been proposed to allow clients to offload path construction to servers, and we will assume that these work adequately.

Now if virtual smart card servers in conjunction with path construction servers accomplish the same things as DSs but work with current PKI protocols and data formats, isn't that good enough? Why bother to add explicit support for DSs? For a few reasons: For one, storing end-entity private keys on a server abuses certificate semantics: someone verifying an end-entity signature will have no way of knowing that the corresponding private key was actually in the possession of a third party. This is a significant fact and should be

somehow represented. For another, a virtual smart card server requires a separate certificate for each user; exchanging and updating these is inefficient and will reveal much information about enterprise members, including their affiliation with the enterprise, their contact information, and the revocation status of their private keys (which an attacker can check to determine whether his compromise of a private key has been detected, for example). A DS would need only a single certificate to represent an entire enterprise, and wildcards within the name forms (such as DNS names, IP addresses, telephone numbers, etc.) would not reveal anything about the enterprise's internal structure.

Another problem with current PKI technologies is that you can only revoke keys from a particular date and time, you cannot revoke particular operations. A DS could allow the user to sift through audit trails after a compromise and revoke private key operations on a fine-grained basis. Another advantage of DSs is that if DS messages are treated as opaque by clients, then DSs acquire significant flexibility in terms of algorithms, certificate formats, etc., and clients are shielded from these details. For all these reasons (improved semantics, fewer certificates, fine-grained revocation, shielding client software from infrastructure), we believe that it is worthwhile to insert DS support into current PKI protocols and data formats. We will turn our attention to this in the next section.

4 Protocols and Data Formats

Below is an example protocol demonstrating DS-secured messages. We assume all communications between clients and DSs are mutually authenticated and confidential.

C1, C2: end-users
S1, S2: Delegate Servers for the respective end-users
D1, D2: Delegate Servers' private keys (RSA-like)
E1, E2: Delegate Servers' public keys (RSA-like)
k: symmetric encryption key
m: message
h(): hash function
k(): symmetric encryption function
D1(): asymmetric signature function
E2(): asymmetric encryption function

Signed Message
C1 → S1: h(m),C1
C1 ← S1: D1(h(m),C1)
C1 → C2: m,D1(h(m),C1)
C2 → S2: h(m),C1,D1(h(m),C1)
C2 ← S2: true|false

Encrypted Message
C1 → S1: k,C2
C1 ← S1: E2(k,C2)
C1 → C2: k(m),E2(k,C2)
C2 → S2: C2,E2(k,C2)
C2 ← S2: k

Signed and Encrypted Message
C1 → S1: h(m),C1,k,C2
C1 ← S1: D1(h(m),C1),E2(k,C2)
C1 → C2: k(m,D1(h(m),C1)),E2(k,C2)
C2 → S2: C2,E2(k,C2)
C2 ← S2: k
C2 → S2: h(m),C1,D1(h(m),C1)
C2 ← S2: true|false

If the sending and/or receiving clients were not using DSs, the messages sent between clients would be the same but the asymmetric keys D1 and E2 might refer to the sender's private key or the receiver's public key instead of to the corresponding DS keys, and clients could perform the processing themselves without engaging DSs. In fact, clients could always perform public key operations (i.e. encrypting and verifying) without engaging their DSs, so we should be very clear that only private key operations are rigorously auditable.

When DSs are employed, clients need only a minimal understanding of the DS data blocks. From the perspective of client software, the protocol looks like:

Signed and Encrypted Message (client perspective)
C1 → S1: h(m),C1,k,C2
C1 ← S1: X,Y
C1 → C2: k(m,X),Y
C2 → S2: C2,Y
C2 ← S2: k
C2 → S2: h(m),C1,X
C2 ← S2: true|false

This gives the protocol a pluggable structure, allowing the DS blocks to change without affecting client software (to incorporate a new asymmetric algorithm or data format, for example).

4.1 Operation Certificates

To emphasize the differences between the DS and non-DS approaches, consider what a non-DS signed and encrypted message would look like:

Signed and Encrypted Message (without DSs)
C1 → C2: k(m,D1(h(m)),E2(k)

First, since the asymmetric keys uniquely identify end-entities, there is no need to bind sender or receiver names into the data format. Second, there are obviously no sideband protocol exchanges with DSs. When retrofitting DS support, then, we must determine some standard representation of the DS signature and encryption blocks X and Y which will allow us to represent operations and the names they apply to in a packaged format whose processing can be delegated to DSs.

One way to address these tasks is to embed names into the data structures used to represent signed hashes or encrypted keys. A name that was bound to a signature would be a declarative statement from the signer about whom the signature was produced on behalf of. A name that was bound to an encryption would be an imperative statement to the encryptee about whom the encrypted data is destined for.

For example, to produce a delegated XML Signature[68], a client could authenticate to his DS and forward it a ds:SignedInfo containing the hash values the client would like signed. The DS would add a SAML[9] Authentication Assertion as a ds:SignatureProperty, then sign the resultant ds:SignedInfo and return a ds:Signature to the client. The Authentication Assertion would name the client, the authentication method he used, and advice or conditions relating to these. To validate an XML Signature using a DS, a client could validate each hash within the ds:SignedInfo himself, then forward the ds:Signature structure to his DS and receive back a boolean.

To produce an XML Encryption[69] targeted to a DS client, one would encrypt some data with a symmetric content encryption key, then generate an XACML[70] xacml:policyStatement that expresses to whom you would like the content encryption key delivered, and encrypt this xacml:policyStatement with a symmetric policy encrytion key. Then one would generate a symmetric key encryption key and encrypt both the content and policy encryption keys with it, and finally encrypt the key encryption key with the DS's public key. The client who received all these data would authenticate to his DS and forward them to it. The DS would recover the content encryption key and the xacml:policyStatement, then evaluate the policy statement against the client's Authentication Assertion, and return the content encryption key to the client if access is permitted. The use of an access control language like XACML would allow the sender to specify the intended recipients in sophisticated terms (i.e.: "give this key to Bob or Carol, but only if they have a Top Secret clearance, and only if they authenticate with a hardware token").

To verify a DS-produced signature, or encrypt to a DS client, one must be capable of determining the DS public key. We could incorporate a flag into a ds:KeyInfo to represent DS public keys, and perhaps even add an xacml:policyStatement into a ds:KeyInfo to express to whom and for what the key should be used for. Someone wanting to encrypt a message to Bob could perform an XKMS[67] query and receive back an explicitly flagged DS key, and would thus know to embed an xacml:policyStatement into the encryption to represent the intended final recipient.

But XKMS is only intended as an interface to PKI, so this raises the question of how we can represent DSs within PKI data formats. For example, we would need to modify the X.509 certificate format to support DS certificates as opposed to CA or end-entity certificates. DS certificates would be like CA certificates in that they are authoritative over some group of users (they should support the name constraints extension to express which group), but like end-entity certificates in that the corresponding private key can perform signatures or be encrypted to directly. We could add a boolean into the basic contraints extension to identify DS certificates (which might also be CA certificates, allowing a DS to not only perform operations for clients, but perhaps issue these clients short-lived certificates).

We might also wish to retrofit DS support into PKI protocols that don't use XML, such as SSL[4] or S/MIME[71]. As we recall, adding DS support requires a standard format for representing signature and encryption operations with names bound into them. In XML there are standard formats for representing signatures and encryptions which we could easily add names into. In the X.509 PKI world there are not; however, instead of binding names into operations, we can add operations into bound names. In other words, we can generalize the notion of certificates so that instead of only binding names to public keys, certificates can bind names to hashes as well, and thus represent delegated signatures, and we can also invert the notion of signed certificates to yield encrypted certificates, which are imperative requests that a binding should be made to exist in the future, instead of declarative assertions that a binding did exist in the past.

In more detail, consider an X.509 end-entity certificate. Typically such a certificate is said to bind a name to a key. In truth, it binds not only a name, but also a serial number so the certificate can be referred to later and possibly revoked, a validity interval which delimits the binding in time, and a policy which clarifies the binding's semantics. And when we say that these things are bound to a key, we really mean that they are bound to the particular operations performed by this key: that is, that they are attributes of the signatures which it verifies and the encryptions it can be used to produce. In other words, an X.509 certificate is a mechanism for binding (within the limits of a validity period and policy) an end-user name and a serial number to operations as expressed through the indirection of

a public key.

It seems logical, then, to use certificates to bind these same attributes directly to particular operations. For example, consider an end-entity who wants to sign a document with his private key. He could hash the document and then collect this hash along with a serial number, a validity interval, and a policy, and then use his private key to sign these, producing a signature operation certificate (OC). The serial number would allow him to later revoke this particular signature by including its number in a revocation list. The validity interval would allow him to represent the time period over which he is asserting this binding. The policy would allow him to express the particular semantics of his signature on this document. Someone verifying this signature should validate the entire certificate chain, including first the CA certificates, then the end-entity certificate, and finally the OC, before extracting and checking the hash value inside the OC.

An encryption OC would be similar to a signature OC but would contain a symmetric encryption key instead of a hash value, and would be encrypted to the target's public key, instead of signed by the issuer's private key. The policy identifier would identify a request instead of a statement: that is, instead of a statement from the signer saying "I authenticated Alice to degree X and assume liability Y for the assertion that this data is associated with her", it would say "Please authenticate Bob to degree X and only deliver this data to him if you are willing to assume liability Y". One difference between signature and encryption OCs is that signature OCs represent past occurrences, whereas encryption OCs represent conditions on future occurences (mirroring the distinction between SAML assertions and XACML policies). Thus while signature OCs would be similar to end-entity certificates in that they bind a particular name, encryption OCs would be like CA certificates in that they might bind a range of names (using the name constraints extension), representing all the users who would be allowed to decrypt this data.

Looking back at our protocol diagrams, the D1(h(m),C1) blocks represent signature OCs, and the E2(k,C2) blocks represent encryption OCs. The chief problem with OCs is that they don't yet exist: current cryptographic protocols and data formats such as CMS[72] (used by the S/MIME email security standard) or TLS[73] (derived from SSL) would need surgery to support them. Below we will consider exactly what this entails.

A signature OC will be mostly identical to an end-entity OC except that the issuer field will refer to the end-entity certificate or DS certificate that issued it, and the subjectPublicKeyInfo field will be replaced by digestAlgorithm and digestValue fields. An encryption OC will be a little different; in particular, its top-level structure will be something like this:

```
EncryptionOperationCertificate ::= Sequence{
  encryptedCertificate  EncryptedCertificate
  encryptionAlgorithm AlgorithmIdentifier
  encryptedKey          BIT STRING
  target                TargetIdentifier }
```

Instead of this:

```
Certificate ::= Sequence{
  tbsCertificate        TBSCertificate
  signatureAlgorithm  AlgorithmIdentifier
  signatureValue        BIT STRING}
```

It may be desirable to support signature and encryption OCs that have both an issuer and a target, so that hash values and encryption keys could be transmitted securely using key agreement algorithms, and this could be done with straightforward extensions to the EncryptionOperationCertificate.

Clients could create and process OCs on their own or by authenticating to DSs and engaging in a request/response protocol. We could use TLS for confidential and authenticated session establishment, and modify it to support SRP for mutual authentication[74] between the client and server. The request/response protocol would allow clients to request signature OCs along with the certificate chains leading up to them by sending hash values and to-be-signed attributes to DSs, and to request encryption OCs by sending symmetric encryption keys and the names of intended recipients to DSs. We would also want to allow clients some input into the validity and policy fields of the OCs, and allow clients to retrieve the certificate chain up to their DS in a separate step from procuring an OC, for use in protocols where one party sends a certificate chain to a second who then encrypts something to the first's certificate (such as TLS). To process OCs (i.e. to verify signatures and extract symmetric encryption keys) would involve similar protocol exchanges.

OCs would work with revocation-checking mechanisms such as CRLs and OCSP. The issuer (whether an end-entity or DS) would be capable of revoking signature OCs, and the target (whether an end-entity or DS) would be capable of revoking encryption OCs. Reason codes should be added that are suitable for use by DSs and end-entities. For example, DSs should be able to specify that an operation was revoked because it was accessed using stolen credentials. Revocation-checking of OCs would not need to take place for online operations where timeliness was guaranteed (such as verifying a signature OC on a nonce). For operations where the overhead of retrieving and checking CRLs is too great, revocation-checking can be deferred and done periodically: for example, a DS might download all CRLs only at midnight every day and then

compare them against its audit logs to determine if any of its users were affected. For point-to-point operations (i.e. operations involving key agreement, or where signature and encryption OCs have been cryptographically linked in some way), the DSs can notify only the affected parties instead of having to make the revocation public.

4.2 Using Operation Certificates

Finally, we need to add OCs into application protocols and data formats. These formats already have ways of representing signed hashes and encrypted keys, and we will simply replace these older representations with the corresponding OCs. For example, a CMS SignerInfo could be changed from something like this:

```
SignerInfo ::= Sequence{
  version              CMSVersion
  sid                  SignerIdentifier
  digestAlgorithm      DigestAlgorithmIdentifier
  signedAttrs          SignedAttributes
  signatureAlgorithm   SignatureAlgorithmIdentifier
  signature            SignatureValue
  unsignedAttrs        UnsignedAttributes}
```

To this:

```
NewSignerInfo ::= Choice{
  oldSignerInfo        SignerInfo
  opSignerInfo         OperationSignerInfo}

OperationSignerInfo ::= Sequence {
  version              CMSVersion
  signOpCert           SignatureOperationCertificate
  unsignedAttrs        UnsignedAttributes}
```

The sid, digestAlgorithm, signatureAlgorithm, and signature fields would all be replaced by the signature OC, and the signed attributes could be incorporated into the OC as extensions. To add DS-based encryption to CMS, we could extend the RecipientInfo type with:

```
OperationRecipientInfo ::= Sequence{
  version              CMSVersion
  encryptOpCert        EncryptionOperationCertificate}
```

To add DS support to TLS we could similarly replace the Signature structure with a signature OC and replace the EncryptedPreMasterSecret with an encryption OC. On these lines, we believe any public key protocol or format (such as ssh[75], IPsec[76], OpenPGP[18], etc.) could be retrofitted to use OCs.

In sum, OCs are a powerful primitive even apart from DSs. OCs extend certificate validation to the level of particular operations, allowing policies and validity periods to be bound to operations, signed and encrypted attributes to be incorporated into them, and revocation-checking to occur upon them. By defining a standard structure that uses asymmetric keys to secure this information and bind it to hash values and symmetric keys, protocol designers are given a higher-level building block that makes their job easier. With DSs, OCs become even more valuable, since OCs allow the binding of names to particular operations and can be easily passed back and forth between clients and DSs and embedded in protocols.

It may be objected that we are abusing the notion of certificates, but we feel that we are generalizing it in a coherent way. A conventional certificate authenticates a binding between attributes such as names and a public key and qualifies this binding via policies, validity periods, etc.. This public key can then be used to produce authenticated or confidential bindings between these attributes (or some subset of them) and further data. In the case where an authenticated binding is produced between attributes and another public key, this is called a certificate.

In our opinion, this is a restrictive notion of certificates: the idea of a qualified binding between attributes and data is sufficiently important and general that the same data format and terminology should be used when binding attributes to data that are not public keys (i.e. OCs) and when producing confidential instead of authenticated bindings (i.e. encryption OCs versus signature OCs). By treating all such bindings consistently, the scope of concepts such as revocation-checking, policies, and validity intervals is increased, and the bindings are packaged into a standard format which makes it easy to reuse them in the context of different protocols and easy to delegate their processing to DSs. This approach seems promising, but it clearly needs a much more thorough analysis and explication then we have provided here.

5 Conclusion

Delegated cryptography splits the problem of end-to-end security into an intra-enterprise portion that can be addressed with authentication techniques and an inter-enterprise portion that can be addressed with PKI. This exploits the strengths and avoids the weaknesses of both technologies: Authentication techniques are easy to use and widely deployed, but can only secure interactive sessions between two parties. PKI can secure sessions or messages between a large number of parties, but imposes complex and difficult burdens on these parties. By using authentication techniques to access a PKI-enabled server we can confine the burdens of PKI to a single point within an enterprise while making its benefits available throughout.

There are proposals to improve authentication techniques by having one authentication stand in for several (single sign-on), and to improve PKI through piecemeal delegation of various functions (private key storage, path construction, etc.). We believe these proposals are in the right direction but don't go far enough. We think authentication should be used to access more than simply further authentications, and that delegation should be pushed to its logical extreme. Taken together, these points indicate an infrastructure that would be easy to use, easy to write software for, full-featured, highly secure, and efficient, and could be built on top of the data formats and protocols in use today. We encourage and hope to participate in further research in this direction.

Acknowledgements

We thank Sayan Chakraborty and the anonymous reviewers for their encouragement and helpful comments on the ideas and organization of this paper.

Notes

[1] Actually, we could expect much more from a cryptographic infrastructure, and from cryptography in general: we might want notary, timestamping, and nonrepudiation services, protocol support for things like voting, simultaneous contract signing, and digital cash, steganography and watermarking functionality, anonymous communications, etc.[77]. Here we focus on the more prosaic objectives of confidentiality and authentication, but it would be interesting to explore more exotic uses of DSs.

[2] Much of our argument against conventional PKI, and our proposed solution, was anticipated by Don Davis' paper "Compliance Defects in Public-Key Cryptography"[78]. In particular, after reviewing PKI's advantages in reducing trust, availability, performance, and reliability demands on the infrastructure, he points out that "these attractive features come at the cost of transferring corresponding burdens onto users". His suggestion, similar to ours, is a hybrid system: "We can combine both cryptosystems' administrative benefits, by restricting public-key deployment to servers, and by using symmetric-key protocols for desktop clients". This paper is highly worth reading, and provides further evidence for many of our arguments.

[3] Here as elsewhere we assume that principals possess global names and form their trust relationships in terms of these. This approach has been criticized: often the name of some party to a communication is less relevant than some attribute of this party (such as his organizational affiliation, security clearance, credit rating, etc.)[38]. If trust relationships are expressed and calculated in terms of names then some other mechanism (such as an access control list) must be used to map identities to these authorizations or attributes, which is both clumsy and a threat to privacy since user identities are exposed in situations where they are not strictly necessary. We agree with this criticism, but we believe the debate is orthogonal to our approach: DSs could wield attribute or authorization certificates just as easily as identity certificates. For simplicity of presentation we will continue to speak in terms of names but no loss of generality should be assumed.

References

[1] R. Owens, *One-Time Passwords: Functionality and Analysis*, October 2000
http://rr.sans.org/authentic/onetime2.php

[2] N. Haller, C. Mertz, P. Nesser, and M. Straw, *RFC 2289: A One-Time Password System*, February 1998
http://www.ietf.org/rfc/rfc2289.txt

[3] A.K. Jain, R. Bolle, and S. Pankanti, *Biometrics: Personal Identification in Networked Society*, Kluwer, 1991
http://www.wkap.nl/prod/b/0-7923-8345-1

[4] A.O. Freier, P. Karlton, and P.C. Kocher, *The SSL Protocol Version 3.0*, November 1996
http://www.netscape.com/eng/ssl3/draft302.txt

[5] T. Wu, *The Secure Remote Password Protocol*, Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium, March 1998
http://www-cs-students.stanford.edu/~tjw/srp/ndss.html

[6] T. Wu, *RFC 2495: The SRP Authentication and Key Exchange System*, September 2000
http://www.ietf.org/rfc/rfc2945.txt

[7] Microsoft .NET Passport
http://www.microsoft.com/myservices/passport/security.doc

[8] The Liberty Alliance Project
http://www.projectliberty.org/

[9] P.H. Baker, E. Maler, et. al, *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*
http://www.oasis-open.org/committees/security/docs/

[10] W. Diffie and M.E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information The-

ory, 22, 1976
http://citeseer.nj.nec.com/diffie76new.html

[11] L.M. Kohnfelder, "Toward a Practical Public-Key Cryptosystem", B.Sc. thesis, MIT Department of Electrical Engineering, 1978

[12] C. Daniel, *Internet Security cannot be left to technologists alone*, Financial Times, September 2001
http://specials.ft.com/ftit/FT34WRFC6RC.html

[13] J. Lewis, *PKI Won't Hit The Mainstream Until Vendors Reduce Complexity*, InternetWeek, January 2001
http://www.internetweek.com/columns01/lewis010801.htm

[14] B.D. Reimers, *PKI's Are Still Tough To Deploy*, InternetWeek, April 2001
http://www.internetweek.com/security/secure040901-1.htm

[15] GAO, *Information Security: Advances and Remaining Challenges to Adoption of Public Key Infrastructure Technology*, Item No. 0546-D; SuDocs No. GA 1.13:GAO-01-277, February 2001
http://www.gao.gov/new.items/d01277.pdf

[16] A. Arsenault and S. Farrell, *RFC 3157: Securely Available Credentials – Requirements*, August 2001
http://www.ietf.org/rfc/rfc3157.txt

[17] S. Garfinkel, *PGP: Pretty Good Privacy*, O'Reilly & Associates, 1995
http://www.oreilly.com/catalog/pgp/

[18] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, *RFC 2440: OpenPGP Message Format*, November 1998
http://www.ietf.org/rfc/rfc2440.txt

[19] A. Whitten and J.D. Tygar, *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0*, Proceedings of 8th USENIX Security Symposium, August 1999
http://www-2.cs.cmu.edu/~alma/johnny.pdf

[20] J. Raymon, *Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems*, Workshop on Design Issues in Anonymity and Unobservability, 2000
http://citeseer.nj.nec.com/454354.html

[21] S.W. Smith and S.H. Weingart, *Building a High-Performance, Programmable Secure Coprocessor*, Computer Networks (Special Issue on Computer Network Security), 31, pp. 831-860, April 1999
http://www.research.ibm.com/secure_systems/papers/arch.pdf

[22] S.W. Smith and D. Safford, *Practical Private Information Retrieval with Secure Coprocessors*, IBM Research Report RC-21806, July 2000
http://www.research.ibm.com/secure_systems/papers/rc21806.pdf

[23] J. Daugman, *High Confidence Visual Recognition of Persons by a Test of Statistical Independence*, IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 15 no. 11, pp. 1148-1161, November 1993

[24] S.M. Bellovin and M. Merritt, *Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks*, Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1992
http://www.research.att.com/~smb/papers/neke.ps

[25] ITU-T Rec. X.509, *The Directory: Public-key and attribute certificate frameworks*, March 2000
http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.509

[26] R. Housley, W. Ford, W. Polk, and D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, January 1999
http://www.ietf.org/rfc/rfc2459.txt

[27] R. Housley and Tim Polk, *Planning for PKI*, John Wiley & Sons, Inc., 2001
http://www.wiley.com/cda/product/0,,0471397024,00.html

[28] J.G. Steiner, B.C. Neuman, and J.I. Schiller, *Kerberos: An Authentication Service for Open Network Systems*, Proceedings of the Winter 1988 Usenix Conference, pp. 191-202, February 1988
ftp://athena-dist-mit.edu/pub/kerberos/doc/usenix.ps

[29] J. Kohl and B.C. Neuman, *RFC 1510: The Kerberos Network Authentication Service (V5)*, September 1993
http://www.ietf.org/rfc/rfc1510.txt

[30] C. Rigney et. al, *RFC 2865: Remote Authentication Dial In User Service*, June 2000
http://www.ietf.org/rfc/rfc2865.txt

[31] W. Yeong, T. Howes, and S. Kille, *RFC 1777: Lightweight Directory Access Protocol*, March 1995
http://www.ietf.org/rfc/rfc1777.txt

[32] M. Wahl, T. Howes, and S. Kille, *RFC 2251: Lightweight Directory Access Protocol (v3)*, December 1997
http://www.ietf.org/rfc/rfc2251.txt

[33] RSA Security Inc., *Delivery of Ten Millionth RSA SecurID Authenticator*, Press Release, December 2001
http://www.rsasecurity.com/news/pr/011212.html

[34] P. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, Advances in Cryptology–CRYPTO '96, Springer LNCS

1109, pp. 104-113, 1996
http://www.cryptography.com/resources/whitepapers/TimingAttacks.pdf

[35] P. Kocher, *Differential Power Analysis,* Advances in Cryptology-CRYPTO '99, Springer LNCS 1666, pp. 388-397, 1999
http://www.cryptography.com/resources/whitepapers/DPA.pdf

[36] R.J. Anderson and M.G. Kuhn, *Tamper Resistance – A Cautionary Note*, Proceedings of the Second Usenix Workshop on Electronic Commerce, pp. 1-11, November 1996
http://www.cl.cam.ac.uk/~mgk25/tamper.html

[37] ITU-T Rec. X.500, *The Directory: Overview of concepts, models and services*, February 2001
http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.500

[38] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, *RFC 2693: SPKI Certificate Theory*, September 1999
http://www.ietf.org/rfc/rfc2693.txt

[39] P. Gutmann, *X.509 Style Guide*, October 2000
http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt

[40] P.C. Van Oorschot, W.S. Ford, S.W. Hillier, and J. Otway, *Method for efficient management of certificate revocation lists and update information*, U.S. Patent 5,699,431, December 1997
http://www.uspto.gov/

[41] D.A. Cooper, *A More Efficient Use of Delta-CRLs*, Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 190-202, May 2000
http://csrc.nist.gov/pki/documents/sliding_window.pdf

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

[43] D. Pinkas, *Internet Draft: Delegated Path Validation and Delegated Path Discovery Protocols*, work in progress, July 2001
http://www.ietf.org/internet-drafts/draft-ietf-pkix-dpv-dpd-00.txt

[44] R. Housley and T. Polk, *Planning for PKI*, John Wiley & Sons, Inc., chapter 11, 2001
http://www.wiley.com/cda/product/0,,0471397024,00.html

[45] B. Kaliski, *RFC 2314: PKCS #10: Certificate Request Syntax Version 1.5*, March 1998
http://www.ietf.org/rfc/rfc2314.txt

[46] B. Kaliski, *RFC 2315: PKCS #7: Cryptographic Message Syntax Version 1.5*, March 1998
http://www.ietf.org/rfc/rfc2315.txt

[47] C. Adams and S. Farrell, *RFC 2510: Internet X.509 Public Key Infrastructure Certificate Management Protocols*, March 1999
http://www.ietf.org/rfc/rfc2510.txt

[48] M. Myers, X. Liu, J. Schaad, and J. Weinstein, *RFC 2797: Certificate Management Messages over CMS*, April 2000
http://www.ietf.org/rfc/rfc2797.txt

[49] X. Liu, C. Madson, D. McGrew, and A. Nourse, *Cisco System's Simple Certificate Enrollment Profile*, 2000
http://www.cisco.com/warp/public/cc/pd/sqsw/tech/scep_wp.htm

[50] S. Berkovits, S. Chokhani, J.A. Furlong, J.A. Geiter, and J.C. Guild, *Public Key Infrastructure study: Final Report*, MITRE Corporation, April 1994
http://csrc.nist.gov/pki/documents/mitre.ps

[51] D.C. Feldmeier and P.R. Karn, *Unix Password security – ten years later*, CRYPTO Proceedings, 1989
http://www.ja.net/CERT/JANET-CERT/../Feldmeier_and_Karn/crypto_89.ps

[52] T. Wu, *A Real-World Analysis of Kerberos Password Security*, Proceedings of the 1999 Network and Distributed System Security Symposium, 1999
http://www.isoc.org/isoc/conferences/ndss/99/proceedings/papers/wu.pdf

[53] A.K. Lenstra and E.R. Verheul, *Selecting Cryptographic Key Sizes*, to appear in The Journal of Cryptology, Springer-Verlag
http://www.cryptosavvy.com/Joc.pdf

[54] C.G. Günther, *An identity-based key-exchange protocol*, Advances in Cryptology-EUROCRYPT '89, Springer LNCS 434, pp. 29-37, 1990

[55] R. Shirey, *RFC 2828: Internet Security Glossary*, May 2000
http://www.ietf.org/rfc/rfc2828.txt

[56] Cryptographic Appliances, *Cryptographic Appliances Releases Two PCI Accelerators*, Press Release, August 2001
http://www.cryptoapps.com/press08072001.html

[57] V. Miller, *Uses of elliptic curves in cryptography*, Advances in Cryptology: proceedings of Crypto '85, LNCS 218, pp. 417-426, 1986

[58] N. Koblitz, *Elliptic curve cryptosystems*, Mathe-

matics of Computation, 48, pp. 203-209, 1981

[59] D. Chaum, *Blind Signatures for Untraceable Payments*, Advances in Cryptology: Proceedings of Crypto 82, Plenum Press, pp. 199-203, 1983

[60] D. Chaum, *Security without Identification: Transaction Systems to Make Big Brother Obsolete*, Communications of the ACM, v. 28, n. 10, pp. 1030-1044, October 1985
http://www.chaum.com/articles/Security_Wthout_Identification.htm

[61] P. Karn and W. Simpson, *RFC 2522: Photuris: Session-Key Management Protocol*, March 1999
http://www.ietf.org/rfc/rfc2522.txt

[62] M. Hur, B. Tung, T. Ryutov, C. Neuman, A. Medvinsky, G. Tsudik, B. Sommerfeld, *Internet Draft: Public Key Cryptography for Cross-Realm Authentication in Kerberos*, work in progress, November 2001
http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-cross-08.txt

[63] S. Tuecke, D. Engert, I. Foster, V. Welch, M. Thompson, L. Pearlman, and C. Kesselman, *Internet Draft: Internet X.509 Public Key Infrastructure Proxy Certificate Profile*, work in progress, February 2002
http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-02.txt

[64] J. Novotny, S. Tuecke, and V. Welch, *An Online Credentials Repository for the Grid: MyProxy*, Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 20001
http://www.globus.org/research/papers/myproxy.pdf

[65] Y. Hsu and S.P. Seymour, *An Intranet Security Framework Based on Short-Lived Certificates*, Proceedings of the 6th workshop on Enabling Technologies Infrastructure for Collaborative Enterprises, 1997
http://www.computer.org/internet/ic1998/w2073abs.htm

[66] Secure Computing Corporation, *Virtual Smart Card Server Solution*, July 2000
http://www.securecomputing.com/pdf/safeword_plus_wp_vscs.pdf

[67] W. Ford, P.H. Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein, and J. Lapp, *XML Key Management Specification (XKMS)*, March 2001
http://www.w3.org/TR/xkms/

[68] D. Eastlake, J. Reagle, and D. Solo, *RFC 3275: (Extensible Markup Language) XML-Signature Syntax and Processing*, March 2002
http://www.ietf.org/rfc/rfc3275.txt

[69] D. Eastlake, J. Reagle, T. Imamura, B. Dillaway, and E. Simon, *XML Encryption Syntax and Processing, W3C Candidate Recommendation*, March 2002
http://www.w3.org/TR/xmlenc-core/

[70] S. Godik and T. Moses, *OASIS eXtensible Access Control Markup Language, Committee Draft*, April 2002
http://www.oasis-open.org/committees/xacml/docs/

[71] B. Ramsdell, *RFC 2633: S/MIME Version 3 Message Specification*, June 1999
http://www.ietf.org/rfc/rfc2633.txt

[72] R. Housley, *RFC 2630: Cryptographic Message Syntax*, June 1999
http://www.ietf.org/rfc/rfc2630.txt

[73] T. Dierks and C. Allen, *RFC 2246:The TLS Protocol Version 1.0*, January 1999
http://www.ietf.org/rfc/rfc2246.txt

[74] D. Taylor, *Internet Draft: Using SRP for TLS Authentication*, work in progress, June 2001
http://www.ietf.org/internet-drafts/draft-ietf-tls-srp-01.txt

[75] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen, *Internet Draft: SSH Protocol Architecture*, work in progress, January 2002
http://www.ietf.org/internet-drafts/draft-ietf-secsh-architecture-12.txt

[76] R. Thayer, N. Doraswamy, and R. Glenn, *RFC 2411: IP Security Document Roadmap*, November 1998
http://www.ietf.org/rfc/rfc2411.txt

[77] B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, chapters 2-6, 1996
http://www.counterpane.com/applied.html

[78] D. Davis, *Compliance Defects in Public-Key Cryptography*, Proceedings of the 6th USENIX UNIX Security Symposium, July 1996
http://world.std.com/~dtd/compliance/compliance.ps

Security Characteristics of Cryptographic Mobility Solutions

Sarbari Gupta
Electrosoft Services, Inc.
sarbari@electrosoft-inc.com

Abstract

This paper focuses upon the security characteristics of cryptographic mobility (CM) solutions. CM solutions allow the roaming user to make use of their cryptographic credentials from any workstation or system that has network connectivity to the appropriate credential server(s), without the need to carry portable software or hardware tokens. While CM implementations have a greater potential for security vulnerabilities than traditional (non-mobile) cryptographic implementations, it is anticipated that the demand for products in this technology category will continue to grow in the future.

1 OVERVIEW

Traditionally, systems that use public-private key pairs for user authentication, digital signature or message confidentiality protection store the user's private keys and private user data in encrypted form on the client system's hard drive. However, this mechanism does not allow the user to `roam`, that is, to access the private key information from any generic client terminal, in order to digitally sign or encrypt material from that terminal.

Within a public key infrastructure (PKI), a user *credential* is a cryptographically protected object, that may contain the owner's private key(s), public key certificate(s), certificates for CAs within the owner's PKI hierarchy, trust roots relevant to the user, and other domain-specific parameters such as user IDs, cryptographic algorithm names, salt values, etc. PKI credentials may reside in hardware or software tokens.

Cryptographic roaming is highly advantageous for many business and consumer applications. Such solutions make cryptography accessible from a wide variety of client systems, including public kiosks and terminals. Currently, there are two fundamental mechanisms for providing roaming access to public key credentials – these are described below.

- *Portable credentials* – the user carries their cryptographic credentials in a portable format (which may be hardware or software). Smart cards and other types of hardware tokens, and software credentials stored on portable media such as floppies are examples of portable credentials. While the portable hardware token approach is sound from the security perspective, it often requires special hardware at the client workstation, and is often cumbersome, impractical or cost-prohibitive for most roaming scenarios. Conversely, software portable tokens are cheap and easy to deploy. However, very often, software portable tokens are protected using password-based encryption techniques (such as PKCS#5 [P1]). Within a roaming environment, where public terminals and kiosks may be used for secure transactions, such a software portable token may easily be subjected to offline brute-force password guessing attacks, against a reasonably small password space.

- *Credential Servers* – this approach makes use of online Credentials Server(s), which allows the credential owner to make use of their private key material after they have successfully authenticated themselves to one or more online Authentication Servers. In this approach, all or portions of the user's private key material and private data are stored, in a protected form, on a system that is accessible to the online authentication and credential servers.

In this paper, we will focus on the latter approach for mobility, and analyze the generic security characteristics. We will refer to such solutions as ***Cryptographic Mobility (CM)*** solutions. There are several CM products and techniques that are currently available. References to some of the major schemes are listed at the end of this paper. This paper is organized as follows. Section 2 defines a generic architecture for CM systems, while Section 3 describes the generic operational phases. Section 4 discusses some of the attributes that characterize a CM solution, while Section 5 describes the security issues that are more likely to arise in CM implementations. Section 6 analyzes the applicability of a CM solution, Section 7 provides brief,

high-level descriptions of some of the currently available CM products, and Section 8 presents some conclusions.

2 ARCHITECTURE OF A CRYPTOGRAPHIC MOBILITY SOLUTION

A generic CM system comprises several functional components as illustrated in Figure 1. Each of the components is further described below. It may be noted that although the functional components are shown as distinct boxes the figure, two or more of the components may be instantiated on the same physical system for a given CM implementation.

Client Station: This component represents various shared workstations and/or public kiosks that may be utilized by a CM user to interact with the CM system. The Client Station can be used to initialize a roaming credential through interactions with the Initialization Server. The Client Station can also be used to activate and use the roaming credential for secure data exchange. The CM user is required to authenticate to one or more Authentication Servers, following which, the user's credentials are made available with the cooperation of the Credential Server(s). The Client Station may run some kind of GUI-based client software that is provided as a part of the specific mobility solution.

Initialization Server: The Initialization Server is responsible for the creation of roaming credentials, and the establishment of the authentication information for the roaming user. If new certificates need to be issued to the user during the creation of the roaming credentials, this component may interact with a Certification Authority. The Initialization Server will also typically interact with the Authentication and Credential Servers to populate their databases with the appropriate data for the user.

Authentication Server(s): This is the component that is responsible for authenticating the roaming user before they are allowed access to their credentials. The Authentication Server may need to maintain an authentication database that allows it to determine whether a given user's authentication attempt was successful.

Credential Server(s): This component may hold all or portions of the user's cryptographic credentials. The held credentials may be stored in a backend data store.

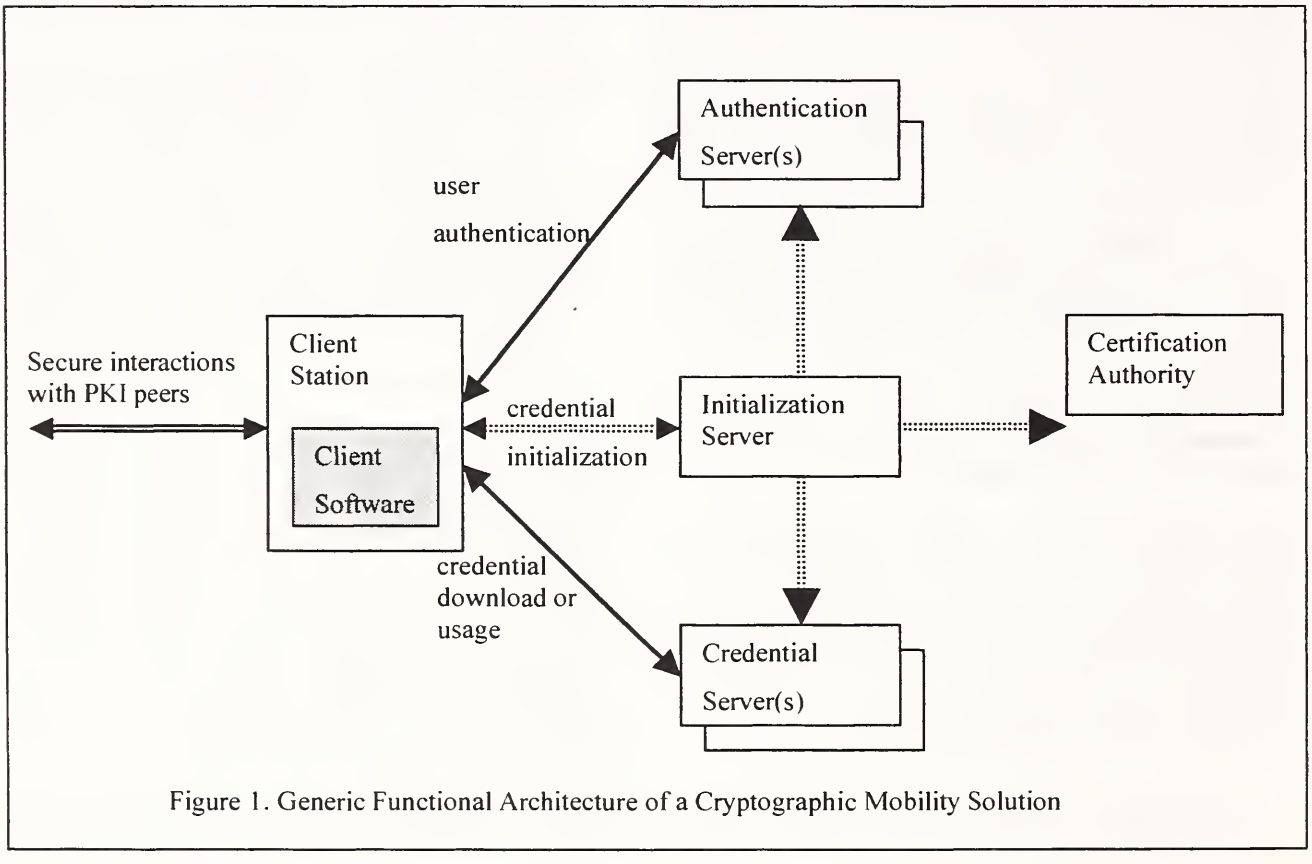


Figure 1. Generic Functional Architecture of a Cryptographic Mobility Solution

Certification Authority (CA): The CA component is responsible for the generation of signed certificates for roaming credential use, if that is necessary to the CM solution. The Initialization Server interacts with the CA component to create and initialize special roaming credentials.

3 OPERATIONAL PHASES OF A CRYPTOGRAPHIC MOBILITY SOLUTION

Although there are varied schemes for implementing a CM solution, the majority of the schemes can be broken down into some generic operational phases. These are described below.

Credential Initialization Phase - During this phase, the user's PKI credentials may be created and/or packaged to accommodate roaming usage. The Client Station component interacts with the CM Initialization Server to perform the steps needed to establish a set of roaming-capable credentials, and make them accessible to the user. Some CM solutions can package existing user credentials into a roaming accessible form, while others require the generation of specially-crafted credentials to support roaming usage. In the latter type, the CM solution will typically require generation of new key pairs, and the issuance of new certificates. In some cases, the CM solution may provide its own Certification Authority or certificate generation function. In other cases, the CM solution can pass the certificate signing request to an external Certification Authority.

During the Initialization Phase, information is collected for the authentication of the roaming user. This information is used to populate the Authentication Server database. Additionally, the roaming credential package is handed to the Credential Server for access over the Internet.

Authentication Phase - This phase of the roaming protocol occurs each time the user wishes to establish their connection to their online credential server to download or make use of their roaming credentials. During this phase, the online user operating from a generic workstation or terminal authenticates to the Authentication Server to assert and establish their identity. Although there are wide variations in the technologies and protocols used between the Client Station and the Authentication Server, the human user at the Client session almost uniformly is required to provide an account identity and a password. The user may also be required to provide answers to one or more questions to provide restricted, personal, information.

The user provided account name, password, and perhaps other user secrets that are required, may be used in a variety of ways to authenticate to the remote Authentication Server(s). Some schemes use a challenge-response protocol using the user-provided password, others use the password for a local operation (at the Client Station) to unlock private key functions to authenticate to the Authentication Server(s). Yet other schemes use some form of strong password schemes using strong secrets that are provided by remote, online server(s).

The frequency of the authentication phase during a user session, and the ability to invoke multiple uses of the user's private roaming credentials, varies considerably with the particular scheme under consideration. For ease of use, some schemes allow caching of the user provided authentication information so that the user is only required to provide this information once during the course of a session regardless of the number of times the roaming credential is used and the variety of applications that invoke the user private key. While very convenient for the user, this approach of single sign-on is fraught with risk in terms of credential hijack by subsequent users of the Client Station. Other CM schemes take the conservative approach of necessitating a user authentication to the Authentication Server(s) each time the user private key is used. This approach provides much greater security for the roaming credential and supports other associated security functions, such as auditing the use of the private keys for purposes of fraud detection and non-repudiation; however, the user's convenience factor is greatly reduced.

Credential Download Phase - Many of the CM technologies involve an intermediate phase of credential download to the Client Station. The credential download follows the authentication phase and precedes the credential usage phase. The downloaded material may be part or all of the user's credentials and other private user data, and is always protected with additional layer(s) of cryptography to prevent credential misuse at or to the Client Station. The additional layer(s) of cryptographic protection may be unlocked using a session authentication key or through user provided secrets.

Credential Usage Phase - The most significant operational phase of a CM solution is the actual use of the user's private key for authentication, digital signature, or message decryption functions. During this phase, the user has the ability to make use of their credentials, either by unlocking a local copy of their

credentials, or by availing of online services that assist in completing the usage of the credentials.

Some CM schemes allow the download of a copy of the user's credentials to the local client station. This copy is cryptographically protected and may be unlocked for use after the user provides a PIN or passphrase. The user is able to make repeated use of the local copy of the credential without involving the remote Authentication or Credential Servers. This type of scheme is typically faster and easier to use. However, the local copy of the user's credentials on the Client Station may be subjected to offline attacks and unauthorized reuse.

In certain CM schemes, the user's credentials or key material is never actually downloaded to the local Client Station. Each use of the roaming credential requires the involvement of one or both of the Authentication and Credential Servers. This type of scheme, though possibly slower and more tedious to use, has the benefit that it never exposes the user's credentials at the Client Station or allows its copy or reuse by an attacker.

Credential Release Phase - The final phase of a CM scheme is the Credential Release Phase, during which the Client Station scrubs any downloaded key material and authentication information from memory and magnetic storage, and formally ends the current user session. This phase may be implemented unilaterally within the Client Station, or it may require the Client Station to interact with the Credential and/or Authentication Servers to inform them about the termination of the roaming user session.

4 CHARACTERISTICS OF CM SOLUTIONS

There are some common characteristics in nearly all of the cryptographic mobility solutions that are available. These are enumerated below.

- Client Station does not need special hardware such as token readers – The fundamental reason to seek out a CM solution is to avoid the use of special hardware tokens and token readers. Thus all CM implementations share this common attribute.

- Client Station needs vendor-specific CM client software that has to be either downloaded or installed in a trusted manner – In all of the CM products studied, there is a need for a vendor-specific client software module that performs the needed operations (such as authenticating the user, downloading and storing a local copy of the credentials or key materials, and enabling the use of the user's PKI credentials for private key operations. Since this piece of software collects the user's authentication information as well as handles the user's private key usage, the assurance level for the software has to be fairly high.

- User needs to remember authentication information, whether it is a password, or answers to a series of personal questions – The roaming user has to authenticate to an online server to acquire the ability to use public key credentials for subsequent authentication operations. However, the user cannot use public key operations during the initial authentication phase for obvious reasons. The user also cannot typically use other strong mechanisms such as hardware One-Time-Password generators (e.g. SecurID cards) since that would involve the usage of hardware tokens. Thus, most CM implementations make use of secret sharing schemes, such as passwords or answers to personal questions for the initial authentication phase.

- Users interact with remote Authentication Servers to authenticate themselves to the system – Since the user is assumed to be working from a Client Stations that does not have a local copy of their credentials, all roaming solutions necessarily involve a remote authentication function where the authentication information supplied by the human user is transported through some means to a remote server which verifies them to identify and authenticate the user.

- An authenticated user is able to perform cryptographic operations using their private key – The fundamental goal of a roaming PKI user is to ultimately use their private key for digital signature or data decryption operations. All CM implementations provide this facility through different mechanisms.

- Credential is unusable after the end of the user's session – The premise of a roaming user is that they avail of a shared Client Station when attempting to use their PKI credentials. Thus, it is very important that upon the last user leaving the Client Station, there be no residual ability to make use of the last user's credentials by the subsequent user. All CM implementations use this as a common functional goal.

- Part or all of a user's PKI credentials are stored on an online remote credential server -The user's

authentication information is stored on a database accessible to an online authentication server.

5 SECURITY ISSUES WITH CM SOLUTIONS

CHARACTERISTICS THAT ADD SECURITY VULNERABILITIES

Due to the fundamental nature of a cryptographic mobility solution, in that it makes use of remote authentication and credential servers, there are a number of additional security issues that may arise. Depending upon the particular implementation of CM, some or many of these issues may be sidestepped through the use of novel schemes. This section will describe some of the security issues that are particularly relevant when assessing a CM implementation. The various architectural components of a CM system have their own characteristics that may introduce additional security vulnerabilities. Some of these characteristics are described below.

- The Client Station is assumed to be a shared access workstation or kiosk that has network connectivity to the CM Server entities, possible over the Internet. The Client Station is also assumed to use some form of CM client software that has to be installed.

- The Authentication Server(s) are assumed to be available online, possibly over the Internet, for access by Client Stations. The Authentication Server is also expected to have some form of database (possibly on a backend system,) that holds user data that can be used to complete the authentication step.

- The Credential Server(s) are also assumed to be online and available for network-based attacks. The Credential Server has to ascertain that the user has been properly authenticated before allowing the download or use of their private keys. The private key material for CM users is typically held in some kind of database at the backend of the Credential Server.

- The three primary architectural components of a CM system interact with each other using online protocols over shared and (often) untrusted networks. Thus, these protocols may be attacked by network intruders.

POTENTIAL SECURITY VULNERABILITIES

When evaluating the security of a CM solution, a number of questions should be asked. The answers must then be taken collectively to determine the specific security vulnerabilities that exist for a given system. The security relevant questions to be asked include:

- How and where are client key pairs generated? Depending upon whether the user's key pair is generated at the Client Station or on a server, the non-repudiation claims of a private key may be stronger or weaker. To support a strong case for non-repudiation, the server system must never handle the unencrypted private keys or private key material for a user.
- Where is the user's private key actually used – at the Client Station or on a remote server component? The location of "usage" of the private key has an impact on the non-repudiation properties of the CM implementation.
- How is the client private key deposited at the Credential Server? The Credential Server must hold all or part of the user's private key in order to allow the user to have roaming access to the private key. However, the mechanism for protecting the private key while the credential server holds it is very significant in determining whether a capture of the protected private key container, leads to the ability to use that private key.
- How is the client private key protected at the Credential Server?
- What are the security characteristics of the authentication protocol between the Client Station and the Authentication Server(s)? Are the protocols susceptible to man-in-the-middle and eavesdropping attacks? Does the scheme reveal the CM user's authentication information to the Authentication Server?
- How is the client private key made available for use at the Client Station?
- Can the user's private key be compromised at the Client Station?
- How is the client private key disabled at the end of the user's session?
- How does the user establish trust in the Client Software? How does the user know the source of the Client Software and establish trust in its integrity?
- How does the Client Module handle the sensitive authentication information that is collected from the user – is it held in memory or is it cleared after each use?

- How does the Client Module handle the local copy of the user's credentials that is obtained from the Credential Server – is it held in memory for ease of use or is it cleared after each use?
- How does the Client Module establish trust in the Authentication and Credential Servers? If SSL is used, how are the PKI trust roots established in the Client Station?
- Can the Authentication Server(s) be compromised such that the authentication database becomes available to the attacker? If so, what can the attacker do with the captured information?
- Can the Credential Server be compromised such that the Credential Database becomes accessible to the attacker? If so, what can the attacker achieve with the captured information?
- Can the CM user be subjected to Denial-of-Service attacks through the compromise or disablement of the Authentication and Credential Servers?

6 APPLICABILITY OF CM SOLUTIONS

In this section, the major issues that affect the decision to deploy a CM solution are briefly explored. While CM solutions may be recommended in certain usage scenarios, they are definitely not advisable in others. This section attempts to clarify some of the issues that should be considered before adopting a CM product.

REQUIREMENTS THAT DRIVE THE SELECTION OF A CM SOLUTION

The decision to deploy a cryptographic mobility solution is usually made because of some requirements that are levied due to the characteristics of the user, the user's IT environment, or the secure application. Some of the typical requirements that drive an organization to consider a CM implementation are:

- Users are highly mobile, and need to use variety of systems/workstations, operated and controlled (possibly) by various organizations
- Hardware cryptographic tokens too expensive or cumbersome or infeasible due to requirement to have compatible readers
- Users are in an IT environment where dedicated workstations are infeasible or prohibitively expensive
- Software cryptographic tokens not practical or secure enough
- Simple user interface is required – user only needs to provide user ID and password, and answer simple personal questions
- User or application requires strong authentication, and/or message encryption

CONTRAINDICATIONS FOR SELECTION OF CM SOLUTIONS

Some environments and user populations exhibit requirements that are contraindications for certain types of CM products. When these requirements exist within an environment, extra caution must be exercised in selecting a CM product that meets these requirements. These include:

- Strong, legally binding non-repudiation of electronic transactions is an absolute must
- Recovery of encryption keys is essential
- Long term archival and possible usage of the protected data
- Guaranteed access to credentials for decryption and signatures – zero tolerance for denial-of-service situations

7 A SAMPLING OF CM TECHNOLOGIES AND PRODUCTS

In this section, several of the leading products and technologies that provide CM solutions are identified and described very briefly. It should be mentioned that the information contained in this section is based upon data collected from the vendor websites and dialogue with vendor personnel. The goal was to develop a brief, high-level description of each product, rather than to provide detailed technical coverage of each product. These descriptions should not be used to evaluate the products – the interested reader is directed to contact the vendor directly to obtain more technically accurate and up to date information on each product.

ENTRUST ROAMING PKI

Entrust has been providing a PKI mobility solution within Entrust/Roaming™, a complementary product to Entrust/PKI® 5.0 [E1]. Entrust/Roaming™ makes use of a public Directory Server to store the cryptographic profiles for users, encrypted with a strong symmetric key. A strong password authentication mechanism named SPEKE is used to securely download the strong keys that can decrypt the user's protected cryptographic

profiles, and hence make use of the private key material held inside.

SPEKE stands for Simple Password-authenticated Exponential Key Exchange [E2]. It provides *strong password authentication* to prove knowledge of a small secret (namely, a password) without revealing it to anyone.

An Entrust profile contains the PKI credentials for a given user. Typically, the profile is stored locally on the hard drive in a form that is protected with a user-chosen password. This protection format provides very little resistance against a concerted offline dictionary-based attack. Hence, in the Entrust Roaming solution, the standard user profile is further encrypted with a strong symmetric key K ($K \geq 128$ bits) and stored on a Directory Server. The Entrust solution also makes use of an online Roaming Server that authenticates the user using the SPEKE protocol, establishes a shared strong key S based upon the authentication, and provides the user with $E_S(K)$ such that the user is then able to retrieve K and hence unlock and use their cryptographic profile. The downloaded roaming profile can then be used similar to a local Entrust profile stored on the local hard drive.

VERISIGN ROAMING

The VeriSign PKI roaming solution is a part of the VeriSign OnSite PKI offering [V1, V2, V3, V4]. It uses multiple, independent Roaming servers, each of which provides a component of the key that the user employs to retrieve and decrypt his or her `roaming credentials` from the Storage Server. The technique for utilizing multiple Roaming Servers, to recreate the strong key that can be used to decrypt the protected roaming credentials, is based upon the password-hardening protocol published by Warwick Ford and Burt Kaliski. In the Ford-Kaliski scheme, a user interacts with two or more Roaming Servers to harden the user's password into a strong secret, without revealing the user's password or the derived strong secret to any of the Roaming Servers. The user's roaming credentials are held on an online Storage Server in a strongly encrypted form. The user may download the protected credentials from the Storage Server, and unlock them using the strong secret that is derived with the assistance of the Roaming Servers.

ARCOT ID MOBILITY

Arcot has a patented cryptographic camouflaging scheme that it uses as the cornerstone of its ArcotID mobility solution [A1, A2, A3, A4]. In this solution, multiple PKI credentials for a user may be bundled into a protective package called a "key bag", encrypted with a strong symmetric key. Each user also possesses an ArcotID, which comprises the Arcot certificate, and the camouflaged Arcot private key. The user may download his or her "key bag" and ArcotID from an online Card Server, after authenticating to it using shared secrets. The user then supplies a PIN to the ArcotID allowing the de-camouflaging and use of the Arcot private key for authenticating to an Arcot Authentication Server (AS). [It may be noted that the unique feature of the cryptographically camouflaged Arcot private key is that it can only be unlocked with the correct PIN – however, many incorrect PINs will also yield a plausible private key to attackers, who now have to use the candidate key to authenticate to the AS. The AS is configured to lock out a user after a certain number of failed attempts.] Upon successful authentication to the Authentication Server, the user is able to retrieve a portion of the symmetric key that protects the user's "key bag". The user's supplied PIN is used to generate the other portion to recreate the key that may be used to decrypt the "key bag" to allow access to the contained credentials for normal PKI based operations.

SINGLESIGNON.NET APPLIANCE

SingleSignon.Net's *Secure Identity Appliance™* is at the heart of its Practical PKI offering [S1, S2, S3]. The Secure Identity Appliance is a hardened "black box" that can be connected to a corporate network, and store sensitive information for users. In this scheme, each user's private key is split into two components, one of which is held by the appliance, and the other is derived from the user's password. When a roaming user needs to make use of their PKI credentials for secure transactions, they authenticate to the appliance using a password-based strong mechanism to establish a secure channel. A digest of the data to be signed is then transported to the appliance over the secure, authenticated channel, and the appliance generates a partial signature using the component of the user's private key that is held by the appliance. The user then performs another partial signature operation on the returned data using the other component of the private key (that is derived from the user's password) to complete the signature on the target data. The final signature may be validated using the user's public key using the normal mechanisms. Since the appliance has to participate in every invocation of the user's private key, it can perform other operations as well, such as revocation checking, usage analysis, auditing, etc.

MICROSOFT ROAMING PROFILES

In recent versions of its operating systems, Microsoft provides a roaming profile scheme that allows the profile to be a container of PKI credentials for a user [M1, M2]. A properly authenticated domain user is able to download their profile from a central server to the local workstation. The user profile is protected using the MS Data Protection API (DPAPI). Under the MS DPAPI scheme, a master key is created for each user at first logon. Two copies of the master key are stored in the user's profile. The first is copy is protected using a 160-bit RC4 key that is derived from the user's logon password. The second copy is protected using a derivative of the Domain Controller's master key. In order to use the encrypted profile that is downloaded to the local workstation, the user's password is used to unlock the user's master key. The master key is used to retrieve the key that protects the private keys in the user's key store.

RSA SECURITY KEON WEBPASSORT

RSA Security's Web Passport offering is primarily for organizations that require the use of PKI credentials with Web Applications that provide security services such as digital signatures, VPN access or secure email [R1, R2, R3]. The product has two main components, the Web Passport Server and the Web Passport Plug-in. The former resides on a web server and is used to enforce authentication and authorization policies that determine the authorizations that users have to web resources.

Users can authenticate to the Web Passport Server using a variety of mechanisms, including passwords and SecurID authenticators. Once authenticated, the user's virtual (smart) card is downloaded from a LDAP directory to the Web Passport Plug-in on the Client Station. The virtual card is a protected container for the user's PKI certificates and private keys. Once downloaded, the virtual card can be accessed through the Microsoft Cryptographic API or the PKCS#11 interface from any application that has the capability to invoke these APIs.

The Web Passport Client Plug-in may be installed on the Client Station manually. If it is not present when the user tries to access a Web Passport protected resource, the plug-in is automatically downloaded from the web. The Web Passport virtual card contains up to two user certificates as well as the corresponding private key(s). The private key(s) are encrypted with 112-bit 3DES2EDE-CBC secret key, while the secret key is protected using a PIN Unlock Key (PUK). The PUK is a random 128-bit RC4 key. Web Passport uses cookies to keep track of authentication state, PKI credential state, key contained names, etc.

The Web Passport product supports PKI credentials from any of the industry leading CAs. It allows users to have multiple virtual cards (possibly issued by different CAs and different organizations) and allows the user to have simultaneous access to multiple sets of virtual cards.

BALTIMORE UNICERT OPTION FOR ROAMING

The UniCERT PKI product offers an optional component for roaming credential usage [B1, B2, B3]. It allows subscribers to digitally sign transactions and participate in secure online applications from a web browser without requiring the use of hardware tokens. The Baltimore CM product comprises a number of components. The Roaming Server coordinates the operation of the UNICERT roaming facility. The Roaming Administrator component allows system administrators to initialize and manage the UniCERT Roaming system by creating and updating roaming users. The Protected Encryption Key (PEK) Server deals with roaming user authentication before allowing them access to their signing key, and comprises hardware cryptographic modules. In order to insulate the Roaming and PEK Servers from direct network-based attacks from the Internet, a Proxy Server is used. There are two kinds of applets that are used within the UniCERT roaming system: a Signing Applet that can download and make use of roaming credentials to sign web data, and a Change Passphrase Applet which allows the passphrase protecting the user's signing key to be changed.

The use of two dedicated servers (Roaming and PEK) implies that both servers need to be successfully attacked in order to compromise the system. The PEK Server stores double encrypted end-user keys, while internal sequence numbers protect against brute force attacks. If fault-tolerance and high availability is required, or high volume is anticipated, multiple PEK and Roaming Servers may be deployed. The Baltimore roaming solution will work with certificates issued by any standards-compliant CA including Baltimore's UniCERT.

HUSH COMMUNICATIONS ROAMING SOLUTION

The Hush Key Server Network provides outsourced management and hosting of PKI credentials [H1, H2]. The Hush Key Server stores and manages the subscriber public and private keys through the use of a Private Key Database and a Public Key Database. The former holds the user private keys, protecting them with a "private alias" derived from a user-generated passphrase that the user never shares with any other entity. The User ID and passphrase are passed through a message digest repeatedly to generate over 1 million characters that comprise the "private alias" for the user. The "private alias" is used as a means of anonymizing and strengthening the storage of user private keys on remote servers. The private alias is used as an index into the Hush Key Private Key Database such that the private keys are nearly anonymous. The private alias is also used to authenticate the user within the Hush system.

The Public Key Database stores the corresponding user public keys. It is indexed by the user's email address and contains the user's public key certificate and revocation status. The Hush Encryption Engine facilitates public key exchange between two parties in a transparent fashion – when needed, a connection is automatically made between the first party and the Hush Key Server to retrieve the public key of the second party. The Hush Key Server also supports user key pair generation and registration with a CA. Hush offers a secure email solution using this roaming PKI scheme.

8 CONCLUSIONS

In studying various technologies and products that are currently available to support cryptographic mobility, it is clear that some areas of vulnerability remain as common elements to most available solutions. It is interesting to note that all of the systems referenced in this paper, offer strong mechanisms for user authentication, and use strong protocols for authentication and credential download that are not susceptible to active or passive man-in-the-middle attacks. All of the systems use Client Station modules that store password and key information in volatile memory only, depending upon operating system facilities to keep the information from being copied to disk. However, some of the common vulnerabilities are discussed below.

Most of the techniques described above, rely upon the use of downloaded software that comprise the Client Station Module. The downloaded module is a signed component, in most cases. However, when used from a shared workstation or public kiosk, it is difficult to have assurance regarding the trust roots that are configured into the web browsers and other PKI applications. It is also possible that rogue software implanted on these workstations captures the users keystrokes, (and hence their passwords and other authentication information,) and transfers them to some configured location. The rogue software may also affect the entropy of the random numbers generated on the workstation and hence adversely influence key pair generation, symmetric session key generation, etc.

Another area of vulnerability of roaming solutions is the susceptibility to denial-of-service attacks. A roaming solution implicitly requires the availability of one or more online servers. If any of these servers are made unavailable, the user will not have access to their cryptographic credentials, and may have to settle for unsecured interactions to meet their functional objectives. All roaming solutions should therefore address this problem by providing a high degree of redundancy to ensure that the roaming user is able to access their credentials.

Many of the solutions described above, store the user credentials on a single online server, in such a way, that the password-protected version of the credentials are available to an attacker that compromises that online server. It is well known that credentials protected by PKCS#5 type password-based cryptography are susceptible to offline password cracking attacks. Thus, the solutions that employ two or more servers in a way that the compromise of one server does not allow password-protected credentials to fall into the hands of an attacker are inherently more secure than solutions that employ a single server.

Additionally, online servers that hold credential or authentication information are high value targets for attackers. Hence, these systems must be implemented to use various types of available protections to lessen their vulnerability to such attacks. The use of proxy servers, firewalls, FIPS 140-1 approved hardware cryptographic devices, hardened operating systems, physical, personnel and operational security measures, should be employed to strengthen the security of these systems.

Some of the solutions studied cause the user's private keys, and/or the passwords that provide access to private keys, to be available to a roaming server system at some point during the initialization process. If these private keys are used for authentication or digital signature operations, the non-repudiation claims of the system are intrinsically weakened in such situations.

Despite these common weaknesses and potential vulnerabilities, it is our belief that cryptographic mobility solutions will continue to see greater adoption in the future. Due to the intrinsic nature of our current lifestyle, the user will necessarily be away from their home/office/workstation, but will continue to require access to high-grade cryptography as they pursue their personal and work-related goals. Thus, it is anticipated that the security issues with mobility solutions will be resolved with the help of innovative engineering skills, and CM implementations of PKI will gain rapid acceptance.

9 FURTHER INFORMATION

Further information about the analysis of cryptographic mobility solutions may be obtained by contacting the author, Sarbari Gupta at sarbari@electrosoft-inc.com.

10 REFERENCES

[A1] "Securing Digital Identities," Presentation to the Federal PKI TWG, September 2000.

[A2] "Arcot Key Authority: Solution for controlled access to Conventional Private Keys," Arcot Systems White Paper.

[A3] D. Hoover, B. Kausik, "Software smart cards via cryptographic camouflage," IEEE Symposium on Security and Privacy, 1999.

[A4] "Arcot WebFort™ Overview: Strong Authentication and Secure Signing Using Software," Arcot System White Paper.

[B1] "Roaming: Secure Electronic Transactions Without Boundaries", http://www.baltimore.com/unicert/unicert/roaming.html

[B2] UniCERT Roaming UniCERT Extended Technology" Baltimore White Paper.

[B3] "UniCERT Extended Technology – Roaming Version 1.0 Administrator's Guide," Baltimore UniCERT documentation.

[E1] "Secure Roaming with Software Tokens," Presentation to the Federal PKI TWG, September 2000.

[E2] Jablon, David, "Strong Password-Only Authenticated Key Exchange," ACM Computer Communication Review, vol. 26, no. 5, Oct. 1996.

[H1] "Hush Encryption Engine™ White Paper Version 2.0," Hush White Paper, July 2001.

[H2] "Services: Hush Key Server Network" http://www.hush.com/services/key_server_network/.

[M1] Finnegan, Sean, "Crypto, Key Protection, and Crypto, Key Protection, and Mobility in Windows Mobility in Windows," Microsoft Presentation.

[M2] Guttman, Peter, "How to recover private keys for Microsoft Internet Explorer, Internet Information Server, Outlook Express, and many others," White paper available at http://www.cs.auckland.ac.nz/~pgut001/pubs/breakms.txt.

[P1] "PKCS#5 v2.0 - Password-Based Cryptography Standard," RSA Laboratories, March 1999.

[R1] Carboni, E., "RSA Keon Mobile Credentials," Presentation to the Federal PKI TWG.

[R2] "RSA Keon Web PassPort: Technical Overview," A white paper from RSA Security.

[R3] Mark Diodati, "Frequently Asked Questions, RSA Keon Web PassPort, RSA Security Paper, May 2001.

[S1] "The SingleSignon.Net Difference," SingleSignOn.Net White Paper.

[S2] Bhatt, Harish, "Towards Practical PKI," SingleSignOn.Net White Paper.

[S3] Boyd, Colin, "Digital Multisignatures," *Cryptography and Coding*, Oxford University Press, 1989, pp 241-246.

[V1] Ford, Warwick, "Server- Assisted Generation of a Strong Secret from a Password," Presentation to the Federal PKI TWG.

[V2] Ford, W. and Kaliski, B., "Server-Assisted Generation of a Strong Secret from a Password," Proceedings of the IEEE Fifth International Workshop on Enterprise Security, 2000.

[V3] "VeriSign Personal Trust Service," VeriSign Product Literature.

[V4] "Administrator's Guide: ROAMING SERVICE," VeriSign Product documentation.

A Note On SPKI's Authorisation Syntax

Olav Bandmann*
Industrilogik L4i AB
Odengatan 87, SE–113 22 Stockholm, Sweden
olav@L4i.se

Mads Dam†
LECS/IMIT, Royal Institute of Technology (KTH)
KTH Electrum 229, SE–164 40 Kista, Sweden
mfd@kth.se

Abstract

Tuple reduction is the basic mechanism used in SPKI to make authorisation decisions. A basic problem with the SPKI authorisation syntax is that straightforward implementations of tuple reduction are quadratic in both time and space. In the paper we introduce a restricted version of the SPKI authorisation syntax, which appears to conform well with practice, and for which authorisation decisions can be made in nearly linear time.

1 Introduction

SPKI [3, 4] is a framework for authorisation intended particularly for networked applications. In SPKI, authority is bound to principals primarily identified by public keys. An SPKI authorisation certificate $<I,S,D,A,V>$ specifies the following items of information:

- I: An issuer as a public key.
- S: A subject which is identified primarily through a public key.
- D: A delegation flag, indicating whether or not the authorisation at hand is delegable.
- A: A "tag", or authorisation, determining the authority assigned to the subject by the certificate.
- V: A validity field determining optional intervals and online conditions for validity.

Authorisations are given in the form of S-expressions, following on from the work of Rivest [8]. S-expressions are essentially parenthesized list expressions in the style of LISP. To give an example, the right for subjects in the group `admin`, belonging to unit `finance`, to read the `income` attribute of all objects of type `person` might be given as a nested list structure

```
X :  (obj person
      (conds (grp admin)
             (unit finance))
      (op income read))
```

Authorisations and requests are given in the same syntax. If we consider X as a request a corresponding authorisation might have the shape e.g.

```
Y :  (obj person
      (conds (grp admin))
      (op income read))
```

meaning that all members of the group `admin` are permitted read access, not only members

*Work done while at SICS, Swedish Institute of Computer Science. Project at SICS supported by a grant from Microsoft Research, Cambridge, U.K.

†Partially supported by the Swedish Agency for Innovation Systems, project "Policy-Based Network Management", and by the Swedish Research Council grant 621-2001-2637, "Semantics and Proof of Programming Languages"

of the `finance` unit. Or, as another example, the authorisation might have the shape:

```
Z :  (obj person
      (conds (grp admin)
             (unit finance))
      (op income))
```

intended to be interpreted such that now the `income` attribute can be read *and* written. In both cases X should be granted, since, in an intuitive sense which we make precise in the paper, X is "more specific than", or, "authorised by", both Y and Z. The example gives the game away: An authorisation expression becomes more specific by extending lists to the right.

In order to be able to specify more complex authorisations in a concise manner, SPKI adds a number of auxillary constructions to be interpreted, essentially, as abbreviating sets of basic S-expressions. The following extensions are considered:

- `(*)` is the wildcard.
- `(* set` $X_1 \ \cdots \ X_n$`)` is the union of the sets $X_1, \ldots, X_n$, $n \geq 1$.
- `(* range` $R\ l\ u$`)` is the set of all X in the interval determined by the ordering R, lower limit l and upper limit u.
- `(* prefix` w`)` is the set of all strings having w as prefix.

Thus, to give an example, the authorisation

```
X' :  (obj person
       (conds (grp admin)
          (* set (unit finance)
                 (unit personnel)))
       (op income (* set read write)))
```

is just an abbreviation for the obvious size 4 set.

In SPKI, authorisation decisions are made through a process of "tuple reduction". Authorisations and requests are compared by computing their intersection using the operation `AIntersect`. As an example, with X and X' as defined above, `AIntersect(`X'`,`X`)` $= X$. The intersection of X' and X is the most permissive authorisation granted by both X' and X. If `AIntersect(`X'`,`X`)` $= X$ then the most permissive authorisation granted by both X' and X is X itself, or in other words, all authorisation granted by X is also granted by X', i.e. X is authorised by X'.

Computation of the `AIntersect` function is in many cases quite unproblematic, in particular when one of the arguments lack one of the special `*` forms. In general, however, `AIntersect` may cause a quadratic blowup, and this is the basic problem we address in this note.

The problem arises when comparing `* set` forms. The naive algorithm simply expands an S-expression involving `* set` forms to one without them. In many applications this procedure is in fact quite adequate. First, it will often be the case that one of the arguments to `AIntersect` is without `*` forms. Second, requests will often be small, and a quadratic blow-up will be without much consequence. The SPKI standard opens up for implementors to provide set-to-set transformations to alleviate the problems that may remain, but no concrete suggestions are given.

On the other hand one will in fact sometimes want to compute using complex authorisations. For instance, one will want to subject authorisations to simple analyses of the type:

> Q: Is authorisation X stronger than authorisation Y?

where X and Y are general S-expressions. Observe that Q is just a different way of saying that `AIntersect(`X`,`Y`)` $= X$. Secondly, simply by providing the tools to describe complex authorisations, users may eventually want to use them, for instance to precompute sets of authorisations, or to use the `* set` notation as a macro facility.

This is discussed in slightly more detail in section 9. As another example we have, in the Amanda project at SICS, been exploring a general mechanism for delegation based on a modelling of delegation as the constrained issuance of new authorisations [6, 1]. The resulting S-expression can become quite complex, and furthermore the need arises, in the decision making process, to compare authorisations of a general shape.

For these reasons we have found a need to subject the SPKI authorisation syntax to a deeper analysis. In the paper we obtain the following main results:

1. A characterisation of the SPKI entailment relation in terms of a partial ordering $\leq_S$.
2. A weak version of $\preceq_S$, which is *sound*, so that $x \preceq_S y$ implies $x \leq_S y$.
3. A restricted S-expression syntax for which the weak relation $\preceq_S$ is *complete*, i.e. coincides with $\leq_S$.
4. An efficient algorithm to compute `AIntersect`, and a proof that `AIntersect` is the greatest lower bound with respect to the $\leq_S$ ordering.

The key idea for the restricted S-expression syntax is simply to require that non-atomic elements of `* set` expressions are tagged with a unique tag (or, in SPKI terminology, type). On the evidence we have so far gathered this is nothing more than a formalisation of existing SPKI practice, and all examples in the SPKI documents [3, 4, 5] stay within the restricted syntax.

The paper is organised in the following way: In the first sections we describe authorisation trees as the basic form of *-free S-expressions, and then the syntax and semantics of S-expressions is given as sets of authorisation trees. The syntax is given in slightly abstract terms; instead of the concrete range and prefix constructions we just assume a set of primitive set constants, as this makes the presention less cluttered. In section 5 we proceed to introduce the partial orders $\leq_S$ and $\preceq_S$, and in section 6 we relate $\leq_S$ and $\preceq_S$ by showing first soundness, and then pinpointing the condition in the definition of the weak partial order which causes completeness to fail for general expressions. Then, in section 7 we turn to `AIntersect`, to establish the results (4) above.

2 Authorisation Trees

We start by defining authorisation trees, used to give semantics to the complete SPKI authorisation element. Let A be a denumerable set of "atomic" elements ranged over by a of one or several data types such as strings or integers. The set T of *authorisation trees*, ranged over by t, is determined by the following BNF style grammar:

$$t ::= a \mid (a\ t_1 \cdots t_n)$$

where $n \geq 0$.

The intention is that authorisation trees are positional. Types, in particular the type of an atom a' appearing as a subtree t_i of the tree $(a\ t_1 \cdots t_n)$, are determined by two pieces of information:

- The position i
- The label a

Types are determined by some external means; here it suffices to assume some fixed binding of types to labelled tuple positions. We define a partial order $\leq_T$ on T inductively as follows. Let $x, y \in T$.

1. If $x \in A$ or $y \in A$ then $x \leq_T y$ if and only if $x = y$.
2. If $x = (x_1 \cdots x_m) \in T$ and $y = (y_1 \ldots y_n) \in T$, then $x \leq_T y$ if and only if $m \geq n$ and $x_i \leq_T y_i$ for $i = 1, \ldots, n$.

A simple proof by induction shows that $\leq_T$ is indeed a partial order.

Elements in T represent authorisations, and the partial order $\leq_T$ represents the "is authorised by" relation, which in SPKI normally is represented in terms of the `AIntersect` operation.

Example 1 Consider the authorisation trees X, Y, and Z of section 1. We obtain that $X \leq_T Y$ and $X \leq_T Z$, but not $Y \leq_T Z$ and neither $Z \leq_T Y$. If we let

U : `(obj person`
`(conds (grp admin))`
`(op income))`

then $Y \leq_T U$ and $Z \leq_T U$.

In terms of the partial ordering $\leq_T$, the intended use of authorisation trees is as follows. Assume that a certain principal p wants to perform an action a requiring the authorisation x. Then p has the authorisation for a if (and only if) p has some authorisation y satisfying $x \leq_T y$.

A problem here is that the language is too restricted to be very useful. The solution is to use *sets* of authorisation trees instead of singletons. In the example above, p has the authorisation for a if p has some authorisation Y (a set of authorisation trees) such that there exists a $y \in Y$ satisfying $x \leq_T y$.

For this reason SPKI extends the basic S-expression syntax by notation for sets of authorisation trees.

3 Syntax of S–expressions

S–expressions represent *sets* of authorisation trees. Essentially, authorisation trees are extended with notation for set unions, in addition to primitive range and prefix constructions. To cater for these primitives we assume a denumerable set B of set constants, and a mapping $Val: B \to 2^A \setminus \{\emptyset\}$ assigning to each constant in B the nonempty set of atoms it represents.

Definition 1 (S–expressions) The set S of *S–expressions*, ranged over by X, Y, is determined as follows:

$$X ::= \texttt{(*)} \mid a \mid b \mid \texttt{(}a\ X_1 \cdots X_n\texttt{)} \mid \texttt{(* set } X_1 \cdots X_m\texttt{)}$$

where $a \in A$, $b \in B$, and $n \geq 0$, $m \geq 1$.

So, an S-expression can be either an atom (in A), a primitive set of atoms, a tuple, or a `(* set ...)` form, used to denote unions. We assume, of course, that A does not contain the special wildcard symbol `(*)`. S-expressions of the form either $a \in A$ or $b \in B$ are called *atomic*. In SPKI, two types of set constants are considered:

1. Elements representing *ranges* of elements in A. E.g. all strings in A between "bird" and "fish", alphabetically, or all integers in A greater than 5. There are many options here including type of interval and type of order. Note that, by the definition of *Val* above, we do not allow empty ranges.
2. Elements representing sets of strings in A which have a certain strings as *prefixes*. E.g. all strings in A beginning with "/pub/".

4 Semantics of S–expressions

An element X of S represents a non empty subset of T: the set of trees that are authorised by X.

Definition 2 (S–expression Semantics) We define the function $\|\cdot\| : S \to 2^T \setminus \{\emptyset\}$ as follows:

1. $\|\texttt{(*)}\| = T$
2. $\|a\| = \{a\}$ for all $a \in A$

3. $\|b\| = Val(b)$ for all $b \in B$
4. $\|(X_1 \cdots X_m)\| = \{(t_1 \cdots t_l) \mid l \geq m, \forall_{i:1\leq i\leq m} \; t_i \in \|X_i\|\}$
5. $\|(* \text{ set } X_1 \cdots X_m)\| = \|X_1\| \cup \ldots \cup \|X_m\|$

Note that, in (4), X_1 and t_1 are constrained to be atoms, by definition 1. We expect $\|X\|$ to be lower closed, so that if $t \in \|X\|$ and $t' \leq_T t$ then also $t' \in \|X\|$, or in other words, if t is authorised by X and t' is authorised by t then t' should be authorised by X as well. This property is easily verified.

Proposition 1 *For all $X \in S$, $\|X\|$ is lower closed.*

Proof A trivial induction. □

The naive way of deciding whether or not $t \in \|X\|$ is to rewrite X to a normal form where all occurrences of the `* set` construction are pushed to the outermost level, thus reducing questions of the form $t \in \|X\|$ to the case where X does not have occurrences of the `* set` construction. To make this clear, say that X_1 and X_2 are equivalent, $X_1 \cong X_2$, if $\|X_1\| = \|X_2\|$.

Proposition 2

$$(X_1 \cdots (* \text{ set } X_{i,1} \cdots X_{i,n}) \cdots X_m) \cong (* \text{ set } (X_1 \cdots X_{i,1} \cdots X_m)) \cdots (X_1 \cdots X_{i,n} \cdots X_m))$$

Example 2 Let

$$X = (a \; (* \text{ set } b \; (c \; (* \text{ set } d \; e))))$$

where all a, b, etc. are atoms in A. This represents the set of authorisation trees which are lists of length at least two beginning with a and having either b or another list t of length at least two as its second element, where t begins with c and has d or e as its second component. Using (2) along with the obvious idempotency law we obtain:

$$\begin{aligned} X &\cong (* \text{ set } (a \; (* \text{ set } b \; (c \; d))) \\ &\qquad (a \; (* \text{ set } b \; (c \; e)))) \\ &\cong (* \text{ set } (a \; b) \; (a \; (c \; d)) \\ &\qquad (a \; b) \; (a \; (c \; e))) \\ &\cong (* \text{ set } (a \; b) \; (a \; (c \; d)) \\ &\qquad (a \; (c \; e))) \end{aligned}$$

Note that, according to def. 2, the set $\|X\|$ includes not only a list such as $t = (a \; (c \; e))$, but also any authorisation tree t' for which $t' \leq_T t$. As an example, t' can have the shape $(a \; (c \; e \; f) \; g \; h)$.

5 Preorder on S–expressions

Clearly, calculations like the one in example 2 are not very efficient. To circumvent this, we need to be able to decide the following problems *without* actually calculating $\|\cdot\|$:

1. Given $t \in T$ (an authorisation request) and an S-expression X (stored, perhaps, as the authorisation element of some certificate), does $t \in \|X\|$ hold?
2. Given S-expressions X and Y, is every authorisation request granted by X also granted by Y?

Observe that both questions can be put in the same form, since t is trivially represented as an S-expression denoting the lower closure of $\{t\}$. We thus define a preorder, $\leq_S$, on S-expressions to reflect the semantics of 2. above:

Definition 3 (S–expression Preorder) The preorder $\leq_S$ on S is defined by

$$X \leq_S Y \iff \|X\| \subseteq \|Y\|$$

In other words, whatever is authorised by X is also authorised by Y. The difficulty in computing $\leq_S$ is illustrated by the following example, which also shows why $\leq_S$ is not a partial order.

Example 3 Let X = `(a (* set b c))` and Y = `(* set (a b) (a c))`. By definition 3, $X \leq_s Y$ and $Y \leq_s X$, even though $X \neq Y$ (X and Y are syntactically different). It is easy to deduce that $Y \leq_s X$ since `(a b)` $\leq_s X$ and `(a c)` $\leq_s X$ both hold. To verify $X \leq_s Y$, on the other hand, essentially requires the computation of $\|X\|$, to realize that $\|X\|$ is the lower closure of the set containing `(a b)` and `(a c)`.

This example shows the case which is to be avoided, namely where the right hand side of the equality is a set expression with at least two elements. In order to ameliorate the worst case behaviour we propose a *weaker* preorder on S, which is reasonably efficient to compute, and which does not rely on computing $\|\cdot\|$ (but it does rely on the computation of *Val*, since this function has not been explicitly defined).

The definition of the weak preorder uses the operation *flt*, which uses the equivalences such as

$$\texttt{(* set } X_1 \texttt{ (* set } X_{2,1}\ X_{2,2}\texttt{) } X_3\texttt{)} \cong \texttt{(* set } X_1\ X_{2,1}\ X_{2,2}\ X_3\texttt{)}$$

to flatten all *immediate* nestings of the `* set` constructor.

Definition 4 (Weak Preorder) Define the preorder $\preceq_s$ on S by induction in the following way. Let $X, Y \in S$. Then $X \preceq_s Y$ if and only if one of the following cases hold:

1. $Y = (*)$
2. $X, Y \in A$ and $X = Y$
3. $X = a \in A$, $Y = b \in B$, and $a \in \mathit{Val}(b)$
4. $X = b \in B$, $Y = a \in A$, and $\mathit{Val}(b) = \{a\}$ (a rather unusual situation)
5. $X, Y \in B$ and $\mathit{Val}(X) \subseteq \mathit{Val}(Y)$
6. $X = (X_1 \cdots X_m)$, $Y = (Y_1 \cdots Y_n)$, $m \geq n$, and $X_i \preceq_s Y_i$ for $i = 1, \ldots, n$
7. $X =$ `(* set` $X_1 \cdots X_m$`)` and $X_i \preceq_s Y$ for $i = 1, \ldots, m$
8. $X = b \in B, \mathit{flt}(Y) =$ `(* set` $Y_1 \cdots Y_n$`)`, and $\mathit{Val}(X) \subseteq \bigcup\{\|Y_i\| \mid 1 \leq i \leq n$ and Y_i is either atomic, or $Y_i =$ `(*)`$\}$.
9. X is of the form neither b nor `* set`, $Y =$ `(* set` $Y_1 \ldots Y_n$`)`, and $\exists_i\ X \preceq_s Y_i$

Referring to example 3 note that $Y \preceq_s X$ holds, but $X \preceq_s Y$ does not. The clause 4.9 is the cause of incompleteness. The problematic case is when X is a list and Y a `* set` expression, as in example 3. Observe also that 4.8 does in fact appeal to the function $\|\cdot\|$. However this is only a convenience, and does not introduce extra computational overhead, since all Y_i in that case are either atoms or sets of atoms. The reason for using the *flt* operation is to avoid otherwise pathological cases such as $b \preceq_s$ `(* set (* set b))`.

Since this is not completely apparent we check that $\preceq_s$ indeed defines a preorder.

Theorem 1 *The relation $\preceq_s$ is a preorder.*

Proof We must prove that

1. $X \preceq_s X$ for all $X \in S$, and
2. $X \preceq_s Y$ and $Y \preceq_s Z$ implies $X \preceq_s Z$ for all $X, Y, Z \in S$.

The first part is proved by a simple induction over the definition of $\preceq_s$. We'll skip the details.

The second part is a rather tedious induction over the structure of first Y, and then X and Z, as needed. So, assume $X \preceq_s Y$ and $Y \preceq_s Z$:

$Y =$ `(*)`: Since $Y \preceq_s Z$ the only cases that can apply are $Z =$ `(*)` (which is trivial) and $Z =$ `(* set` $Z_1 \cdots Z_m$`)` such that, in the latter case, $Y \preceq_s Z_i$ for some $i : 1 \leq i \leq m$. By the induction hypothesis, $X \preceq_s Z_i$ whence $X \preceq_s Z$ as well, completing the case.

$Y = (Y_1 \cdots Y_n)$: In this case Z has one of the forms $Z =$ `(*)`, $Z = (Z_1 \cdots Z_m)$, or $Z =$ `(* set` $Z_1 \cdots Z_m$`)`. In each case the proof is easily completed.

$Y = (* \text{ set } Y_1 \cdots Y_n)$: We may assume that $\mathit{fit}(Y) = Y$. One of the following subcases apply:

- $X = (* \text{ set } X_1 \cdots X_l)$ and $X_i \preceq_s Y$ for all $i : 1 \leq i \leq l$.
- $X = b$ and $\mathit{Val}(X) \subseteq \cup\{\|Y_i\| \mid 1 \leq i \leq n, Y_i \text{ atomic, or } Y_i = (*)\}$
- $X \preceq_s Y_i$ for some $i : 1 \leq i \leq n$

The first and third subcases are immediately dismissed by the induction hypothesis. For the second subcase we know that $Y_i \preceq_s Z$ for each $i : 1 \leq i \leq n$. We proceed then by cases on Z, noting that we need only consider the case of Y_i atomic or $Y_i = (*)$. Thus, $\mathit{fit}(Z)$ has one of the forms a, b, $(*)$, or $(* \text{ set } Z_1 \cdots Z_m)$ such that, for each choice of i we find a j such that $Y_i \preceq_s Z_j$. The former three cases are resolved by a little calculation. For the latter we may assume that Z_j is either atomic, or $Z_j = (*)$. Thus, since $\preceq_s$ is sound for atomic expressions, we know that $\|Y_i\| \subseteq \|Z_j\|$. This suffices to establish the conclusion.

The remaining cases for Y atomic are quite simple and left to the reader. □

6 Soundness and Completeness

In this section we relate the definitions of $\leq_s$ and $\preceq_s$. First we show soundness.

Theorem 2 (Soundness of $\preceq_s$) *For all* $X, Y \in S$

$$X \preceq_s Y \implies X \leq_s Y \ . \qquad (1)$$

Proof By induction over the definition of $\preceq_s$ (def. 4). We begin with the base cases 1–5. Assume that $X \preceq_s Y$ and that one of the cases 1–5 in definition 4 applies. We want to show that $\|X\| \subseteq \|Y\|$. Consider the five cases:

1. $Y = (*)$:
   $\|X\| \subseteq T = \|Y\|$
2. $X = Y = a \in A$:
   $\|X\| = \|Y\|$
3. $X = a \in A$, $Y = b \in B$, and $a \in \mathit{Val}(b)$:
   $\|X\| = \|a\| = \{a\} \subseteq \mathit{Val}(b) = \|b\| = \|Y\|$
4. $X = b \in B$, $Y = a \in A$, and $\mathit{Val}(b) = \{a\}$:
   $\|X\| = \mathit{Val}(b) = \{a\} = \|a\| = \|Y\|$
5. $X = b_1 \in B$, $Y = b_2 \in B$, and $\mathit{Val}(b_1) \subseteq \mathit{Val}(b_2)$:
   $\|X\| = \mathit{Val}(b_1) \subseteq \mathit{Val}(b_2) = \|Y\|$

Hence, cases 1 to 5 are proved. We continue with the inductive step in cases 6–9:

6. $X = (X_1 \cdots X_m)$, $Y = (Y_1 \cdots Y_n)$, $m \geq n$, and $X_i \preceq_s Y_i$ for $i = 1, \ldots, n$:
   Let $t \in \|X\|$. Then t has the shape
   $$t = (t_1 \cdots t_l)$$
   $l \geq m$, and $t_i \in \|X_i\|$ for all $i : 1 \leq i \leq m$. By the induction hypothesis, $t_j \in \|Y_j\|$ whenever $1 \leq j \leq n$ and it follows that $t \in \|Y\|$.
7. $X = (* \text{ set } X_1 \cdots X_m)$, $Y \neq (*)$, and $X_i \preceq_s Y$ for all $i : 1 \leq i \leq m$. By the induction hypothesis, $X_i \leq_s Y$ as well, so $X \leq_s Y$ follows.
8. $X = b \in B$, $\mathit{fit}(Y) = (* \text{ set } Y_1 \cdots Y_n)$, and $\mathit{Val}(X) \subseteq \bigcup\{\|Y_i\| \mid 1 \leq i \leq n$ and Y_i is atomic, or $Y_i = (*)\}$. By calculation.
9. X is of the form neither b nor $* \text{ set}$, $Y = (* \text{ set } Y_1 \ldots Y_n)$, and $\exists_i X \preceq_s Y_i$. By the induction hypothesis, $X \leq_s Y_i$ hence also $X \leq_s Y$. □

As we have pointed out, $\preceq_s$ is incomplete in general. To attain completeness the only change required is to make the final clause of 4 more inclusive.

Definition 5 *Define the preorder $\preceq'_s$ on S by replacing the clause 9 of def. 4 by the following condition:*

9′. *X is of the form neither b nor $* \text{ set}$, $Y = (* \text{ set } Y_1 \ldots Y_n)$, and $\|X\| \subseteq \|Y\|$.*

So, the source of incompleteness is clause 9, i.e. that there should exists a universal i such that every element in $\|X\|$ is bounded from above by some element from $\|Y_i\|$. The result is that this completely explains the difference between $\leq_s$ and $\preceq_s$.

Theorem 3 (Soundness and Completeness for $\preceq'_s$)
For all $X, Y \in S$

$$X \preceq'_s Y \iff X \leq_s Y \ . \qquad (2)$$

Proof The implication $\Longrightarrow$ is a simple extension of the soundness proof, taking the modified clause 9' into account. This is an easy exercise.

The completeness argument hinges on the following auxillary observation, namely that if $Y = (* \texttt{ set } Y_1 \cdots Y_n)$ and $(*) \leq_s Y$ then $(*) \leq_s Y_i$ for some $i : 1 \ldots n$. For a contradiction suppose that for all i, $(*) \leq_s Y_i$ does not hold. We may assume that Y is flattened. Each Y_i will be either atomic or have the shape $(a_i \ \ldots)$. Pick some a distinct from all the a_i. No authorisation tree of the shape $(a \ t_1 \cdots t_l)$ is in $\|Y\|$, so $(*) \leq_s Y$ cannot hold.

We now assume $X \leq_s Y$ and proceed by induction over the structure of Y. First, however, note using clause 7 of def. 4 we may assume that X is not a set expression.

1. $Y = (*)$: Since $\|(*)\| = T$ the result is immediate.
2. $Y = a$. Either $X = a$ as well, or $X = b$ and $Val(b) = \{a\}$. In either case the proof is complete.
3. $Y = b$. Either $X = a$ and $a \in Val(b)$ or else $X = b'$ and $Val(b') \subseteq Val(b)$. Either cases are immediate.
4. $Y = (Y_1 \cdots Y_n)$. The only possibility is $X = (X_1 \cdots X_m)$, $m \geq n$, and $X_i \leq_s Y_i$ for all $i : 1 \leq i \leq n$. The result then follows directly from the induction hypothesis.
5. $Y = (* \texttt{ set } Y_1 \cdots Y_n)$. By the above observation we can assume that $X \neq (*)$. If $X = a$ then $X \preceq_s Y_i$ for some i and we are done. If $X = b$ then clause 8 can be seen to hold. The final case, then, is for X of the shape $(X_1 \cdots X_m)$, and in this case the modified clause 9' applies. The proof is thus completed. □

7 Restricted S-Expressions

We then turn to the identification of a syntax fragment for which the weak preorder, even without the modification of Theorem 3, is complete. The idea is to use tagging: Every authorisation tree appearing in a set expression must contain a leading a, making it distinct from trees appearing in other elements of that set. Formally, the restricted syntax can be defined thus:

Definition 6 (Restricted S-expressions) The set R of *restricted S-expressions*, ranged over by r, along with the set of *a-restricted S-expressions*, ranged over by r^a, $a \in A$, is defined by the following grammar:

$$\begin{array}{rcl} r & ::= & (*) \mid a \mid b \mid (a \ r_1 \cdots r_n) \mid \\ & & (* \texttt{ set } r^{a_1} \ \ldots \ r^{a_m}) \\ r^a & ::= & a' \mid b \mid (a \ r_1 \cdots r_n) \end{array}$$

where $a, a' \in A$, $b \in B$, $n \geq 0$, $m \geq 1$, and where all a_i, $1 \leq i \leq m$ are distinct.

The purpose of the r^a form is to ensure that if r^a is actually a list then it is tagged by a. Choices of r^a as atoms or set constants can be done freely.

Example 4 The S-expression

$$r = (a_1 \ (* \texttt{ set } (a_2 \ c) \ (a_2 \ d) \ a_2))$$

is not restricted. The S-expression

$$s = (a_1 \ (* \texttt{ set } (a_2 \ c) \ (a_3 \ d) \ a_2)),$$

on the other hand, is restricted, as is the S-expression

$$r' = (a_1 \ (* \texttt{ set } (a_2 \ (* \texttt{ set } c \ d)) \ a_2)).$$

Note that $r \cong r'$.

In fact, the restriction appears to merely codify existing SPKI practice. All the examples of [3, 4, 5] fit the restricted syntax, and indeed it is not hard to show that that any S-expression can be rewritten into restricted form, by flattening nested `* set`'s and pushing tags out of `* set`'s, as in example 4. Thus, whenever a "real" set union (as opposed to the disjoint union provided by the restricted syntax) is needed, it suffices to use atomic S-expressions only, which is permitted.

We obtain that the weak preorder is actually complete for the restricted fragment.

Theorem 4 (Completeness, Restricted S-expressions)
For all restricted S-expressions $r_1, r_2 \in R$,

$$r_1 \leq_s r_2 \implies r_1 \preceq_s r_2$$

Proof By 3 it suffices to show $r_1 \preceq'_s r_2 \Longrightarrow r_1 \preceq_s r_2$. To establish this by induction it is sufficient to show that, for restricted expressions, condition 3.9' implies condition 4.8. We may thus assume that r_2 has the form $(* \texttt{ set } r^{a_1} \cdots r^{a_m})$, and for r_1 there are three cases to consider:

- $r_1 = (*)$. Since r_2 is restricted the only possibility is that $r^{a_i} = (*)$ as well for some i.
- $r_1 = a'$. Either $r^{a_i} = a'$ for some i, or else $r^{a_i} = b$ for some i and $b \in B$ such that $a' \in \mathit{Val}(b)$. In either case we are done.
- $r_1 = (a\ r_{1,1} \cdots r_{1,n})$, $n \geq 0$. Since all a_i are distinct, we can infer that $(a, r_{1,1}, \ldots, r_{1,n}) \leq_s r^{a_i}$ for some i : $1 \leq i \leq m$, and we are done by 4.9. □

8 SPKI's AIntersect

In this section we show that SPKI's AIntersect behaves as we expect when $\leq_s$ is interpreted as set containment, and when applied to the restricted syntax.

Since `AIntersect` is not completely defined in the SPKI documents we define this operation ourselves below. It is quite straigthforward to verify that our version fits the examples given in the draft standards.

To define `AIntersect` in the present slightly abstracted setting we need to assume that intersections exist at least on the level of set constants $b \in B$. That is, for all $b_1, b_2 \in B$ there is a b, denoted $b_1 \cap b_2$, such that $\mathit{Val}(b) = \mathit{Val}(b_1) \cap \mathit{Val}(b_2)$. We assume that $b_1 \cap b_2$ can be computed in time linear in the size of representation of b_1 and b_2.

Now, to define the `AIntersect` operation the set S is extended by the special constant $\perp$, denoting failure. For lists, if one of the argument positions is $\perp$, the entire list is $\perp$. For unions, if one of the argument positions is $\perp$ that argument is ignored. With these comments, the definition is given on fig. 1. In the figure a few symmetric cases are left out, in order not to clutter up the picture unnecessarily. Note that `AIntersect` is indeed well-defined as an operation on $S \cup \perp$. For time complexity we obtain:

Proposition 3 `AIntersect`(r_1, r_2) *is computable in time* $\mathcal{O}(n \log n)$ *where* n *is the sum of the lengths of* r_1 *and* r_2.

Proof Start by sorting the input such that elements of set expressions appear in order. This can be done in time $\mathcal{O}(n \log n)$. Once ordered, the computation of `AIntersect` is linear. □

Observe that proposition 3 applies to the restricted syntax only. Notice also that if authorisations can be assumed to be already sorted, a linear scan of the expressions suffices.

Finally we need to show that `AIntersect` is indeed the greatest lower bound with respect

$\texttt{AIntersect}((*),r) = r$

$\texttt{AIntersect}(r,(*)) = r$

$\texttt{AIntersect}(\bot,r) = \bot$

$\texttt{AIntersect}(r,\bot) = \bot$

$\texttt{AIntersect}(a,a) = a$

$\texttt{AIntersect}(a,b) = a$, if $a \in \mathit{Val}(b)$

$\texttt{AIntersect}(a,b) = \bot$, if $a \notin \mathit{Val}(b)$

$\texttt{AIntersect}(a,(a'\ r_1 \cdots r_n)) = \bot$

$\texttt{AIntersect}(a,(*\ \texttt{set}\ r_1 \cdots r_i = a \cdots r_n)) = a$

$\texttt{AIntersect}(a,(*\ \texttt{set}\ r_1 \cdots r_i = b \cdots r_n)) = a$, if $a \in \mathit{Val}(b)$

$\texttt{AIntersect}(a,(*\ \texttt{set}\ r_1 \cdots r_i \cdots r_n)) = \bot$, if none of above two cases apply

$\texttt{AIntersect}(b,b') = b \cap b'$

$\texttt{AIntersect}(b,(a,r_1 \cdots r_n)) = \bot$

$\texttt{AIntersect}(b,(*\ \texttt{set}\ r_1 \cdots r_n))$

= $(*\ \texttt{set}\ \texttt{AIntersect}(b,r'_1) \cdots \texttt{AIntersect}(b,r'_m))$,
where $r'_1,\ldots,r'_m$ is the sequence of atomic elements in $r_1,\ldots,r_n$

$\texttt{AIntersect}((a\ r_1 \cdots r_n),(a\ r'_1 \cdots r'_n\ r'_{n+1} \cdots r'_m))$

= $(\texttt{a}\ \texttt{AIntersect}(r_1,r'_1) \cdots \texttt{AIntersect}(r_n,r'_n)\ r'_{n+1} \cdots r'_m)$,
where $m \geq n$

$\texttt{AIntersect}((a\ r_1 \cdots r_n),(a'\ r'_1 \cdots r'_m)) = \bot$, if $a \neq a'$

$\texttt{AIntersect}((a\ r_1 \cdots r_n),(*\ \texttt{set}\ r'_1 \cdots r'_i \cdots r'_k))$

= $\texttt{AIntersect}((a\ r_1 \cdots r_n),r'_i)$, if r'_i has tag a

$\texttt{AIntersect}((a\ r_1 \cdots r_n),(*\ \texttt{set}\ r'_1 \cdots r'_m)) = \bot$,
if no r'_i has tag a

$\texttt{AIntersect}((*\ \texttt{set}\ r_1 \cdots r_n),\ r\ \ \texttt{as}\ (*\ \texttt{set}\ r'_1 \cdots r'_m))$

= $(*\ \texttt{set}\ \texttt{AIntersect}(r_1,r) \cdots \texttt{AIntersect}(r_n,r))$

Figure 1: Definition of AIntersect

to $\leq_S$ for the restricted syntax. This verifies that

- The operation `AIntersect` behaves as we expect of an intersection operation
- The preorder $\leq_S$ behaves as we expect with respect to `AIntersect`

For this purpose recall that a semilattice is a structure with a binary operation which is idempotent, commutative, and associative. Further, we extend $\|\cdot\|$ to the domain $S \cup \bot$ by $\|\bot\| = \emptyset$.

Theorem 5 (Correctness of `AIntersect`)

1. $(S, \texttt{AIntersect})$ *is a semilattice.*
2. *For all* $r_1, r_2 \in R$, $\|\texttt{AIntersect}(r_1, r_2)\| = \|r_1\|$ *iff* $r_1 \leq_S r_2$.

Proof Both proofs are routine inductions. We leave out the proof of (1) altogether. For (2) we proceed by induction on the structure of r_1. We cover a couple of representative cases:

$r_1 = (a\ r_{1,1} \cdots r_{1,n})$: We proceed by cases in r_2. The cases where r_2 is one of $(*)$, $\bot$, or atomic are resolved by symmetric counterparts of equations in fig. 1. Remaining are:

- $r_2 = (a'\ r_{2,1} \cdots r_{2,m})$: If $a \neq a'$ then $\|\texttt{AIntersect}(r_1, r_2)\| = \emptyset \neq \|r_1\|$ and $\|r_1\| \not\subseteq \|r_2\|$. If $a = a'$ we can assume that $m \geq n$ the case otherwise is symmetric. The conclusion now follows directly by the induction hypothesis.
- $r_2 = (\texttt{* set}\ r_{2,1} \cdots r_{2,m})$: We obtain $\|\texttt{AIntersect}(r_1, r_2)\| \neq \emptyset$ just in case exactly one $r_{2,i}$ has tag a, which is sufficient to establish the case. □

9 Conclusion

We have shown how a restricted syntax for the SPKI authorisation element can be defined such that general authorisations and entailments between authorisations can be decided in almost linear time. Moreover, the restricted syntax appears to follow existing SPKI practice, so no real restriction in expressive power or usage is incurred.

To which extent our results are important in practice can be discussed. The computation of `AIntersect` is simplified when queries do not involve unions, i.e. the `* set` construct. This is the assumption made, for instance, in the Pisces implementation (see url: www.cnri.reston.va.us/software/pisces/). At any rate, as long as authorisation expressions and certificate chains remain small, the overhead may be negligible. Moreover, SPKI's simple delegation model enables chaining to be decided in polynomial time [2].

So one may argue that the problem is in practice negligible. We do not think this point of view is necessarily valid. First, we have not found such a thing as a clear and well-established SPKI practice. Nothing in the draft standards prohibits the use of unions in requests, and this capability might very well be used in practice. Several examples can be given. For instance, an application programmer might wish to exploit the revocation predictability built into the SPKI framework by computing a set of requests in advance. Or, as another example, it might be deemed useful to use the union construction to introduce macros. For instance, `USLocs`, `MidWestLocs`, etc., might be introduced as macros (at the application level) representing S-expressions of the form e.g.

```
MidWestLocs =
(* set
...
(location Nebraska Lincoln)
(location Kansas Topeka Centre)
(location Kansas Topeka North)
(location Kansas Wichita)
...  )
```

There is no prior reason why such a macro might not appear as part of a request, say, to determine whether access to Midwestern branch office sales statistics is permitted or not. The result, however, can be serious per-

formance degradation at request time.

Going beyond SPKI as it currently stands there is also the possibility that new mechanisms, for instance for delegation (cf. [1, 6, 7]), will be introduced which require comparisons to be made between authorisations of a general shape. An important purpose of the present paper is to set the stage for further studies in this direction, in terms of an evaluation model with good computational properties.

Acknowledgements Thanks to Dieter Gollmann, Microsoft Research, Cambridge, also to Babak Sadighi and Roland Hedberg, SICS, and to Thom Birkeland at IMIT/KTH.

References

[1] O. Bandmann, M. Dam, and B. Sadighi Firozabadi. Constrained delegation. In *Proc. 23rd Annual Symp. on Security and Privacy*, 2002. To appear.

[2] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI, 1999.

[3] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI Certificate Theory, May 1999. RFC 2693, expired. URL: ftp://ftp.isi.edu/in-notes/rfc2693.txt.

[4] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. Simple public key certificate, July 1999. Internet Draft, expired. URL: http://world.std.com/ cme/spki.txt.

[5] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI examples, March 1998. Internet Draft, expired. URL: http://world.std.com/ cme/examples.txt.

[6] B. Sadighi Firozabadi, M. Sergot, and O. Bandmann. Using Authority Certificates to Create Management Structures. To appear in Proc. 9th Security Protocols Workshop, Cambridge, UK, April 2001.

[7] Jon Howell and David Kotz. A formal semantics for SPKI. In *Proc. 6th European Symposium on Research in Computer Security*, 2000.

[8] Ron Rivest. S-expressions, May 1997. Internet Draft, expired. URL: http://theory.lcs.mit.edu/ rivest/sexp.txt.

Public-key Support for Collaborative Groups

Steve Dohrmann, Carl Ellison
steve.dohrmann@intel.com, cme@jf.intel.com
Intel Labs

Abstract: *In this paper, we describe a use of public-key cryptography to achieve access control over communication and data transfers in order to support the work of collaborative groups. The participants form themselves into groups and access is granted to group members. The use of cryptography in this project is exceptional only in the care with which we designed the protocols for identity establishment. Our goal is to produce a working application that has the potential to be more secure than earlier alternatives, because it is easier to use correctly. This paper compares our identity establishment process, along the lines of SDSI, to that of an X.509 PKI or PGP, and shows the security advantages of the process we use. We also describe an experimental method for key verification intended to make strong key verification both easy and enjoyable for the average user.*

1 Introduction

This paper describes a project to build and run collaborative groups over ad hoc networks with strong access control for communications and data transfers, strong encryption for the privacy of those interactions and strong but easy to use administration of access control. It was our initial premise that cryptography and protocol development had achieved adequate security long ago, and yet weaknesses remained in fielded implementations that came primarily from human mistakes attributable to user interface elements [6], such as

1. confusion when the user is forced to deal with unfamiliar concepts,
2. mistaken identity when referring to people by name, or
3. the simple refusal to employ security features because of a distasteful user interface.

It was our intention to address these issues and thus make a family of devices that improve on the security offered by PKI-based mechanisms, such as PGP, S/MIME, and SSL. We want to handle corporate sensitive data with improved security while simplifying the user interface to the extent that an untrained user at home would use the system correctly. We stop short of implementing MAC (Mandatory Access Control) and labeling of data, although that is an area for future development.

The devices we use are PDAs and laptops. These are mobile computing platforms, and in our prototype implementation they are connected by wireless networking, although nothing in this design rules out interoperating with wired devices. Because we are using wireless networks, we have no control over who might connect to that network. We have no secure perimeter and therefore do not rely on one. In retrospect, this appears to be a good design choice even for wired networks, since it is becoming difficult, if not impossible, to establish a secure perimeter in wired networks as well.

We take as our paradigm of collaborative group the pattern we experience at Intel, where groups are formed to address tasks, perform their function and then dissolve when the function is complete. Such groups remain active anywhere from a half hour to years. These groups are formed via personal invitation (sometimes indirectly, via a referral from an invitee) and are constructed based on availability and needed skills without any special regard to the corporate organization chart. As a result, it is not uncommon for an individual to be a member of multiple groups and be the only participant in common among those groups. It is also not uncommon to meet more than one new person in each new group a person joins. These groups might address extremely sensitive matters, such as designs for new features for future microprocessors, but they might also address non-sensitive matters, such as planning an annual departmental party or raising money for a needy family. We assume that this model covers more than just Intel. It applies clearly to people's behavior at home. If there is a more structured work environment where task groups are constrained by an organization chart, such constrained groups can still fit into our model.

Although we envision creating small collaborative groups, typically the size of a group one would find in a conference room, the mechanism defined here scales easily to a community of any size. Meanwhile, even though the group may be small, the population from which we choose that group is large, up to the size of the global Internet. This introduces a naming problem, discussed in more detail below. It is that naming problem that would make a global PKI unacceptable for our purposes, even if such a PKI were to exist. Fortunately, from our experience there is no need for such a global PKI. Instead, we expect to see a

proliferation of the kind of public key authentication and authorization mechanism that we have implemented and that we describe in this paper.

This paper describes the full process of achieving strong authorization of communication and file access. In section 2, we cover physical discovery of other devices. In section 3 we cover the process of establishing identity of other participants, specifically of linking their identities as established biometrically with their keys as provided over the network. That section is perhaps the most controversial and accordingly occupies the bulk of this paper. In section 4, we describe the process of group formation, based on identities that have been established by the methods of section 3. In section 5, we list some of the uses to which these groups can be put. In section 6, we consider some user interface issues, especially the issue of key verification – something vitally important for security but something that most users find annoying and wish to skip entirely. In section 7, we give our conclusions and in section 8 we consider areas for future research.

2 Discovery

In our current implementation, we use laptops or PDAs with dual networks, one local-area (802.11 ad hoc) and one wide-area (GPRS). The discovery mechanism is different for the two, not merely because the underlying hardware is different but because the population size is radically different. Under 802.11, one would expect fewer than 200 machines within range. Under GPRS, there might be millions of users online (just as there would be on the whole Internet). In neither case do we trust information obtained by discovery without the further proof that is provided during the identity establishment phase, but in both cases we need to find the party with whom we intend to do that identity establishment.

2.1 GPRS Discovery

Discovery here is by sign-on name, a name programmed into the cellular card at time of service activation. It is by these names that the cellular provider identifies and catalogs subscribers. These names are arbitrarily chosen and not necessarily known by the person encountering the name, so they are not necessarily meaningful to users. They are used as indexes into a database, typically under verbal instruction.

As part of our project, we have created a directory to be operated by the cellular provider, in which we record presence information: whether a given subscriber is online at the moment and the current IP address of that subscriber. Discovery over GPRS is achieved by consultation of that directory. Write-access to that directory is authenticated strongly, via public key operations, using a key installed during provisioning and bound to the user's sign-on name. This key is empowered only to give directory access and is not used for other access control.

2.2 802.11 Discovery

With 802.11, there is no need for a sign-on name, but in order to be consistent across networks, we invent and use a sign-on name for the 802.11 discovery process as well. This is a potential weakness. There is a certain level of security provided by the GPRS discovery mechanism, since one must be strongly authenticated to place an entry in the presence directory. When the directory under GPRS returns an IP address for a given sign-on name, one can rely on the fact that the binding of name to IP address was strongly authenticated and was provided by the holder of that sign-on name. However, under 802.11, there is no authentication of the sign-on name. It is merely a claim. If the user had been trained by GPRS experience to rely on the validity of this name binding, this is not safe. We do not rely on our users to keep that distinction between GPRS and 802.11 in mind. As a result, a machine that has been freshly discovered over either network is assumed by our system not to have been authenticated at all and is not granted any restricted access until after Identity Establishment and Authorization.

3 Identity Establishment

During the **introduction** phase, we establish the identity of correspondents. The identifier we use is not the sign-on name, for two reasons.

1. We allow a user to generate multiple personae and use them as she sees fit, choosing which one to use in introducing herself, just as a user chooses which business card to beam from a PDA.
2. We do not believe in one-name-fits-all-uses. The login name, introduced in the 1960's (or even earlier), is a good method of identifying a person to a computer, but we have seen numerous failures in attempts to use such names to specify a person, through a computer, to another person.

The description of **introduction**, which follows, may seem pedantic and perhaps elementary. However, we have tried to show all our steps so that we can compare this process to that used by a more traditional global-name PKI such as X.509 or PGP.

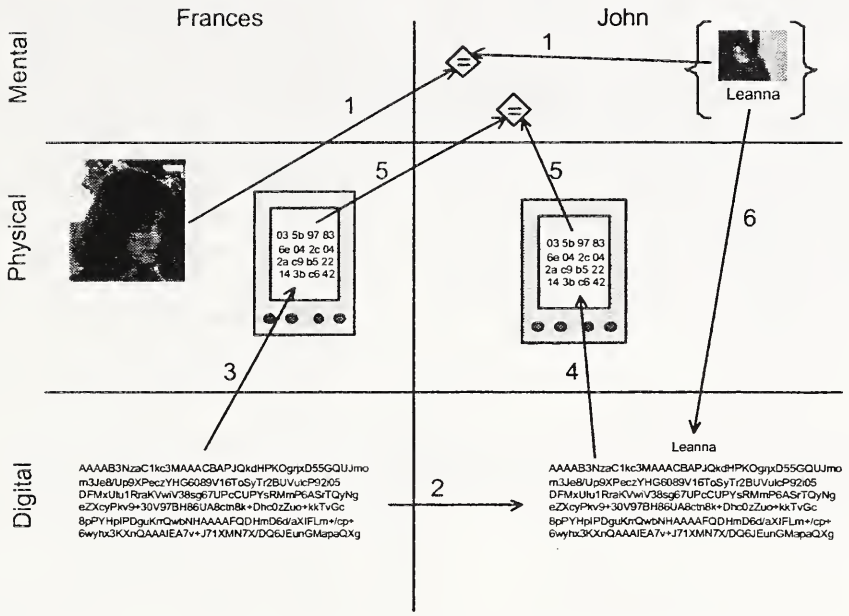


Figure 1: The Process of Establishing Identity

Within the computers and over the network, nodes in our networks are known by various transitory addresses, such as an IP address, but also by a permanent, globally unique ID: a public key associated with the user's chosen persona. The **introduction** job is therefore to establish the identity of that key.

By "establishing identity of a key", we mean **establishing that the key belongs to the person you think it does**. The phrase "the person you think it does" implies that you have some concept of the person. If you have never met the person, and therefore have no concept of him or her, the phrase has no meaning and you cannot establish identity. The most you can do in that case is to learn facts about that keyholder based on statements by some other party. However, here we are interested in establishing identity.

In our analysis of the introduction process, we look at three slices of reality:

1. **Digital:** things that reside in and happen inside computers and networks (keys)
2. **Physical:** people and things that have physical existence (people, computer screens), and
3. **Mental:** thoughts and memories inside a person's mind (knowledge about a person, biometric matching procedures, decision making, etc.).

"The person you think it does" exists in the mental slice of reality. It is a body of memories about the person in question. The purpose of introduction is therefore to establish a binding between a body of memo ries and a public key. This implies that the introduction phase requires personal acquaintanceship. Our system does not limit all system use to personal acquaintances of one person. Non-acquaintances are made accessible during the **invitation** phase. But, we do block system use by complete strangers (those not known to anyone in the collaboration group).

3.1 The Identity Establishment Process

Each person who is party to an introduction operates in three different spaces: one physical, one digital and one mental. For mutually establishing identity between two parties, there are then six spaces involved, and steps in the process used to establish identity must cross from one space to another. The boundary crossings must be considered carefully because they offer increased likelihood of errors.

In Figure 1 we show the process of introduction from Frances Chamish to John Wilson. Ms. Chamish does not use the name Frances, except on official documents, like her driver's license, passport and income tax return. With everyone else, she uses the name Leanna. So, we will refer to her by the name Leanna, unless we are being formal.

The process described in Figure 1 might be mistaken for the PGP key signing ritual, but it is different in that it does not assume knowledge or relationships that are not actually present. [The comparison to the process used by a traditional PKI like PGP or X.509 is given in section 3.2.]

The identity establishment process of Figure 1 has six steps.

1. John sees Leanna and since he knows her already, he compares the person he sees before him to a template stored in his memory. This is a biometric comparison, based on face or voice recognition and possibly other characteristics, processed by John's senses and brain, rather than some hardware biometric sensor. A similar biometric comparison would happen if the encounter between them were by telephone or videoconference.
2. As part of normal background activity, Leanna's and John's PDAs broadcast **discovery** messages containing their sign-on names and IP addresses. By mutual agreement, Leanna and John start the **introduction** phase by releasing their public keys and associated information to each other's PDAs, using the IP addresses learned during the discovery phase. [Figure 1 shows only one half of this exchange.]
3. John wants to change Leanna's key (an entry in his Contact List) from anonymous to known. This requires a verification phase. For Leanna's part of that phase, she displays verification graphics of her public key, on her PDA. [In Figure 1, this is shown as a key hash, but it could be any appropriate display carrying enough entropy to verify the key.]
4. John's computer simultaneously displays the verification graphics of the newly arrived key.
5. John compares these two images, by seeing them displayed on the two PDAs, held side by side (or if they are connected over a telephone connection, he listens to Leanna read displayed data or listens to her computer and his own simultaneously render verification data as sounds). From this, John now knows that the key he has selected in his PDA is the one belonging to Leanna. He knows this in his own brain, the same brain that established Leanna's biometric match in step 1. Therefore, those two match results are communicated to his decision-making without having to cross reality-slice boundaries.
6. With the success of the two equality tests, John gives a name to that selected public key using the name "Leanna", which is the name he uses to index his set of memories that include her biometric templates. This name comes from his memory and its sole purpose is to be a link back to his memory from his computer display. It does not have to be a name that anyone else would recognize as belonging to Frances L. Chamish. This binding of the name Leanna to her public key must be protected from tampering. John establishes that protection by leveraging the protection of his own private key. He creates a SDSI [5] name certificate binding the name "Leanna" to her public key and signs the certificate with his private key. After John has accepted and labeled Leanna's key, future encounters with her will not require any of the steps of this introduction process. Her key remains marked as fully introduced.

At the conclusion of this protocol, John has a Contact List entry that ties a public key to a body of memories, including one or more biometric templates, that stands for his concept of the person he calls Leanna. In other words, **he has established that the key belongs to the person he thinks it does**.

The relationship established here is immediately between John's mind and John's PDA's digital memory (with linkage by use of the name "Leanna"). There is a secondary linkage to Leanna's private key, by virtue of the fact that a given public key has only one corresponding private key, at least in our public-key algorithms. From there, there is a linkage to any digital signature made by Leanna's private key, and from there to any message or file thus signed.

This process has been tuned to link information via identifiers appropriate to the domain in which they are used. Between John's mind and his PDA's memory, a local name, meaningful only to him, is used. Between John's PDA and Leanna's PDA, a globally unique identifier (the public key) is used.

3.2 Establishing Identity via Traditional PKI: X.509 or PGP

X.509, PGP and SDSI ID certificates differ in format, process and meaning. The difference in format is irrelevant for this paper. We focus on the difference in process and meaning. Most especially, we note that both X.509 and PGP deal with globally unique IDs that are expected to be meaningful to whoever intends to use the key. Since this ID carries a global meaning, the binding of ID to key is an act that must be performed by a trusted service. In X.509, that trusted service is a specially trusted Certificate Authority (CA). In PGP, that trusted service is a collection of less trusted key signers who, taken together, constitute a distributed trusted service (the web of trust). By contrast, the SDSI (local) names we use are intended to have meaning only for the person who creates the name and binds it to a key. That one person is the sole authority on this name

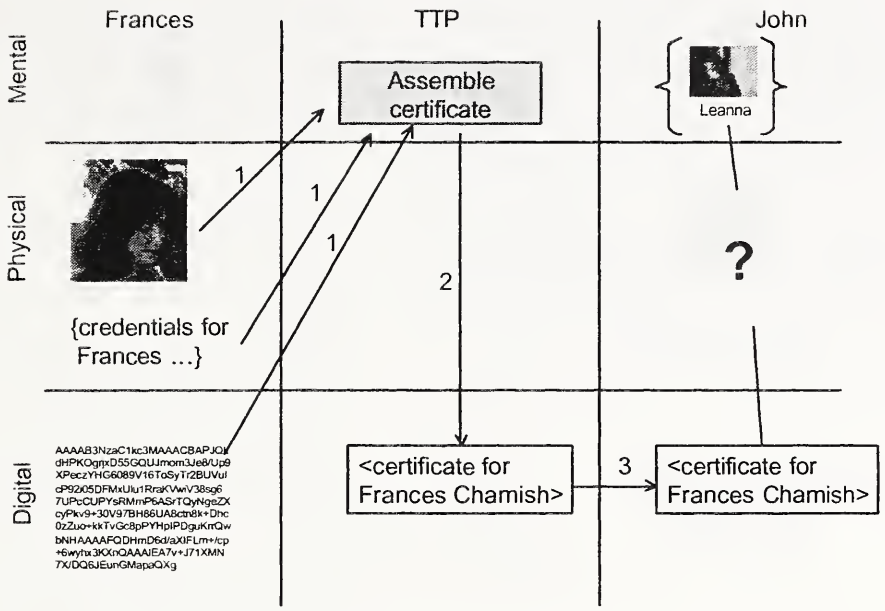


Figure 2: Establishing Identity via PKI

binding and therefore the only one who can bind that name to a key.

In Figure 2 the "person" labeled "TTP" stands for a Trusted Third Party and can be either an X.509 CA or a set of PGP trusted introducers.

3.2.1 TTP Process: Leanna to John

The process of Figure 2 appears simpler than the process of Figure 1, because it omits the detail effort involved in creating a certificate. In the case of PGP, for example, that effort often involves the hash computation and comparison steps shown in Figure 1.

The process shown in Figure 2 is:

1. Leanna takes various credentials and a copy of her public key to the TTP. At PGP key signing parties, those credentials might include a driver's license or passport. By means of these credentials, Frances lays claim to her true name. That is, she demonstrates to the TTP that she is not impersonating someone else. These official credentials all list Leanna as "Frances Chamish", some using the middle initial "L".
2. The TTP instructs his or her computer to generate a certificate binding Leanna's name, "Frances Chamish", to her public key. In the case of PGP, the certificate construction will have been done already by Leanna and the TTP(s) merely sign(s) that certificate body. In the case of an X.509 CA, the TTP builds the certificate and most likely chooses a name for Leanna in the process. PGP does not require that IDs in certificates be globally unique, but X.509 practices often require name uniqueness, at least over the set of individuals certified by that CA. As a result, the X.509 certificate will bind a Distinguished Name (DN) to the public key, where that DN may include the name Frances Chamish but may also include other information to make the DN unique.
3. The certificate issued to/for Leanna is delivered to John at a time when John is not in direct contact with Leanna and he must make a decision based on the information contained within that certificate. This delivery can be via a directory service (e.g., the PGP key server or some directory of X.509 certificates) or from Leanna as part of a communication (e.g., via S/MIME). If he is acting properly, he will fail to make any connection between the certificate and his memory of Leanna, since the two have too little information in common to confirm with high probability that they refer to the same physical person.

Note that PGP has a slight advantage here. Under PGP, Leanna chooses the name she wants bound to her public key and needs only to convince some number of key signers to sign that association. On the other hand, a high quality PGP key signer should refuse to sign a key with a name not backed up by official documents.

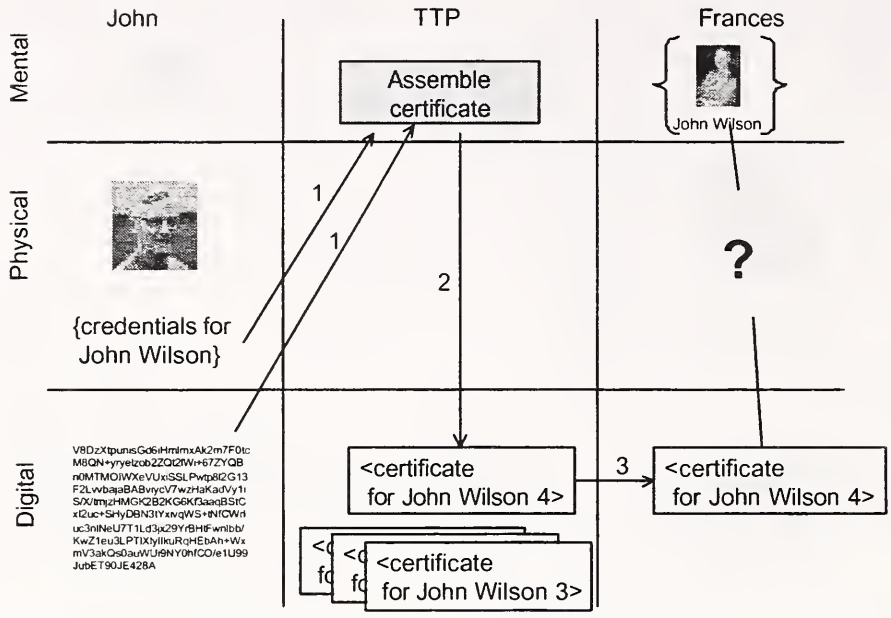


Figure 3: Second PKI Example

3.2.2 TTP Process: John to Leanna

In the other direction, there is a different problem. John Wilson uses the same, true name in all his official credentials, on all his documents and with all people. But, in Figure 3 we see that Leanna is still unable to connect his certificate to his identity in her mind. Although the names compare between the certificate and Leanna's memory, Leanna does not know which of the TTP's John Wilsons this certificate corresponds to. She knows only one John Wilson, but the TTP might know and have certified hundreds. It is true that a good CA will make the certificates for each John Wilson different, by including additional information beyond the common name "John Wilson". (That information is shown in Figure 3 as serial numbers.) However, if Leanna does not know this additional information about John, then all of these certificates would equally match Leanna's memory of John and therefore the certificate in Leanna's computer could be for any of those John Wilsons. In the best case, she will discount the certificate as worthless to her because she knows she doesn't know which John Wilson it belongs to, but there is a more serious threat. She does not get all of the certificates issued to all the John Wilsons. She gets only one, especially if it is delivered (e.g., by S/MIME or SSL) from someone claiming to be John Wilson. If she were a naïve user, she might not think about the hundreds of other John Wilsons that the TTP could have certified and, since she knows only one John Wilson, accept the offered certificate as referring to the John Wilson she knows. That is, she might assume that she has verified John's identity via that certificate when she hasn't.

By contrast, when Leanna creates a SDSI name certificate with the name "John Wilson" by the process of Figure 1, since she knows only one John Wilson she knows to which John Wilson her certificate refers. If she knows more than one John Wilson, then she must choose additional information to append to the name to make it unique for her, just as a CA needs to do. However, she will choose information that she knows and that should therefore be meaningful to her when she gets around to using that certificate in the future.

3.3 Security of Private Keys

There may be suspicion of the personal introduction processes we use for their lack of use of a CA. As we have shown above, the use of global names that comes along with using a CA adds substantial insecurity to the introduction process. However, an X.509 CA is expected to be very good at protecting its own keys. In our mechanism, by contrast, certificates are generated by keys that are not specially protected. In PGP, the key signers do not specially protect their keys, but the fact that a key is supposed to be signed by multiple signers (the web of trust) implies that any attacker must have compromised all of those keys. PGP aims to achieve through redundancy what an X.509 CA tries to achieve through a guarded vault. At some number of signatures, the attack effort required becomes greater for PGP than for an X.509 CA and therefore the strength of PGP would be greater. Our certificates have neither form of protection.

In spite of the relatively unprotected signing keys in our mechanism, we can show that we have lost no security for lack of the TTP. At the same time, as shown in the previous sections, we would have lost security via naming had we used either of the global-name ID mechanisms.

Our argument is that if an attacker can steal (or operate at will) a user's private key, that attacker can impersonate the user as well as generate certificates. Since confidentiality keys are established in our system by signed Diffie-Hellman key agreement[3], forward secrecy is maintained and the attacker does not gain access to any past (recorded) messages or file transfers.

This is not to deny the severity of theft of a private key. The ability to impersonate the attacked user is a wide security breach. The ability to generate certificates as that attacked user, however, does not give any extra access. No user in our system is in the role of a TTP – certifying memberships, IDs or authorizations that the attacked party does not herself possess and therefore that the attacker does not himself possess after theft of her key.

If the attacker chooses to use the stolen key to generate a certificate for his own key, to invite it to join a group (see section 4, below), then the attacker would have access to activities of that group as a full participant without continued use of the stolen key. However, he would also leave a trail of use of his own key. That key, although not tied to any locator information, is an identifier and has forensic value. Therefore, a savvy attacker would continue to impersonate the attacked person by using her stolen key, rather than generate a certificate giving group membership to his key.

In summary, the theft of a private key is undesirable, but the ability of the thief to generate certificates gives the thief no powers beyond those already gained just by possession of the private key and might, in fact, work against the attacker. A TTP would not increase private key security on an individual node. It would only increase certificate-issuing security, and therefore is of no benefit to us.

4 Group Formation

We start with the concept of a secured collaboration, or **collaboration** for short. A **collaboration** is a group of principals, known as **members**, who are permitted to share messages and files as part of that collaboration. Some of these members also have the permission to add new members to the collaboration.

A collaboration starts out as a name in the namespace of the creator of the collaboration. It is expressed as an SPKI/SDSI name: "(name <public key> <ASCII name of collaboration>)". [4]

The creator of a collaboration might be a private individual, creating a set of friends, or a project leader in a corporation, creating a digital reflection of her project team. The official or unofficial nature of a collaboration is a function of the intention of the creator and does not show up in any difference in the software used.

Given correspondents who are known with assurance, the process of **Invitation** is that of granting authorization to those known correspondents to participate in a secure collaboration. An invitee can be granted membership in the collaboration and might also be granted the right to invite others into that collaboration.

We grant membership without permission to add new members by creating an SPKI/SDSI ID certificate:

```
(cert
  (issuer (name <public key> <ASCII
name of collaboration>))
  (subject <public key of invitee>)
  (valid (not-after <end date>))
)
```

We grant the ability to add new members as well by issuing the certificate:

```
(cert
  (issuer (name <public key> <ASCII
name of collaboration>))
  (subject (name <public key of
invitee> <large random value>))
  (valid (not-after <end date>))
)
```

That is, we create a named group in the grantee's namespace and add that named group to the collaboration. That grantee then adds individual members to that new named group, via certificate:

```
(cert
  (issuer (name <public key of
invitee> <large random value>))
  (subject <public key of next
invitee>)
  (valid (not-after <end date>))
)
```

The members of a collaboration are those public keys that are direct members of the top level named group or of some named group contained within that top level group, at whatever nesting depth.

4.1 Cross-corporate Invitations

In our system, invitations are issued only to acquaintances, but these do not have to be close

personal friends. These can be people one had met for the first time just prior to issuing the invitation.

Such might be the case with cross-corporate working groups, such as standards bodies, corporate acquisitions or venture capital funding activities.

The invitation process does not require an act of the IT departments of the various corporations involved. It does not give any access into any of the corporations by members of the other except for the strictly limited functionality of the collaboration for which the invitation was issued. In this way, it models current business practices.

Other PKI mechanisms for permitting cross-corporate interactions do not share this attribute. A bridge CA [1], for example, effectively merges the certificate space of the two bridged corporations. The very existence of the bridge CA might, in fact, leak sensitive information (for example, evidence that an acquisition or merger is in the secret negotiation stage).

By contrast, with the **invitation** process, corporation A learns nothing about the employee database of corporation B. Members of corporation B are represented in the group as public keys. No names of keyholders are exchanged as part of the invitation. One does not know if a second key invited by someone in corporation B was that of another employee or was a second key of the original employee. Therefore, one does not even learn anything about the headcount of corporation B beyond that which was learned during the in-person negotiation meeting(s) during which the **introduction** phase crossed the inter-corporate boundary.

5 Use

From the point of view of the user, the collaboration tool is just another instant messaging tool that happens to operate over dual networks and offers peer-to-peer file sharing. It happens to have a peculiarly rigid introduction process, but we are tuning the prototype to make sure that that process is not onerous.

The user has no choice over whether or not to use cryptography and, if so, how strong. User keys are all 1024-bit DSA. All messages and file transfers are encrypted with 168-bit triple-DES CBC, with session keys and IVs derived from 1024-bit D-H key agreement. All messages and file transfers are digitally signed. This use of cryptography is transparent to the user.

Full details of the features of this prototype belong in a product data sheet rather than this paper, but that data sheet has not been written yet. In summary, then:

1. Users can send messages to
    a. an entire named group,
    b. a set of members of a named group, or
    c. a single member of a named group
2. Users can make files available to a named group
3. Users can fetch a file that is available to a named group from the machine that holds it
4. Users can send files as if attached to a message (i.e., addressed the same way)

With every operation, a group must be specified. It is the named collaboration group that constitutes the only access control at this time. That is, in order to keep the UI simple, we provide for only one level of access control. If you are in the group, you can read any message or file made available to that group.

Each computer in a group maintains state for that group, including the list of group member keys and the list of any files that have been made available to the group. Whenever two group members regain contact, they synchronize this group state. The synchronization is automatic and gives users the impression of common state, although at times of network partition, that common state loses consistency.

The resulting use model is very basic and we hope easy to understand. Wider trials of the prototype will let us confirm that hope or give us information with which to improve the user's experience.

6 User Interface issues

It is essential to do proper cryptographic engineering, both in writing code and in designing protocols. However, that careful engineering is not sufficient to achieve security in an end-user product [6]. The user interface needs to be designed in such a way that the user would naturally do the correct thing and avoid doing the wrong thing.

We must assume that there is always an attacker trying to gain access to our collaborations, even though we realize that in most cases there will be no attackers. This lack of evidence of attack makes motivation of the user especially difficult. It is therefore incumbent upon us to make the user interface as pleasant and simple as possible

Computer software engineers, no matter how well meaning, cannot be expected to get a user interface right. There must be extensive testing, with real users. We have just started that extended testing and cannot report full results at the time of this writing. However, we have learned a number of things that are worth reporting here. These are cases where lessons we learned go against the inclination of our own developers.

6.1 Minimizing Choices

We have found that we need to minimize choices and options, especially when there might be a bad choice. Our initial users are more comfortable when given fewer options.

We have limited options by defining a Contact (a Java Object) that goes through state changes. It starts out, after **discovery**, as an anonymous, non-trusted thing. It has a sign-on name and may have an IP address. The only thing that can be done with this non-trusted object is to engage in **introduction**.

After introduction, the Contact has a public key that has been verified and named by the user. An introduced Contact is only then available to participate in **invitations** to join one or more named collaboration groups.

All message traffic and file transfers are associated with a named group and are limited to members of that group. It is not possible to engage in messaging or file sharing outside a named group.

Groups and Contacts are shown to the user as names, but the state of a Contact is shown by color and icon so that the user does not need to look beyond the top-level screen to tell what can be done with the Contact.

One invites a Contact to join a group by dragging and dropping the Contact name onto the group.

6.2 Sign-on Names

For security purposes, it is best not to display sign-on names to the user at all. These names are weak identifiers at best and are subject to the John Wilson problem, described in section 3.2.2.

On the other hand, both developers and experienced users have been well trained to use sign-on names. Many users view sign-on names as a way to deliver a message – e.g., the name "fundude", or the name "fund00d" that conveys a slightly different message.

In the prototype, we have compromised. We use sign-on names during discovery, but have the person who is building a personal Contact List choose a name for each entry in that list. This name will probably turn out to be the original offered sign-on name most of the time and we expect that to be a potential weakness, due to the John Wilson problem, with or without actual attacks.

We take it as ongoing work to look for a solution to this problem that is acceptable to users.

6.3 Key Verification

We recognized early in the development process that key verification (e.g., the comparison of hex key fingerprints) is the geekiest, slowest, most painful and most cumbersome part of the **introduction** process.

This can be made a little easier by converting key hashes to lists of words to be read aloud, as PGP did several versions back. However, the task is still time consuming. The list of words from a SHA-1 hash on a PGP key takes on the order of 24 seconds to read. The hex version of the hash takes about the same length of time.

When keys are verified over a telephone connection, in our prototype, we currently have the two parties read words alternately to each other, to achieve mutual verification of the hash. However, when keys are verified by placing two mobile computers next to each other, so that the person receiving the key can verify correct receipt, we can use a graphical mechanism that permits entropy comparisons to be much faster.

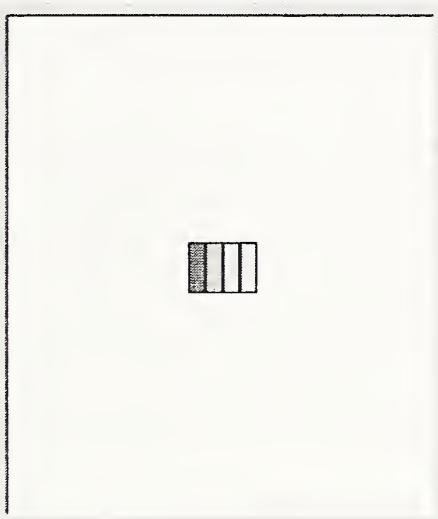


Figure 4: Verification graphic

Figure 4 shows a PDA screen displaying a graphic that we call a "flag". Preliminary experiments show that people can compare a time sequence of these apparently random graphics on two side-by-side screens, at a rate of 2 per second, with comfort. Assuming the verifier is not color-blind, each flag carries 25 bits: 1 for horizontal vs. vertical orientation; 6 for the color of each rectangle. This rate needs to be confirmed by more extensive testing, but assuming it is confirmed, this permits a key hash comparison at 50 bits per second. One can then compare a full 160-bit hash in just over 3 seconds, for a speed-up of a factor of 8.

If the graphic is black and white, e.g., for fully color-blind users, we expect to get at least 10 bits/second of comparison, for a full 160-bit hash in 16 seconds, but we have not yet experimented with shapes to see how much more rapidly we can do comfortable entropy comparison.

It is our goal to get the verification time low enough that a user would verify the correctness of a key's hash in the time it takes to move a stylus or mouse to accept a key as valid. This does not eliminate the verification

step, but does permit it not to add time to the user's process.

7 Conclusions

The problem of making sure that only those who truly should be authorized to access some data actually end up with access to that data is a very hard problem in general. We have addressed a subset of the family of security policies that have this requirement. We provide for policies in which every member of a defined group is permitted the same access as every other member, but we allow for the definition of an arbitrary number of groups. We have been very careful to make sure that groups are made up only of individuals known personally by someone with the authority to add to the group membership. This does not cover all possible groups, in theory, but does cover all groups we encounter in practice, both at work and at home.

Prior to this work, we had observed that the greatest leakage of confidential information came from misdirection of communication, through name confusion, and only secondly from a failure to employ security mechanisms to protect data. To respond to those problems, we have been careful to keep the named people and groups that any individual must deal with down to the personal acquaintances and group memberships of that individual. No choices are made from a larger namespace. We allow the individual to choose his or her own names for these individuals and groups, to minimize confusion of names. We have made all communications encrypted and digitally signed, with no user choice, so that all accesses to data handled by this system must be via the access control mechanisms we have defined.

This system is doubtless not perfect. However, it has addressed the greatest needs we have identified and further improvements can follow as we gain experience with use of this system.

8 Future Work

We have chosen not to deal with revocation of keys or of authorizations (group memberships). The underlying SPKI mechanism supports a variety of revocation methods, but the complication of the user interface did not seem warranted for what are almost always short-lived groups of long-lived keys.

The limitation of operations to group members, and the labeling of files as available to a group, can be thought of as a poor man's MAC/DAC architecture. We could possibly improve the security of our mechanism by implementing it on top of an operating system that supports data labeling and mandatory access controls.

We need to continue our user trials of key hash comparison mechanisms, including audio trials alongside graphical ones, in order to determine the actual number of bits being compared per second by the user.

Because our underlying engine is the AuthCompute library from CDSA [2], we have the full power of SPKI and SDSI at our command. However, we have not found a reason to use all that power. It is still an open question whether the refinement of access controls full SPKI would make possible would be of use to a naïve user base or whether it would add an unacceptable amount of confusion. For example, it is possible to use the SPKI **threshold subject** mechanism to have more elaborate security policies – such as permitting access only if a group member is also still employed and has a non-revoked key. However, this functionality would require a complication of the user interface and might lead to more errors than the extra refinement of authorization would prevent.

9 Acknowledgements

We want to thank Leanna Chamish and John Wilson for granting permission to use their names, stories and images in this paper. We especially want to thank the many engineers within Intel Labs who collaborated in producing the prototype whose characteristics we report here.

10 References

[1] Bridge-CA: for one discussion of a bridge CA, see http://www.bridge-ca.org/english/index.html

[2] CDSA: http://developer.intel.com/ial/security/

[3] Whitfield Diffie and Martin Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, November 1976, pp. 644-654.

[4] Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen, "SPKI Certificate Theory", RFC2693, September 1999.

[5] Ronald L. Rivest and Butler Lampson, "SDSI - A Simple Distributed Security Infrastructure", September 1996, http://theory.lcs.mit.edu/~rivest/sdsi10.html

[6] Alma Whitten and J.D. Tygar, *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0*. Proceedings of the 8th USENIX Security Symposium, August 1999, http://www.usenix.org/publications/library/proceedings/sec99/whitten.html

Authorization Policy in a PKI Environment

Mary R. Thompson, Srilekha Mudumbai, Abdelilah Essiari, Willie Chin

National Energy Research Scientific Computing Division
Ernest Orlando Lawrence Berkeley National Laboratory
Berkeley, CA, 94720
pkidev@george.lbl.gov

Abstract

The major emphasis of Public Key Infrastructure has been to provide a cryptographically secure means of authenticating identities. However, procedures for authorizing the holders of these identities to perform specific actions still needs additional research and development. While there are a number of proposed standards for authorization structures and protocols. [17, 5, 22, 10, 6] based on X.509 or other key-based identities, none have been widely adopted. As part of an effort to use X.509 identities to provide authorization in highly distributed environments, we have developed and deployed an authorization service based on X.509 identified users and access policy contained in certificates signed by X.509 identified stakeholders. The major goal of this system, called Akenti, is to produce a usable authorization system for an environment consisting of distributed resources used by geographically and administratively distributed users. Akenti assumes communication between users and resources over a secure protocol such as secure socket layer (TLS) which provides mutual authentication with X.509 certificates. This paper explains the authorization model and policy language used by Akenti, and how we have implemented an Apache authorization module to provide Akenti authorization.

Background

There is significant and growing set of distributed computing environments where the resources, resource stakeholders and users are geographically and organizationally distributed. The DOE sponsored Collaboratories [1] and various "Computational Grids" [13] are examples of these as well as the ubiquitous Web-controlled sets of documents and services. These systems effectively define a *Virtual Organization* whose members and resources span many different real organizations. These virtual organizations need a way to authenticate and then authorize their users.

One of the characteristics of a collaboratory or Grid is that both the stakeholders and users may come from many different administrative domains. Thus the virtual organization needs to identify its users in a domain neutral manner. The most common candidates for cross-domain identities are Kerberos and PKI. Kerberos is mostly used within a single administrative domain, but there are many examples of cross-authenticated Kerberos realms, where the Kerberos administrators have agreed to accept tokens from another realm. Negotiating cross-realm agreements is often a lengthy and complex process. Some examples of such domains are universities where there may be multiple Kerberos realms within the university, and the DOE's ASCI-DisCom^2 program [9] that connects Lawrence Livermore National Laboratory, Los Alamos National Laboratory and Sandia National Laboratories in a computational Grid.

Looser collaborations, such as Grids based on Globus [14] middleware, [24,27] Collaboratories [8,25] and portals [20] have chosen to use PKI identities to authenticate members. These organizations either run a Certificate Authority of their own and/or accept certificates from a set of trusted CAs. Establishing trusted CA relationships can also be a lengthy process, but since many current collaboratories and grids are experimental in nature, the trust relations have been established on an informal basis by the researchers, rather than the system security administrators. Once a collaboration has decided to use PKI identities to authenticate users, it needs to develop an authorization system using those identities plus some additional access policy information for each of its resources.

This work is supported by the U. S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Mathematical, Information and Computation Sciences office (http://www.er.doe.gov/production/octr/mics), under contract DE-AC03-76SF00098 with the University of California.See the disclaimer at http://www-library.lbl.gov/teid/tmRco/howto/RcoBerkeley-LabDisclaimer.htm.This document is report LBNL-49512.

Another characteristic of collaboratories and Grids is that their resources, such large scientific instruments, computing resources and data stores, may have more than one person (called a stakeholder) who needs to control access to the resource. For example, when remote control of an instrument is allowed the instrument administration may want assurance that any user who can control the instrument has passed a local training course, while the principal investigator may be mostly concerned that the person controlling the instrument during his allowed time is a member of his research group. An authorization system that allows access policy to be defined independently and remotely from the resource gateway is needed.

However, standard access control methods typically require that the stakeholder has privileged access to the machine on which the resource resides to set the access control. Also such systems, to the extent that they use the underlying operating system for actual access control, require that all users of a shared resource must have a local account on the system. The requirement for individual system accounts on the resource machine does not scale well.

We have developed the Akenti [32] authorization system to meet these two needs: to use a virtual organization-wide user identity (in our case an X.509 identity certificate); and to facilitate setting access policy by multiple independent stakeholders remote from the actual resource gateway.

This paper explains the authorization model and policy language that we use, and how we have implemented an Apache authorization module to provide the same authorization policy and mechanism for resources accessed via a Web browser as accessed by other remote methods such as Globus job submission [14] or CORBA object invocation.

Akenti

Akenti is built using X.509 identity certificates [18] and the SSL/TLS [7] connection protocols to securely identify a user that is requesting access to a resource. It represents the authorization policy for a resource as a set of (possibly) distributed digitally signed certificates. These policy certificates are independently created by authorized stakeholders. When an authorization decision needs to be made, the Akenti policy engine gathers up all the relevant certificates for the user and the resource, validates them, and determines the users rights with respect to the resource.

Authorization model

The Akenti model consists of *resources* that are being accessed via a *resource gateway* by *users*. These users connect to the resource gateway using the SSL handshake protocol to present authenticated X.509 identity certificates. The *stakeholders* for the resources express *access constraints* on the resources as a set of *signed certificates*, a few of which are self-signed and must be stored on a known secure host (probably the resource gateway machine), but most of which can be stored remotely. These certificates express what attributes a user must have in order to get specific rights to a resource, who is trusted to make such Use-condition statements and who can attest to a user's attributes. At the time of the resource access, the resource gatekeeper asks a trusted Akenti server, what access the user has to the resource. The Akenti server finds all the relevant certificates, verifies that each one is signed by an acceptable issuer, evaluates them, and returns the allowed access. See Figure 1.

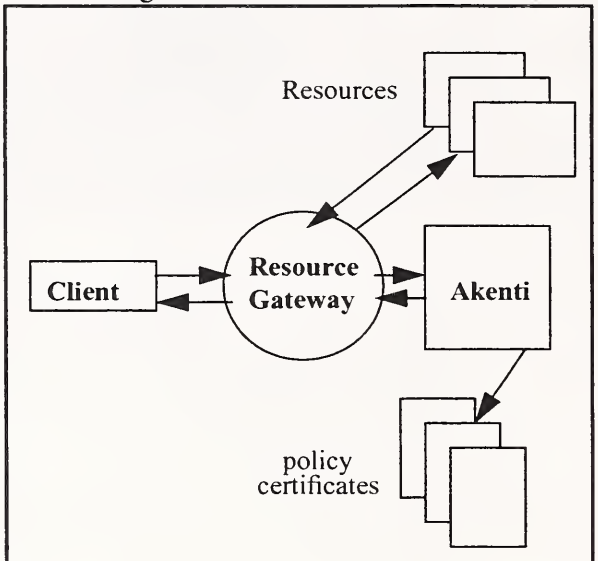


Figure 1. Akenti Authorization Model

There are several models for arriving at access control decisions. One is the classical access control list model, where the user just presents an identity to the gatekeeper who finds the policy information for the resource and evaluates the users access. Another is the capability model, where the user presents a capability which grants the holder specific rights to the resource, and the gatekeeper has to verify that the user has come by the capability legitimately and then interpret the rights that have been presented. There are also hybrids of the two models, where a user may present some identity information and possibly a restricted set of his full rights.

We have mostly concentrated on the first model in order to allow applications to use Akenti authorization over standard SSL connections which can transport and verify X.509 identity certificates. We have also experimented with s capability model where Akenti will return a signed capability certificate containing a subject's Distinguished Name (DN), public key, the Certificate Authority (CA) that signed for this name, the name of the resource and the subject's rights. If this is presented to a resource gatekeeper, along with an authenticated identity certificate, the gatekeeper need only verify the signature of the certificate by using its copy of the Akenti server's public key, and verify that the subject named in the capability is the same as that in the identity certificate. These capability certificates are short-lived in order to avoid the problems of revocation.

Akenti policy language

Akenti policy is expressed in XML and stored in three types of signed certificates: *Policy certificates, Use-condition certificates* and *Attribute certificates*. Policy certificates are self-signed, co-located with the resources to which they apply and contain only minimal information. Use-condition certificates contain the constraints that control access to a resource. Attribute certificates assign attributes to users that are needed to satisfy the use constraints. Akenti attribute certificates are simpler than the proposed IETF Attribute certificates. See the section on Related Work for a more detailed comparison. See Figure 2 for an example of a Use-condition certificate and Appendix A for the DTD definition of the complete Akenti Certificate schema.

Policy certificates specify who the resource stakeholders are, and thus who may sign Use-condition certificates. The Use-condition certificates specify who can attest to the required attributes and thus who may sign Attribute certificates. Whenever a certificate is used, the Akenti policy engine will check that it has been signed by an acceptable issuer, and that the signature verifies.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE AkentiCertificate SYSTEM "/home/g1/proj/akenti/release/common/AkentiCertificate.dtd">

<AkentiCertificate>
 <SignablePart>
 <Header Type="useCondCertificate" SignatureDigestAlg="RSA-MD5" CanonAlg="AkentiV1">
    <Version ver="V1"/>
    <ID id="griffy.lbl.gov#4e6ba338#Mon Mar 01 10:56:51 PST 1999"/>
    <Issuer>
       <UserDN> /C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=Mary R. Thompson </UserDN>
       <CADN> /C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=IDCG-CA </CADN>
    </Issuer>
    <ValidityPeriod start="981224003646Z" end="020123003646Z"/>
 </Header>
 <UseConditionCert scope="local" enable="false">
  <ResourceName> LBL </ResourceName>
  <Condition>
    <Constraint>( o=Lawrence Berkeley National Laboratory | ( group = distrib ) ) </Constraint>
    <AttributeInfo type="X509">
          <AttrName> o </AttrName>
          <AttrValue> Lawrence Berkeley National Laboratory </AttrValue>
          <CADN> /C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=IDCG-CA </CADN>
    </AttributeInfo>
    <AttributeInfo type="AKENTI">
          <AttrName> group </AttrName>
          <AttrValue> distrib </AttrValue>
          <Principal>
            <UserDN> /C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=Srilekha Issuer </UserDN>
             <CADN> /C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=IDCG-CA </CADN>
          </Principal>
    </AttributeInfo>
  </Condition>
  <Rights> read, write </Rights>
  <SubjectCA> /C=US/O=Lawrence Berkeley National Laboratory/OU=ICSD/CN=IDCG-CA </SubjectCA>
 </UseConditionCert>
 </SignablePart>
</AkentiCertificate>
```

Figure 2. UseCondition Certificate

Resources controlled by Akenti authorization may be grouped into a *resource realm*. A resource realm can be organized as a flat structure of resources such as instruments or compute platforms, or a hierarchical structure such as a file system or set of Web documents. Each resource realm has at least one Policy certificate which must be stored in a known and secure place. Normally it is on the same machine that controls access to the resource, but it could also be on the platform where the Akenti server is running, if they are different. Since a Policy certificate is centrally stored and may be administratively difficult to update there is a minimal amount of information in it. It contains information about the Certificate Authorities that are trusted to sign identity certificates, including a copy of their public keys and information about where they publish certificates and certificate revocation lists. It also lists the stakeholders (or stakeholder groups) for the resource and where they store the Use-condition certificates that they issue. It may optionally store URLs in which to search for Attribute certificates.

In the case of hierarchical resources, there must be at least one Policy certificate at the top of the tree (sometimes referred to as the root policy). Then there may be a Policy certificate at any level where there are new stakeholders, or restrictions on the allowed CAs. Levels without their own Policy certificates inherit policy from higher levels. Policy certificates are signed by one of the stakeholders listed in the certificate, making them self-signed certificates. As such they must be uploaded by a trusted method and kept in a secure location.

Each stakeholder group for a resource must create at least one and possibly more Use-condition certificates for its resource. A Use-condition certificate consists of a constraint which is a relational expression of the attributes a user must have to get a certain set of rights with respect to the resource. Components of the X.509 distinguished name can be used such as CN=Mary R. Thompson, or O=Diesel Combustion Collaboratory, or values of attributes that are defined in the context of the resource. For example, role = researcher or group = accounting. These attribute requirements can be combined with the boolean operators && or ||. It is also possible to specify real-time or system parameters such as time<=5PM && time>=9AM, or system_load < 2. If Akenti is unable to evaluate such system parameters it may pass them back to the resource gateway for evaluation. An attribute authority (consisting of an issuer and its CA) is specified as the signing authority for each attribute-value pair. Thus the stakeholder for a resource must specify who is trusted to attest to the attributes that are required.

The Policy certificate contains URLs to search for each stakeholder group's Use-condition certificates. A stakeholder may put Use-condition certificates in more than one place for reliability, but each directory must contain the complete set. Since Use-conditions restrict access to a resource, it is essential that either all or none of them are found. If no Use-conditions are found for a stakeholder group, all access to the resource is denied. This is not the case with Attribute certificates since they only serve to increase access. Thus a missing Attribute certificate may limit or deny a user's access, but will never allow an access that should be denied.

Attribute certificates contain an attribute-value pair and the subject name and issuer to whom it applies. They are signed by attribute authorities that have been specified in a Use-condition certificate. Attributes can apply to more than one resource, although they are likely to be applicable in only a single resource realm.

Creating policy

Since policy is contained in signed XML certificates which are interdependent, a stakeholder needs some tools to assist in their creation. A stakeholder starts by creating the root Policy certificate for the resource realm. The X.509 certificates of all the trusted CAs must be available from a trusted source and are placed in the root Policy certificate. This certificate also contains the URLs of the locations where these CAs publish the certificates that they issue and their certificate revocation lists. The first stakeholder must decide if other stakeholders for the resource are to be allowed and, if so, include their DNs and CAs in the root Policy certificate. In a hierarchical set of resources, only the top level stakeholders need to be known initially. They in turn, can delegate control to other stakeholders for resources lower in the hierarchy.

Akenti certificates can be created either by using a command line tool to sign an XML input certificate, or by a GUI program that steps a stakeholder though a menu of choices for each field in the certificate. The GUI program is supported by a Resource Definition Server running on the resource host which in turn reads a Resource Definition File and any existing Policy files to find stakeholder names, acceptable attributes and actions for a resource realm. The command line method is fine for very simple policy, and for the root Policy certificate, but as soon as the policy becomes hierarchical, or there are many stakeholders, the GUI interfaces which prompt the stakeholder with acceptable choices become preferable. The Resource Definitions File is only used to provide suggestions to the policy creation GUIs. It includes the names of the CAs, and their publishing directories,

principals that are acceptable for issuing specific attribute and values, and a list of actions that are relevant to the resource realm. Information that is used at access decision making time, such as the certificates of the CAs, must be stored in the root Policy certificate, since it is a signed document. In summary the two methods of getting started are:

- Create an XML version of a root Policy certificate, following one of the templates provided by the Akenti distribution, sign it using CertGen with the stakeholder's private key contained in a pkcs12 format file, and store it in the resource tree
- Create a Resource Definition File, start the Resource Definition Server, and then use the GUI, *PolicyCert.sh* to create, sign and store a Policy certificate.

The stakeholder must now create at least one Use-condition certificate for the resource. Anyone can create a Use-condition certificate, but it will only be used during the access control decision if it is issued and signed by one of the stakeholders currently listed in the resource's Policy certificate. As in the case of the Policy certificate, a Use-condition certificate can either be created by inputting an XML version of the certificate and private key to CertGen or can be generated and signed by a GUI program, *UseCondition.sh.* The GUI program uses the Policy certificate to determine the allowed stakeholders, and the Resource Definitions File to determine what attributes, values and actions have been defined for this resource realm. The stakeholder is led through a process to specify who he is, where his private key is, what resource the certificate applies to, what attributes and values are required, which attribute authorities should vouch for them, and what actions are to be granted. It also asks about such details as the length of time for which this certificate should be valid, the scope of the Use-condition (does it just apply to the one resource or to a hierarchy of resources), whether it is a critical Use-condition (it must be satisfied or the user gets no access to the resource even if he satisfies other Use-conditions). The Use-condition certificates must be stored in a directory that is specified in the Policy certificate.

Attribute certificates can also be created by either CertGen or a GUI program *Attribute.sh.* Attribute certificates are actually independent of a particular resource, but the GUI program will look at the Resource Definitions File associated with a particular resource to get a list of attribute names. Resource Policy certificates, and Use-condition certificates may specify where the Attribute certificates should be stored.

Once a set of Policy, Use-condition and Attribute certificates have been stored, the stakeholder can use a Web based interface to see what access they provide. The Resource Definition Server will execute the required CGI script.

Checking access

The Akenti authorization service can be called in several ways: It can be invoked as a function call by a gatekeeper program and thus run as part of the gatekeeper. It can be contacted as a server through an insecure protocol such a TCP. If the akenti server is running on the gatekeeper host, it can return the rights as a simple string. If it is running on another host, it can return a signed certificate. The gatekeeper process must have a copy of the Akenti servers's public key and verify the certificate, before it can trust the information. Or the Akenti server can be contacted as a server through a secure protocol such as SSL and the protocol will do the authentication of the Akenti server and encrypt the returned access string. Akenti returns an authorization answer in one of two ways: a list of strings representing unconditional actions; or a signed capability certificate which may include both conditional and unconditional rights. Conditional rights are rights that may have some conditions attached that only the gatekeeper can evaluate, such as current machine load, disk availability or the state of some related systems.

As has been mentioned previously, the Akenti policy engine finds all the Use-conditions by searching in the URLs specified in the Policy certificates and verifying the issuer and signature on each certificate. If a Use-condition certificate cannot be found for each stakeholder group, access to the resource is denied. Attribute certificates are searched by following URLs in either the Policy certificates and/or Use-conditions. Again, the issuer and signature of each certificate is verified. This signature verification requires that the Akenti policy engine be able to find the X.509 certificates for each issuer. If the CAs who issue certificates publish them in an LDAP server, Akenti will look there. Otherwise there must be some setup actions taken to put all the expected certificate issuers' X.509 certificates in a file system or a web browser where they can be found.

Mod_akenti module for Apache web server

Web-controlled sets of documents and services have rapidly grown from collections of read-only documents that are centrally administered to a vast array of remotely managed documents and services. In the scientific community such Web based systems have become known as portals, and are increasingly used to

provide a common interface to static documents, to allow shared authoring of documents, to allow access to legacy data bases, to allow execution of codes on shared server machines, and practically anything else an inventive scientist can think of. Authorization to perform such access is usually implemented by the http *Basic Authentication* mechanisms, (e.g. user/password or domain based) or by ad-hoc scripts based on the username. These passwords are passed across the internet in clear text and are thus deemed insecure.

In order to make Akenti authorization available for the widest range of distributed resources, we wanted to make it available to Web-accessed resources. There were several ways to accomplish this: referencing resources through CGI scripts that called Akenti, referencing resources through Java servlets or JSPs that called Akenti, or building Akenti authorization into a Web server. The first two methods, involve an indirection between the request and response which is both less efficient and requires more complicated URLs to refer to documents. Since the Apache Web server makes it straightforward to include new functionality, we decided to build a Akenti module for Apache.

The Apache [2] web server is a widely-used, high-performance freeware server which is built around an API [30] which allows third-party programmers to add new server functionality. Indeed, most of the server's visible features (logging, authentication, access control, CGI, and so forth) are implemented as one of several modules, using the same extension API available to third parties. The modules can be statically or dynamically linked to the server. [33]

How apache modules work

Apache divides the handling of requests into different phases:

- URI to file name translation
- Authentication and access checking
- Determining the MIME type of the requested entity
- Returning data to the client
- Logging the request

Each module can contribute to any of these phases. For each phase, a module can completely replace an existing module or can be added to a list of existing modules. The list of modules acts as a queue in which control is passed from one module to another. Each module can return one of three values: OK, DECLINE and FORBID. If a module returns OK, then the server passes the request on to other modules in the queue. A module returns DECLINE when it wishes to ignore a specific request. A FORBID return causes the server to forbid access to the resource requested. The FORBID return veto's other modules replies. Each module can declare a set of handlers to handle specific types of URI requests. The interface between the server core and the extension modules is through a module structure which consists of vector of callback routines. A module provides a callback for each phase that it wishes to handle and NULL for the rest. The module structure for Apache 1.3.x provides the option of defining one or more of the following callback routines.

module MODULE_NAME = {

STANDARD_MODULE_STUFF,
<module initializer routine>,
<per-directory config creator routine>,
<merge routine for directory config>,
<server config creator routine>,
<server config merge routine>,
<command table for defining directives>,
<list of handlers to handle specific requests>,
<filename-to-URI translation routine>,
<check/validate user_id routine>,
*<check user_id is valid *here* routine>,*
<check access routine>,
<MIME type checker/setter routine>,
<module specific fixup of header fields routine>,
<module specific logging activities routine>,
<header parser routine>,
<process initializerroutine>,
< process exit/cleanup routine>,
<post read_request handling routine>
};

Apache allows each module to read directives from the configuration file by specifying a command table structure. The entries in the command table include the name of the command, a pointer to the command handler, an argument which is passed to the command handler, items which tell the server core code where the command may appear (RSRC_CONF), what sort of arguments it takes (TAKE2 means two string arguments), and a description of what arguments should be supplied, in case of syntax errors.

There are three major classes of directives that can be defined in Apache. First Global directives which can occur inside server config files but must be outside virtual host sections. The second class is per-server directives which occur within the context of server config and the virtual host sections. The third class is the per-directory directives which can pretty much occur anywhere

(server config, virtual host, directory,.htaccess). These three classes are subsets of each other.

How mod_akenti works

Mod_akenti is an Apache module that provides Akenti authorization capabilities for the Apache web server. Mod_akenti is implemented as a Dynamic Shared Object module which can be loaded into the server at start-up or restart time. It currently works in Apache 1.3.x. Mod_akenti does not define any handlers as it serves as an access control mechanism for all requests to the web server unless otherwise specified.

Mod_akenti defines two global directives inside the server configuration, and defines a check access call-back. So its interface consists of a call for per-directory configuration, a command table, and a callback for the check access routine.

The two Akenti directives are: AkentiConf, which supplies the name of the configuration file used to configure Akenti policy engine; and AkentiResources, which is used to specify what part of the document tree should be controlled by Akenti. The second directive is of interest as it allows other authorization mechanisms to coexist with that of mod_akenti. It accepts a set of resource names to be controlled, or 'ALL' to control the whole hierarchy or an empty argument to control none of the resources.

Configuration and installation

Mod_akenti is a C++ module, while the core Apache server is written in C. Hence the shared object standard C++ library (ex. libstdc++.so) must be linked at server start-up. This is done through the LoadFile command in httpd.conf. The other shared object libraries can be either in LD_LIBRARY_PATH or defined in the httpd.conf similar to standard c++ library. The Akenti module requires a secure Apache web server (Apache + mod_ssl., which in turn requires that the server be built with the Extended API), the OpenSSL libraries (an open source toolkit that implements SSL and TLS as well as general cryptography), the OpenLDAP libraries (open source library for LDAP suite of applications) and the Akenti suite of libraries. A special program apxs (APache eXtenSion) is used to insert mod_akenti into the web server before start-up. The mod_akenti distribution package [23] provides detailed information about how to build and configure the Akenti module.

Web authentication and authorization methods

Standard Web authentication and access control is based either on the domain in which the request originated, or something called *Basic Authentication* [15] where the user provides a user name and password which the Web browser matches against user information stored on the server machine. There are many authentication modules for Apache based on this mechanism [3]. Mod_auth is the basic module that matches a user and password with an entry in Web specific password and group files. Modules such as mod_auth_dbm and mod_auth_db provide greater scalability by looking up users in a data base. There are also modules available for authenticating users in ldap directories, Oracle, and msql data-bases, and Kerberos users. In all of these schemes the user name and password is passed over the network in plain text. There is one other form of user authentication which is not supported by many browsers called *Digest Authentication* which is implemented by mod_auth_digest. This protocol has the server send a nonce to the browser who then returns an MD5 hash of the nonce, the user name, password, http request and the URI. Thus the password is not sent in the clear.

Mod_ssl [21] which uses X.509 certificates to create encrypted channels between the browser and the server adds a whole new dimension to authentication and authorization. In the typical commercial use of SSL only the server is required to have an identity certificate and private key. This key is used to establish encrypted communication between the browser and server over which passwords can be passed securely. However, SSL can run in a mode that requires the browser to have a certificate and private key for the client. When this mode is used mod_ssl can provide access control based on the client certificate.

The mod_ssl directive SSLVerifiyClient can hold one of the three possible values: none, optional and require. If it is set to require, the browser must provide a certificate that identifies the user. If it is set to optional, the browser will look for a user certificate, but if none exists will attempt the access anyway. If it is set to none, no user certificate is sent.

Once mod_ssl has a client certificate, it provides several more types of access control. It can implement a *FakeBasicAuth* option where it uses the subject of the client's X.509 certificate as a user name, but no password needs to be obtained from the user. It also provides a directive called SSLRequire (see Figure 3.) which specifies constraints which need to be fulfilled in order to allow access. The requirement specification is an arbitrarily

complex boolean expression containing any number of access checks. The variables used in the expression include all the standard CGI/1.0 and Apache variables, plus a large number of variables defined by mod_ssl that refer to parts of both the server and client certificates: e.g. client subject's DN, the client issuer's DN and most components of the client's certificate. The syntax also allows an expression to be used from an arbitrary file. This method is used to match portions of distinguished name compared to the FakeBasicAuth where the whole DN is used.

While the SSLRequire directive is very powerful its main limitation is that the constraints are specified as part of server's configuration file. If many resources need to be controlled, the server configuration will expand to the point where it becomes difficult to manage. In distributed environments where policies for resource access are managed by multiple owners, a centralized access control list does not scale well. For example, WebDAV [16] has been implemented as Apache module, mod_dav, which allows extensions to HTTP protocol in order to provide a shared file system.

If several projects need to be managed by one server, there should be a a way to limit the writing of access policy for a set of resources to the project manager. But since all the policy is in one file, this is not possible.

Mod_akenti, on the other hand, stores all of its policy information outside of the Web server configuration file. The only information in the configuration file is the name of the resources which mod_akenti wishes to control and a pointer to Akenti's own configuration file. The Akenti configuration file points to where the root Policy certificate for each resource tree is. Akenti policy defines who the resource owners are and allows resource owners to express use-conditions on each resource. The use-conditions are signed and stored in a distributed fashion at the owner's convenience. The variables used in the use-conditions are defined by the stakeholder, rather than the Web server. Thus the same access policy can be used for resource referenced via the Web or by another remote method. At run-time Akenti collects all the use-conditions applicable for a certain resource in order to make access decisions. Akenti caches certificates in order to reduce search time. It also caches a Capability which has the access rights of a user for a resource, so that subsequent requests for the same resource require no decision making.

```
<Directory /foo>
  SSLRequireSSL
  SSLRequire %{SSL_CLIENT_S_DN_O} eq
              "LBNL" and
        %{SSL_CLIENT_S_DN_OU}
             in {"DSD", "ICSD", "NERSC"}
```

Figure 3 Example of SSLRequire

mod_akenti could also be used to provide access control for mod_dav which currently uses basic authorization provided by Apache. In this case, the use-conditions have to be specified for WebDAV methods (MOVE, COPY, PROPFIND, DELETE etc.). In addition, a few additional directives are required for mod_akenti inside the per-directory configuration.

Related Work

Policy representations

While there has been a great deal of work in formulating use requirements and standards for authorization protocols or data structures, no single standard has emerged. There is an IETF proposed Attribute Certificate profile [12] to carry attributes associated with an X.509 identity certificate. While the contents and purpose of this certificate are basically the same as an Akenti Attribute certificate, we chose not to use it in our implementation because it is difficult for users and applications to deal with ASN.1 structures. A major goal of Akenti was to make the policy as easy to understand as possible, so using ASCII files to represent policy and principals names consisting of a CA's DN and the user's DN was preferable to using a an ASN.1 structure that identified the holder as a CA and serial number. To understand the meaning of such a certificate, requires a program to print the contents in a readable form, and the ability to find the holder's X.509 certificate and extract the subject name.

KeyNote [5] is a trust management system, which provides a simple language for describing and implementing security policies, trust relationships, and digitally-signed credentials. The KeyNote defines *principal* as any convenient string which may include a cryptographic public key. Authorization policy is contained in *assertions* which consist of a sequence of fields. Each field is represented by a keyword and value. A *credential* asserts some attribute about a principal and is signed by a trusted authority. Both assertions and credentials are represented by the same keyword policy language. Akenti and KeyNote both provide a function call API for compliance-checking for a resource gatekeeper to call when making an access decision. Both systems return list of trusted actions. KeyNote is less tied to one form of authentication than Akenti. A KeyNote principal may be represented by a cryptographic key, or it may

just be an opaque string. They deliberately did not require X.509 certificates in order to separate the issues of secure naming and authorization. While this removes the need for maintaining a PKI, it means that the principals named in the authorization policy may be opaque making it harder for a stakeholder to read and evaluate the policy of a resource.

The mechanisms for creating and storing policy assertions and storing and marshaling certificates are left up to the installer of a KeyNote system. In contrast, one of the emphases of the Akenti system is to support remote creation and storage of policy certificates. It thus provides several tools to help in their creation and signing, while the policy engine supports gathering certificates from file systems, LDAP servers or Web servers. Other systems rely on the user being able to edit policy files on the resource gateway machine which does not meet our goal of accommodating distributed stakeholders.

In our original implementation of Akenti, we chose a simple keyword language for our certificates similar to that used by KeyNote. Eventually, expressing the constraints and trust relationships for all the attributes became increasingly awkward, with too much information being implicit in the ordering of fields or in relationships between fields. For our second implementation we switched to XML for greater flexibility and more precise definition of the semantics. We were also encouraged by the availability of XML parsing tools in a variety of languages and have made use of the Xerces parsers from the Apache XML Project [4]

A recent XML standard specification for security assertions named Security Assertion Markup Language (SAML) [17] has been published by the OASIS [29] consortium. This standard defines both XML protocols and assertion structures. Assertions come in three types: Authentication: the specified subject was authenticated by a particular means at a particular time; Authorization Decision: a request to allow the specified subject to access the specified resource has been granted or denied; Attribute: the specified subject is associated with the supplied attributes. Since Akenti is only supporting X.509 authentication, it does not need a general purpose Authentication structure. It just uses the X.509 certificate (or chain of certificates if delegation is involved) and assumes that the resource gateway has authenticated the certificate. Akenti will check for revocation, since the current implementations of SSL do not do this. The capability certificate returned by the Akenti server differs from the Authorization Decision assertion in that it does not contain the reasons (evidence) of why it made the decision, but may contain unresolved conditions on the actions, so that the gatekeeper can do further checks. Again the attribute assertion/certificate covers has the same purpose as the PKIX Attribute Certificate and the Akenti Attribute certificate: namely, a subject name, an associated attribute-value pair and the authority that attests to this. The SAML standards seem to be focused on letting various peers report security decisions. The focus in Akenti, is more on gathering and interpreting of policy (Use-condition) statements about the resource. The only real communication is the authorization request and reply between the resource gateway and the Akenti server.

Authorization models

The authorization model used by KeyNote is essentially the same as Akenti uses. A principal makes a request to a resource gateway, handing it an identity credential that can be authenticated. In Akenti this is normally just an X.509 certificate, while KeyNote supports other types of credentials. Then the gateway server makes an authorization request to the authorizer, e.g. Akenti or KeyNote. The current implementation of KeyNote only supports function calls, where Akenti will support function or server calls. The authorizer returns a list of allowed actions to the gate keeper for its interpretation or in the case of Akenti being called as a server, it returns a capability certificate signed by Akenti.

Shibboleth [11] is a cross-institutional authentication and authorization service for access control to Web-accessed resources. It is being specified by the InterNet2 middleware architecture committee. It has many of the standard goals of distributed authorization with one additional twist. It wants to be able to grant access to a user who can still maintain anonymity at the resource site. The major motivation for this goal is access to library materials by academics. Their authorization model entails a user making a request to a web server and providing a identity handle back to his home institution. The Web server then asks that institution for attributes about the user. It then checks those attributes against its local policy to allow or deny access. The user need only authenticate to his host site and may use whatever type of credentials that site recognizes. One difference between this trust model and that used by Akenti, is that in Akenti, the resource provider specifies a limited number of trusted authorities that it will accept for authenticating users and attributes. In the Shibboleth case, each member institute must trust all the sites at which any of its user's reside. So for a user to get access to a remote resource, its whole site must be trusted.

While in a more traditional PKI environment, a user only needs to get a credential for himself from an authority that the resource site trusts.

The Community Authorization Server (CAS) [28] is a new authorization service being developed by the Globus Project [13] for Grid environments. Their authorization model allows a resource site to grant a community access to resources and the authorization server for that community to grant access to the community members. This is implemented by having the user go to the CAS server and get a delegated proxy certificate [31] with the CAS server's identity, which includes a rights restriction extension that limits what resources can be accessed. The resource gatekeeper must interpret the restricted rights extension and verify that the community has such rights to the resource. Since the delegated proxy is a short-lived X.509 identity certificate it gets passed between the user and the resource gateway as part of the SSL connection. There is no additional information that needs to conveyed, as is the case when a user needs to hand attribute certificates to the gatekeeper. CAS differs from Akenti in that the examination of policy and granting of rights is done before the gatekeeper is contacted. This means the user must ask for all the rights he will need in advance of referencing the resource. In Akenti, all the gathering and checking of policy is done after the call to the gatekeeper to perform a certain action. Akenti does cache the rights that the user was granted, to deal with the common case of several calls in rapid succession for resources in the same realm.

Policy about resources is stored and managed by the CAS servers and so far mainly consists of lists of objects and allowed rights. This information is included in the rights restriction extension of the delegated proxy. The intent of the CAS project is to extend the policy language as the need arises. The CAS administrator is responsible for adding each community member to the appropriate groups. The CAS administrator may also delegate administration of subsets of the objects to additional people. In contrast, in Akenti, a new user would need to contact the stakeholder for the resource to be added to the policy files.

Conclusions

Akenti is an authorization service that uses authenticated X.509 identity certificates and distributed digitally signed authorization policy certificates to make access decisions about distributed resources. It supports authorization decisions based either on policy that is gathered by the resource gatekeeper, or on a rights-granting capability presented by the user. It supports Globus proxy identity certificates, and could be easily extended to handle restricted delegation credentials. We have implemented an Apache Web server module which allows the same authorization policy to be used to control access to Web accessed resources as well as resources accessed by other remote methods. Thus all the resources in a portal can use the same authorization mechanism.

Akenti differs from most of the other work that we have surveyed in the emphasis on using easily read policy statements that are independently created and signed by multiple stakeholders. This policy can be stored on the resource host or locally to the stakeholder and be gathered and evaluated by the trusted authorization server at the time of resource access. The Akenti distribution also includes several tools for displaying the combined authorization policy for a given resource and for tracking the steps in a user's authorization or rejection.

It has been used as part of the Diesel Combustion Collaboratory [26] to control access to Web-based documents and remote execution and is now being integrated with the Globus job manager to control access to legacy applications in the National Fusion Grid [19].

The code is freely available as C++ source code, or Linux and Solaris executables. (http://www-itg.lbl.gov/Akenti)

Acknowledgments

The original idea for Akenti came from William Johnston. Case Larsen did a large part of the original implementation. Maria Kulick, Guillaume Farret and Xiang Sun have also contributed to the current implementation.

Appendix A: XML definition for the Akenti policy language

```
<?xml version="1.0" encoding="US-ASCII"?>
<!-- This DTD is intended to define all the Akenti Policy elements:
   Policy Certificates, UseCondition Certificates, Attribute Certificates,
   Capability/Authorization Certificates, and Cache Certificates
 -->
<!-- Note: one or more (+), zero or more (*), or zero or one times (?)-->

<!ELEMENT AkentiCertificate (SignablePart, Signature)>

<!ELEMENT SignablePart ( Header, (PolicyCert |  UseConditionCert | AttributeCert | CapabilityCert ) )>

<!ELEMENT Header ( Version, ID, Issuer, ValidityPeriod) >
 <!ATTLIST Header
        Type  (attributeCertificate | cacheCertificate |capabilityCertificate | policyCertificate | useCondCertificate ) #REQUIRED
        SignatureDigestAlg  (RSA-MD5 | RSA-SHA1 | DSA-MD5 ) #REQUIRED
        CanonAlg  (AkentiV1) #REQUIRED >


<!ELEMENT PolicyCert  ( ResourceName, CAInfo*, UseCondIssuerGroup+, AttrDirs*, CacheTime )>
<--
        ResourceName    Name of the resouce to which this policy applies
        CAInfo          The DN and X509 identity certificates of all the CAs we will trust.
                                May include pointers  places where it publishes CRL's and identity certificates
        UseCondIssuerGroups Stakeholders and their Certificate directories
                                At least one UseCondCert must be found from
                                each group.
        AttrDirs        optional list of URLs in which to search for Attribute certificates
        CacheTime       Maximum time in seconds that certificates relevant   to this resource may be cached
-->
<!ELEMENT UseConditionCert ( ResourceName, Condition, Rights, SubjectCA*)>
<!--
        ResourceName    name of the resource to which the useCondition applies
        Condition       A boolean expression stating what attributes a user needs to satisfy the UseCondition and what users
                        and CAs are trusted to attest to specified attribute
        Rights          An opaque list of actions known to the stakeholder and the resource gateway
 -->
   <!ATTLIST UseConditionCert
        enable (true | false) #REQUIRED >
<!--
        scope   if sub-tree the UseCondCertificate applies to all the resources that are in the sub-tree named by the resource
                if local, it applies just to the one resource named
        enable  if true, this UseCondition must be satisfied by anyone wanting to use the resource, if false it need not be satisfied
                if a user satisfies other UseConditions.
-->
<!ELEMENT AttributeCert ( SubjectAndCA, AttrName, AttrValue, Condition*)>
<!--
        SubjectAndCA    Subject to which this attribute applies
        AttrName        name of attribute
        AttrValue       value of attribute
        Condition       An optional Constraint that is placed on how or when the attribute should apply
-->
<!ELEMENT CapabilityCert ( ResourceName, SubjectAndCA, Actions*, ConditionalActions*)>
<!--
        ResourceName    name of the resource to which the rights apply
        SubjectAndCA    user who has the rights
        UnConditionalActionsthe actions that have been authorized
        ConditionalActions  actions that still have some unevaluated constraints.
  -->
<!ELEMENT ConditionalActions ( Condition, Actions )>
  <!ATTLIST ConditionalActions
        critical (true | false) #REQUIRED   >
<!--
        Condition       Constraint that is placed on how or when the attribute should apply
```

```
        Actions         The access rights that are allowed if the condition  is true
        Critical        If this is false, the Condition must evaluate to  true, or even the UnConditionalActions do not apply
-->

<!ELEMENT CAInfo (CADN, X509Certificate+, IdDirs*, CRLDirs*)>
<!--
        CADN            the distinguished name of the CA
        X509Certificate A chain of the X509 identity certificates of the CA,  includes its public key.
        IdDirs          an optional list of directories in which the CA stores the certs it issues
        CRLDirs     a list of 0 or more URLs to directory services  in which to search for certificate revocation lists
-->

<!ELEMENT Condition ( Constraint, AttributeInfo+)>
<!--   A Condition contains a boolean expression stating what attributes a user needs to satisfy the UseCondition and

        what users and CA  are trusted to attest to what attribute/value pairs.
-->
<!ELEMENT CRLDirs (URL+)><!-- list of 0 or more URLs to directory services   in which to search for certificate revocation lists-->
<!ELEMENT AttrDirs (URL+)> <!-- AttrDirs list of 0 or more URLs to directory services in which to search for attribute certificates.-->
<!ELEMENT IdDirs (URL+)>    <!-- list of 0 or more URLs to a directory services  in which to search for identity certificates.-->
<!ELEMENT UseCondIssuerGroup (Principal+,URL+)>  <!-- group of stakeholder and their certificate directories. -->
<!ELEMENT AttributeInfo (AttrName, AttrValue, (CADN | Principal), AttrDirs*, ExtArgs*) >
  <!ATTLIST AttributeInfo type (STANDARD | X509 | AKENTI | EXT_AUTH ) #REQUIRED>
<!--
        STANDARD        attributes if they are evaluated by some system call
        X509             attributes if they are part of an X509 Identity certificate,  e.g. O, OU, CN;
        AKENTI           attributes if there is an Attribute certificate to attest to a user's possession of the attribute
        EXT_AUTH        if some external authority is called to evaluate them

        AttrName        name of attribute used in constraint
        AttrValue       name of value required by constraint
        CADN            name of CA that issues the identity certificate that contains the x509 attribute we need.
        Principal       the name of the attribute issuer and CA for Akenti attr
                        or the name of an external authority that can evaluate  an attribute
        AttrDirs        an optional list of directories in which to search  for Attribute Certificates
        ExtArgs         optional  list of arguments that may be handed to an external authority.
-->

<!ELEMENT ValidityPeriod EMPTY><!-- Beginning and End date in UCTime of when the certificate is valid -->
  <!ATTLIST ValidityPeriod
    start CDATA #REQUIRED
    end CDATA #REQUIRED
  >
<!ELEMENT ExtArgs (String+)>
<!ELEMENT ID EMPTY>  <!--A unique ID assigned to every certificate when it is created   -->
        <!ATTLIST ID id CDATA #REQUIRED   >
<!ELEMENT Version EMPTY>  <!--  Certificate format version -->
        <!ATTLIST Version ver CDATA #REQUIRED    >
<!ELEMENT Issuer (UserDN,CADN,URL* )>
<!ELEMENT Principal (UserDN,CADN )>
<!ELEMENT SubjectAndCA (UserDN,CADN)>
<!ELEMENT URL (#PCDATA)>  <!-- protocol, host, port and file name -->
<!ELEMENT CADN (#PCDATA)>
<!ELEMENT SubjectCA (#PCDATA)>
<!ELEMENT X509Certificate (#PCDATA)>
<!ELEMENT UserDN (#PCDATA)>
<!ELEMENT ResourceName (#PCDATA)>
```

References

1. D. A.Agarwal, S. R. Sachs, W.E.Johnston *The Reality of Collaboratories* Computer Physics Communications, 1998, vol. 110, p. 134-141
2. Apache Software Foundation http://www.apache.org
3. Apache Module Registry, http://modules.apache.org/
4. Apache XML Project; http://xml.apache.org/
5. M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis. *The KeyNote Trust Management System, Version 2. RFC-2704.* IETF,September 1999. http://www.crypto.com/papers/rfc2704.txt
6. N. Damianou, N. Dulay, E. Lupu, M Sloman,: *The Ponder Specification Language Workshop on Policies for Distributed Systems and Networks* (Policy2001), HP Labs Bristol, 29-31 Jan 2001
7. T. Dierks, C. Allen, *The TLS Protocol, Version 1* IETF RFC 2246; http://www.ietf.org/rfc/rfc2246.txt
8. Diesel Combustion Collaboratory (DCC), http://www-collab.ca.sandia.gov/dcc/
9. DisCom^2, http://www.llnl.gov/asci/discom/
10. C. Ellison *SPKI Requirements,* IETF RFC 2692 1999, http://www.ietf.org/rfc/rfc2692.txt
11. M. Erdos, S. Cantor, *Shibboleth-Architecture DRAFT v04*, http://middleware.internet2.edu/shibboleth/docs/raft-internet2-shibboleth-architecture-04.pdf
12. S. Farrell, R. Housley, *An Internet Attribute Certificate Profile for Authorization,* <draft-ietf-pkix-ac509prof-09.txt>, June, 2001 http://www.ietf.org/internet-drafts/draft-ietf-pkix-ac509prof-09.txt
13. I. Foster, C. Kesselman, eds. *The Grid: Blueprint for a New Computing Infrastructure,* 1999, Morgan Kaufmann
14. I. Foster, C. Kesselman, S. Tuecke, T*he Anatomy of the Grid: Enabling Scalable Virtual Organizations.* International J. Supercomputer Applications, 15(3), 2001. http://www.globus.org/
15. J. Franks, et.al. *HTTP Authentication: Basic and Digest Access Authentication*, IETF RFC 2617, http://www1.ics.uci.edu/pub/ietf/http/rfc2617.txt
16. Y. Goland, et al., *HTTP Extensions for Distributed Authoring* -- WEBDAV, IETF RFC2518 http://www.ietf.org/rfc/rfc2518.txt
17. P. Hallam-Baker, E. Maler, eds. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML),draft-sat-core-25, http://www.oasis-open.org/committees/security/docs
18. R. Housley, W. Polk, W. Ford, D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile* <draft-ietf-pkix-new-part1-12.txt>, http://www.ietf.org/internet-drafts/draft-ietf-pkix-new-part1-12.txt
19. K. Keahey, et al., *Computational Grids in Action: The National Fusion Collaboratory*, submitted to Future Generation Computer System, 2001., http://www.fusiongrid.org
20. Launch Pad, Portal to the IPG, http://www.ipg.nasa.gov/launchpad/servlet/launchpad
21. modssl, http://www.modssl.org/
22. J. Myers, *Simple Authentication and Security Layer (SASL)*, IETF RFC 2222, 1997, http://www.ietf.org/rfc/rfc2222.txt
23. S. Mudumbai, *mod_akenti: Akenti Access Control module for Apache* http://www-itg.lbl.gov/Akenti/docs/mod_akenti.html
24. NASA's Information Power Grid, http://www.ipg.nasa.gov/
25. National Fusion Grid, http://www.fusiongrid.org/
26. C. Pancerella, L. Rahn, C. Yang, *The Diesel Combustion Collaboratory: Combustion Researchers Collaborating over the Internet,* Proceedings of ACM/IEEE SC99 Conference, November 13-19, 1999. Portland, Oregon, USA, http://www-collab.ca.sandia.gov/dcc/
27. Particle Physics Data Grid (PPDG), http://www.ppdg.net/
28. L Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. *A Community Authorization Service for Group Collaboration.* Submitted to IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2001, http://www.globus.org/research/papers.html#CAS-2002.
29. Oasis, www.oasis-open.org
30. R. Thau , *Apache API notes*, http://modules.apache.org/doc/API.html
31. S. Tuecke, et al., *Internet X.509 Public Key Infrastructure Proxy Certificate Profile* , IETF draft, http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-01.txt
32. M. Thompson, et.al., *Certificate-based Access Control for Widely Distributed Resources*, Proceedings of the Eighth Usenix Security Symposium, Aug. '99
33. Wainwright P. *Professional Apache,*Wrox 2001, http://www.apache.org/

Invited Talks and Experience Reports

Improvements on Conventional PKI Wisdom

Carl Ellison
carl.m.ellison@intel.com
Intel Labs

Abstract: *This paper contrasts the use of an ID PKI (Public Key Infrastructure) with the use of delegatable, direct authorization. It first addresses some commonly held beliefs about an ID PKI – that you need a good ID certificate to use digital signatures, that the ID certificate should come from a CA that has especially good private key security, that use of the ID certificate allows you to know with whom you're transacting and that the combination gives you non-repudiation. It then identifies flaws in those assumptions and addresses, instead, the process of achieving access control – either through an ACL plus ID, or directly. It then applies each method of achieving access control to two examples – one within a large company and one between companies.*

[This paper is an expanded transcript of the invited talk of the same title prepared for the Internet-2 1st Annual PKI Workshop, which was held at NIST at the end of April 2002.]

1 Introduction

The thesis of this paper is that the PKI community has accepted a number of concepts, listed here as "Conventional PKI Wisdom" that actually get in the way of achieving security. Some of them are false premises. Some of them are not achievable. None of them is necessary to achieve actual security. Instead, it advocates paying attention to the problem of access control and especially the determination of authorization. Authorization usually requires the same level of effort as ID certification. It can be used alongside ID certification, incurring extra load and expense, or it can be used instead of ID certification.

2 History

The concepts at issue here date back to the introduction of public key cryptography by Diffie and Hellman.

In their 1976 paper, "New Directions in Cryptography" [2], Diffie and Hellman postulated that the key management problem is solved, given public key technology, by the publication of a modified telephone directory, which they called the Public File. Instead of a name, address and phone number, the Public File would contain a name, address and public key. When you want to send me a message for my eyes only, you turn to the Public File, find my entry and use the public key associated with that entry to encrypt a message for me. Only I can decrypt that message, since presumably only I have the associated private key. Because of the nature of public key cryptography, there is no need to keep the public key secret, although one must still protect that Public File from tampering.

As a demonstration of the power of public key cryptography, this was a brilliant example. The problem is that there are people who took this example literally and set about creating such a directory, when as I point out here, there is an inherent flaw in this construction. Namely, you cannot find me in that directory. Diffie and Hellman solved the previously difficult key management problem by use of names, but did not offer any solution to the even more difficult name management problem.

In his 1978 MIT Bachelor's thesis [5], Loren Kohnfelder addressed the Public File proposed by Diffie and Hellman, noting that a networked version of this directory would have a performance problem. He proposed instead that each line item of that directory, which he identified as name (presumably login name) and public key, be digitally signed and distributed to anyone who wanted a copy, for them to hold. He coined the name **certificate** for this digitally signed directory line item. This may have avoided the problem of loss of access to the central Public File (e.g., because of network partition), but in fact it made the name management problem worse. On the other hand, no one was especially aware of that problem, so solving it was not part of Kohnfelder's requirement set.

In the 1980's, the X.500 effort set about building a directory like that envisioned by Diffie and Hellman, as a single directory to cover the world's devices and people. For authentication (e.g., to provide notation of the permission to modify an entry in the directory), that standards effort specified the X.509 certificate format, binding a public key to a **Distinguished Name (DN)**, which can be thought of as a pathname into the X.500 directory. For our purposes, it is an identifier that is intended to refer uniquely to the person who holds the key to which the X.509 certificate binds it.

Around 1990, the Privacy Enhanced Mail (PEM) effort in IETF chose to use X.509 certificates to identify mail recipients. There was a fair amount of excitement at the time over the potential of X.500 to make sense of what was already a bewildering set of people connected by the various networks (now just called "the Internet", but still quite small at that time, before AOL

experienced its user explosion). However, PEM failed because X.509 failed. Not only were there no Certificate Authorities (CAs) in place to issue X.509 certificates, the very process of choosing a DN and generating an X.509 certificate appeared to have legal connotations that at least the company where I worked at the time was not willing to accept.

To get around this failure of X.509, there was a version of PEM produced, called RIPEM that did not use X.509. It allowed the use of keys that were delivered out of band and used without certification. To provide for certification without CAs, in 1991, PGP allowed for any keyholder to sign the key of any other keyholder, under the **Web Of Trust** assumption: that multiple independent signatures on a certificate would be as trustworthy as one highly trusted signature on that same certificate, when you had exceeded some number of independent signatures, no matter how vulnerable each of those signers might be.

PGP succeeded where PEM failed, but there was still something wrong with the PKI model. In 1996, three independent efforts (SDSI, SPKI and PolicyMaker) departed from the PKI model in the same way: using a public key itself as the identifier of the keyholder, rather than some name. This has the advantage that there is no ID certificate needed to bind that key to the ID of the keyholder since the key *is* the ID.

3 Conventional PKI Wisdom

There has been a great deal written and discussed about PKI, but there are some frequently encountered items of conventional wisdom about PKI that this paper addresses directly:

1. that you need an ID certificate;
2. that you should get that ID certificate from a CA that protects its signing keys well (e.g., uses a vault with strong physical protection against theft or misuse of keys);
3. that with such an ID certificate, you will know with whom you are dealing when you process a signed message or encrypt a message to some key; and
4. that with all of this, you get non-repudiation, which means that the signer cannot later deny having sent a particular signed message when you present that signed message to a judge and ask for it to be considered binding against the human you have cited as the signer.

As it turns out, all four of these items of wisdom are seriously flawed, if not completely false.

3.1 ID Certificates

The original model of an ID certificate was one that would bind me to my entry in the X.500 directory, by way of the DN that both identified me and uniquely specified my entry in the directory. The assumption was that one needed only one such entry (or perhaps two: one at work and one at home).

By contrast, each of us has multiple identities both at home and at work. I, for example, have five different but equally valid IDs at work. They are used for different functions and their format and nature was determined by the applications in which they are used. At home, I have even more. There are 4 credit card numbers, 1 ATM card number, 4 bank account numbers (all from the same bank), ISP account names, etc.

There are two problems with getting one ID certificate:

1. we would have to change all legacy software and business processes to use that one ID or have that one ID certificate list all of my IDs; and
2. we would have to find one CA with the authority to establish all of those ID to key bindings.

We take it as impossible to change all business processes to use one common ID. It is also a potential privacy violation either to use a single ID or to bind all different IDs into one credential, so that some party can know how to link all of my transactions to one another.

More serious is the problem of finding one certificate issuer that has the authority to do all of these ID bindings. My company will accept only itself to bind my key to my employee ID number. My bank will accept only itself to bind my key to my bank account number. The key used could be the same in both certificates, but the binding must be performed by an entity with the authority to perform that binding. That authority comes from business rules and security policy, not from some CA characteristic like strength of protection of the CA's own private keys.

The conclusion is that we cannot have one ID certificate that is used for everything. We will most likely need as many certificates as we have relationships.

3.2 CA Key Security

It is accepted wisdom that certificates should be issued by a Certificate Authority that operates out of a vault – that is, that protects its signing keys very strongly, with military grade physical and personnel security, multi-factor authentication of people, multi-person access controls, etc. Such a facility is extremely expensive, so there cannot be many of them. Let us consider the use

of a CA in four different ways, discussed below, and improve on this design.

3.2.1 Client goes to the Vault

Early theoretical papers on certification assumed that the client would go to the vault, present credentials proving identity along with a public key and receive an ID certificate in return. This is presumably secure, but has the problem that it is too expensive for the user.

Meanwhile, it actually has a security problem, in that there will be very few such vaults, so the people running the vault have no idea who the user is. They will never have met the user and therefore will have to rely on other credentials to establish the identity of the user. This weakens the overall process to something less than the security of the credentials used and opens the process up to traditional identity theft techniques. Since we see identity theft increasing in frequency, it is doubtful that we could call this mechanism secure.

3.2.2 Client Opens a Channel to the Vault

One early attempt to overcome the expense of the previous method was to permit a client to open a communications channel to the vault. This could be by telephone, but more likely it is by web form over an encrypted channel.

Let us assume for the sake of argument that the connection is established and there is no man in the middle. We know that if you have a confidential channel, you can mutually authenticate the parties on the two ends by use of a shared secret. So, it is possible to establish identity over this channel. Once that has been done, the CA in the vault can issue a certificate for the public key provided by the user, and from then on, that key pair and certificate could be used for authentication.

The problem comes with establishing that shared secret.

At least one company considered making a business relationship with a credit bureau and then using the credit bureau's body of knowledge about the user to quiz the user and establish identity. The problem with this mechanism is that there are no secrets shared between the user and the credit bureau. That is because the credit bureau's primary business is the selling of the information it gathers about people. Making matters worse, even if one were to find a repository of information about people that is not in the business of selling that information, if it uses the same information that some other organization makes publicly available, then that information can still not be used as a secret shared with the user.

So, the problem of establishing a good, high entropy, shared secret with the user boils down to something as expensive as the first mechanism. That is, the user can come to the vault, prove identity to trusted employees of the vault, get that identity recorded along with a high entropy secret generated and shared with the user during that visit. That high entropy secret can then be used later, over a web connection, to get a certificate.

3.2.3 Registration Authorities

With the previous mechanism ruled out because it is either grossly insecure or as expensive as the first mechanism, the next step is to reduce the cost for the user by enlisting registration authorities (RAs). For each CA, there would be a large number of RAs, so that any user could find an RA within easy travel distance. The user could then prove identity to that RA. The RA would then instruct the CA, over a mutually authenticated, cryptographically secured channel, to issue the desired certificate from the vault.

This allocates the cost of the first mechanism to the CA infrastructure rather than the user. The CA has come to the user rather than the other way around. This also could have a security advantage. That is, if there are enough RAs, it could be that the user would be known personally by the RA and identity could be established not by paper or plastic credentials but rather in person. This would reduce the threat of standard identity theft.

Although this is far more secure than the previous mechanism and much cheaper for the user than the first mechanism, its security can be better.

3.2.4 CA on the RA Desk

To improve the security of the previous mechanism, the secured network connection between the RA and the CA should be severed and the computer on the Registration Agent's desk should run a CA and directly issue the user's certificate.

This is categorically more secure than the previous design, primarily because the network connection between the RA and the CA has been eliminated, depriving an attacker of one avenue for attack. There is also a security advantage, since the CA in the vault would now not sign individual certificates but rather sign the certificates of the next layer of CAs – those now on the RA desks. Because this is a much lower volume operation, the CA could operate in a different fashion. For example, it might use split-key (distributed signing) technology rather than a single, secured vault. With enough key shares, split-key technology can be arbitrarily secure, far surpassing the security of any vault, even with key shares held in only moderately secure but tamper-evident storage.

Some may argue that this design exposes a valuable key – the final CA private key – to possible theft because the RA computer is not specially protected. However, this could also be a security advantage. If an attacker

can steal the CA key from the computer on the RA desk, then that attacker could just as easily steal the key by which the RA authenticates its connection to the CA, under the previous design. Under that design, the attacker could then get the CA to sign a false certificate and that false certificate would have the imprimatur of having come from the real CA in the real vault. If the theft were discovered, then all signatures by that CA key would be called into question and the CA key itself might need to be revoked, along with all certificates it had generated. Under this last design, if a leaf CA key were stolen, then only that one key need be revoked along with only those certificates it had generated.

3.3 Know the Other Person

The third element of conventional wisdom is that with a proper ID certificate, you can know the person with whom you are transacting. This idea traces back to the 1976 Diffie-Hellman paper [2], which made the assumption that the first important job was to learn the identity of the party on the other end of a communications connection. The Public File and then the ID Certificate were to achieve that by binding the person's name to the person's public signature key.

This assumes that names work as identifiers.

3.3.1 The John Wilson Problem

The fact is that names do not work as identifiers. This has come to be known as the John Wilson problem, named after a co-worker.

3.3.1.1 E-mail

At Intel, there are (at the time of this writing) eight employees with the name John Wilson, in some spelling. The IT department is very careful to make sure that each of these John Wilsons has a unique name. That is because these names are used as e-mail addresses and to index into the corporate employee database.

In spite of the care with which each John Wilson has been given a unique name (e.g., through the use of middle initials), John keeps getting mail intended for one of the other John Wilsons and they keep getting mail intended for him.

3.3.1.2 Airport

This problem isn't limited to e-mail misdirection.

In August of 2001, John was returning from a one-day business trip to the Bay Area. He had an electronic ticket and no luggage. It was a simple trip.

On the return leg, he went to the ticket counter, was asked for an ID (his driver's license) and was asked if anyone unknown to him had given him anything to carry, etc. The ticket agent printed out his boarding pass and gave it to him. He was looking at it as he started to walk away but turned back to the ticket agent to say, "I'm not going on to Eugene. I'm just going to Portland."

The ticket agent took back his boarding pass, consulted the computer, and said that he had the boarding pass for the other John Wilson on that flight. That other John Wilson had his boarding pass.

So, the solution was for John to go to the gate and have them page John Wilson – and then, when the other John Wilson appeared, trade boarding passes.

Especially in light of the post-9/11 requirement to have luggage removed from a flight if the ticketed passenger does not take the flight, this could have been a serious security problem.

3.3.1.3 Ann Harrison

When I tell the John Wilson stories, instead of getting a reaction of disbelief or scorn at my making too much of a case out of an isolated incident, the reaction is almost always "That's nothing. Listen to this."

A friend of a friend, Ann Harrison, reacted that way.

She told of sitting on the examining table in her doctor's office, waiting for the doctor, when the nurse came in, carrying a syringe. The nurse said, "This will only sting a little". Ann asked, in shock, what the nurse was trying to inject her with and the nurse replied that it was Botox (botulism toxin). Ann said that she doesn't get Botox injections, to which the nurse replied, "but you're Ann Harrison, aren't you?"

3.3.1.4 Carl Carlson

In the early 1900's, Carl Carlson was working in a factory in Wisconsin, in a heavily Swedish community, and was getting annoyed that he kept getting paychecks for another Carl Carlson, one who earned less than he did. So, sitting in the bar after work one payday, he decided to change his name to something really unusual and avoid this problem. He looked across the bar and saw a sign with a really unusual name ... and that's how my ex-in-laws ended up with the family name Miller.

3.3.2 Names are not IDs

These anecdotes illustrate a point that should be of concern to us as computer scientists and especially to those of us involved in PKI.

Human beings do not use names the way we want them to.

The actual process by which humans use names and the psychology behind that process deserve a great deal of study. It is clear, even prior to that study, that computer

developers and computer users deal differently with names.

I speculate that computer developers, and especially PKI or large directory developers, think of names the way we do variable names or file path names. That is, a name is some string, unique within its block or directory or context, that unambiguously identifies some object (value, file, person, ...) – and we further assume that the mechanism that uses this name (a compiler, an operating system, or a human user) will follow that unique string to the same object any other mechanism would follow the string to.

Compilers and operating systems may behave this way, but **human users do not**.

Our PKIs assume they do. Our mail agents assume they do. Much of what we design in computer science makes this same, false assumption. For our immediate concern, the main impact is that PKIs are based on a false assumption and the security of systems using those PKIs suffers as a consequence.

In a way, however, this is good news. This means that there are a great many fresh new research opportunities. For example, how would you build a mail agent that does not use names or e-mail addresses for people?

3.3.3 ID as Dossier

It is doubtful that human beings could ever be trained to read all information offered in a certificate and verify it against their knowledge of a person, before jumping to a conclusion about the identified person. Even if that training could be achieved, however, an ID certificate usable by everyone would become a dossier.

Consider an ID cert for John Smith. The name alone doesn't tell you which John Smith, so you need additional information. Andy works with John, so he needs John's employer (and building and mail stop) in the ID certificate. Betty knows John only at home, so she needs his home address in the ID certificate. Charles knew John at work 10 years ago, so he needs John's work address from 10 years ago. Dan shared a hospital room with John back in 1994, so he needs a record of John's hospitalization from then in order to identify John unambiguously. This process needs to be iterated over all possible relying parties, to make sure the ID certificate works for all of them.

The result would be a nearly complete dossier on the keyholder, and that dossier would almost certainly violate privacy laws, not to mention John's desires. As a result, the ID certificate could not be released to the public. That, however, violates the basic purpose of the ID certificate. A workable alternative would be to have different ID certificates for use by different relying parties [6], but that violates the design goal of one ID certificate that lets an arbitrary relying party know with whom she is transacting.

3.4 Non-repudiation

The fourth item of common wisdom has to do with non-repudiation, which is usually defined as the inability of a person later to deny having digitally signed a document.

The central idea behind the concept of non-repudiation is deferred enforcement of security. That is, one receives a digitally signed document (often described as a contract, when non-repudiation is discussed) and verifies the signature on the document and the certificate chain that identifies the key used, and then acts on the document. In most cases there will be no intention of fraud and the transaction proceeds normally. However, in case there was fraud, the document can be produced along with its certificate chain to present to a judge. The judge can verify those signatures and thus establish that this document was signed by the defendant.

There are several problems with this understanding and this process.

3.4.1 Expense

The process described above is expensive. The digital signature and certificates that bind the signer to a document do not bind that signer to a location. The signer must be located and brought to trial. The process of location and the process of trial are both expensive. If the amount of the loss were small enough, taking the case to trial would not pay.

3.4.2 Not Adequate

Assuming non-repudiation was achievable, technically, and a judge found a defendant responsible, this process works only if the victim can be made whole. In cases of moderate financial loss, this might be adequate. However, if the loss were of something more valuable than the perpetrator's total lifetime worth, then the victim cannot be made whole. Worse, if the loss were of a life or of secrets, then no amount of money could compensate the victim.

3.4.3 Not Achievable

The main problem with the theory of non-repudiation is that it is not technically achievable. That is, the intention is to bind a human being to a digitally signed document. With a holographic signature on a paper document, the human's hand came in contact with the paper of the document. With a digital signature there is machinery between the human and the signed document: at least a keyboard, software (to display the

document and to drive the signature process) and a key storage and use facility (e.g., a smart-card).

No one has demonstrated, in the normal computer for home or office use, the prevention of introduction of hostile software. To the contrary, we have seen a steady increase in such incursions over the years.

There are secure facilities for key storage and use, but no mechanism that an average home or small business user would choose to buy has been proved secure.

Meanwhile, computers are not restricted to isolated rooms with card access entry, raised floors, guards outside the glass walls, etc., that they might have been in the 1970's when much of this thinking about public key cryptography had its nascence. Computers are not only everywhere; they are unprotected to a continually increasing degree. Therefore, even if the computer has no hostile software and its private key is kept in a truly secure facility, access to the keyboard of that computer is not limited to the person certified to be associated with that private key.

What might make this process of non-repudiation work would be hardware that would serve as a witness to a signature, providing tamper-proof evidence of the actions of a human being (e.g., through videotape), of what that human was reading and of the human's positive action to assent to the displayed document. Such a log of human behavior could then be presented in court to prove the claim of non-repudiation.

Of course, if such hardware were available, then we would not need digital signatures, much less the assumption of non-repudiation on digital signatures.

3.4.4 Contractual Commitment

For lack of technical achievability, some people try to legislate non-repudiation. If laws are written to presume that the certified keyholder is responsible for anything done by that key, then the rational thing for a computer owner to do is to refuse to accept ownership and use of that private key. That could bring not just PKI but use of public keys to a screeching halt.

The good news in this is that we do not need non-repudiation in order to do business with digital signatures. If two parties want to do electronic business with each other, they can sign a paper contract with one another in which party A might declare that it would honor any document digitally signed and verified with a public key that is given in the contract (or whose cryptographic hash is printed in the contract). The party accepting that responsibility for that key could then protect that key with mechanisms appropriate to the way that key was empowered. If one is ordering office supplies with that key, then maybe it is kept encrypted by password on the hard drive of a PC on a secretary's desk. If one is ordering millions of dollars worth of industrial supplies, then the key might be kept in a locked room, under 24x7 guard, with multi-factor authentication for people entering the room, special computers with strong key storage facilities that erase their keys if the mechanism is physically moved, no network connections for the computers and strict control over the software that is allowed to be loaded onto the computers.

4 New PKI Wisdom

The reasoning above gives us a new list of PKI Wisdom:

1. There is and will be no single ID, so a single ID certificate makes no sense.
2. Discard RAs and put CAs on the RA desks.
3. Knowing a keyholder's certified name does not tell you who that keyholder is.
4. Non-repudiation is neither adequate for serious problems nor achievable.

So, instead, we need to do strong access control and that requires more than ID certification. There are several ways to achieve access control, as outlined below.

5 Certificate :: DB Trade-off

As we consider the various ways to do access control, we must address the religious battle between those who advocate certificates and those who advocate servers. Each technology can achieve the same results, under certain assumptions. The main difference is in their behavior under network load or partition, but there are security differences, discussed later in this paper, having to do with database administration.

For example, Kohnfelder created certificates by digitally signing a line item from a protected database: the Public File. This has the advantage of making verifiable data available even when the database is not, whether by network partition or by mere performance problem.

This process can be applied with any kind of database. In particular, it applies to all three edges of the credential triangle shown in Figure 1.

5.1 CAP Principle

Fox and Brewer of UC Berkeley have put forth the CAP Principle [4], stating that it is possible to design a distributed system that achieves any two of:

1. **C**onsistency
2. **A**vailability
3. tolerance of network **P**artitions

but it is not possible to achieve all three.

The invention of certificates as signed line items from the Public File was a choice to achieve A&P while the Public File achieves C&A.

There are frequent attempts to criticize one or the other of these mechanisms for not achieving the third desirable attribute and to come up with some new design that tries to achieve all three, but by the CAP Principle such attempts are doomed.

One must look at the specific security requirements of a particular application and decide which of the three desirable attributes can be sacrificed. This choice will be different for different applications.

6 Credential Classes

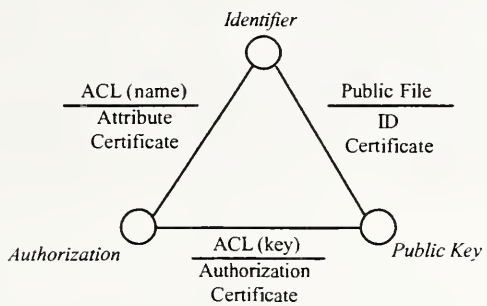


Figure 1: Credential Classes

Diffie and Hellman bound *Identifiers* to *Public Keys* through the Public File. Kohnfelder took line items of that public file and made ID certificates.

Those of us who wanted to use ID certificates as part of implementing access control, needed to get from *Authorization* to *Public Key*. That is, a transaction would come over the net with a digital signature verifiable by a public key and it would require authorization before it could be honored.

The knee-jerk reaction, relying on time-sharing system practice from the 1960's, was to use an Access Control List (ACL) binding authorization to login name. [By the way, Kohnfelder described the names in his thesis as login names, so this use of an ACL is not mixing metaphors.]

By the arguments of section 5, you can also convert line items of the ACL into certificates, and in this case, they become what we know as **attribute certificates**.

In 1996, however, a number of us started developing the third side of the triangle: **authorization certificates**. That is, something directly binding an *authorization* to a *public key*, rather than going through an *identifier*.

Also, by the logic of section 5, one can have protected database versions of the authorization certificate as we find with X9.59 and with the SSH access control file (.ssh/authorized_keys).

7 Authorization via ACL and ID

Figure 2 shows the use of an ACL and ID certificate to determine authorization. The ACL could be held locally in the machine that acts as gatekeeper for the protected resource, or it could live in some central authorization database that the gatekeeper queries over the network to approve any access request.

The security perimeter shown in Figure 2 indicates that both elements of the process – the ACL (or attribute certificate) and the ID must be protected equally. If the attacker can control either, then he or she can get improper access. However, there is a third vulnerability not immediately visible in the triangle diagram: the name. That is, the diagram shows one "Identifier" node at the top of the triangle, but in fact there are two identifiers involved: one on the ACL edge and one on the ID edge. The identifiers need to be the same, to link these two sides together, and some mechanism has to do the comparison to establish that.

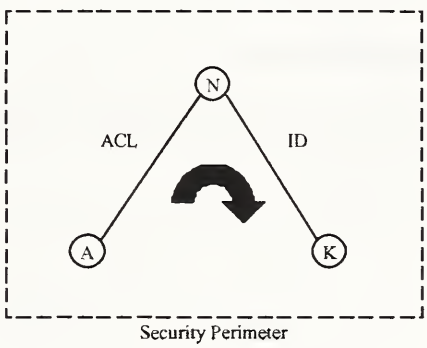


Figure 2: Authorization via ACL and ID

If that mechanism is executed by a computer and the names used are unique, then the comparison can be done with security. If the mechanism is executed by a human, then even if all names are unique, the John Wilson problem shows us that there will be mistakes made, and a clever attacker can exploit those mistakes to gain improper access. A human might make that comparison with each access, as we see with S/MIME or SSL, since in those cases the ACL is kept in the human user's own head. Or the human might make a name comparison when some database is administered by a human or a certificate is issued. In general, it is safe to assume a human will be involved at some point in the process because it is for human use that names are used in the first place.

When the method of Figure 2 is used, there is also the problem of administering the ACL side of the triangle. We consider two possibilities for that, below.

7.1 Authorize Everybody

The job of building an ID PKI is difficult enough that some people rebel against building an ACL as well. Instead, they use a one-line ACL: (*). That is, grant access to anyone who has an ID certificate. This isn't exactly the non-repudiation case, since it's not a question of having a signed contract. Rather, this is a situation like that employed by browsers when they decide whether to show the padlock icon as locked or unlocked. The icon is shown locked if the ID certificate is valid (and refers to the domain name from which the web page (or part of it) came).

The problem there is that users rely on that closed padlock rather than on a personal inspection of the ID certificate to decide whether to trust the web page and its server. This leads to a wonderful quote, from Matt Blaze, in the hallways of the RSA 2000 Convention: "A commercial PKI protects you from anyone whose money it refuses to take."

7.2 Authorization DB

You can, instead, build a real authorization database. Consider the database for something the size of a large PKI, with 6 million users.

If each user changes his or her entry in the database every two years, then there is one change to the database every 2.5 seconds of each normal workday.

Since this database is being kept in a central, secured location, it is being maintained by a staff of people cleared to enter that facility. Those people do not know all 6 million users. So, when a request comes in to change the authorization of some user, it must be investigated. If that investigation were to take a man-week, then the office would need more than 50,000 investigators, making this a very large operation.

No matter how large it is, the process begs the question of what makes these people administering the central database authorities on the data they are entering.

8 Direct Authorization

Another option is to go the other direction around the credential triangle, as shown in Figure 3.

In this process, there is only one point of attack, rather than the three of Figure 2. One would have to attack the authorization certificate issuer (or the maintainer of the authorization-to-key ACL).

One might ask why Figure 3 shows an ID when that ID is not used as part of the authorization process. The reason it is there is for forensics.

One can easily gather an audit log with entries identified by keys used (or their hashes, as more compact identifiers that are still globally unique). From processing those audit logs (or other tests) one might determine that a given keyholder (a given key) has misbehaved and needs to be punished. As Steve Kent quipped, during a DIMACS talk on this topic, 'You can't punish a key. What would you propose doing? Lop a bit off?'

You need to punish the keyholder. The simplest punishment is to put that key on a local black list. That keeps the keyholder from gaining access at the machine where you discovered the misbehavior. However, you might want to actually punish the keyholder, legally. For that, you need to locate the keyholder. So, you need a link from the key to the keyholder. This is indicated as an ID or name, but more likely it would be a whole file of information that would allow a security officer, lawyer or policeman to find the keyholder. This information could include the keyholder's name, address, phone numbers, bank accounts, friends, family, employer, etc.

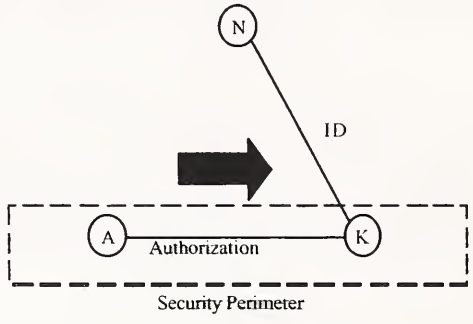


Figure 3: Direct Authorization

More interesting for those interested in PKI is the fact that this information binding a key to ID does not need to be either online or in certificate form. It is not used in the authorization process. It is used only during the manual process of punishing the errant keyholder. Therefore, the information could be kept in a non-networked PC in the security office. It could even be kept in manila folders. This affords the user with a certain amount of privacy. The user's identifying information need not be released to a resource guard whenever an access is made.

9 Delegation of Authorization

SPKI [7] permits delegation of authorization. SDSI [6] permits delegation of group membership. For some cases, the two mechanisms can be shown to be equivalent. The examples below can be achieved either way, but they will be described as authorization

certificate delegation – and contrasted with the use of a corporate authorization DB together with PKI for ID, according to the model of Figure 2.

10 Large Compa ny VPN Example

In this example, we deal with a large company that permits VPN access only to authorized employees. We consider it two different ways, first via a central authorization database and then by distributed, delegated authorization.

10.1 VPN Access via Central DB

Figure 4 shows part of an organization chart for a large company that has decided to give VPN access only to approved employees. We assume that employees are identified by some ID PKI, but authorization is maintained by a corporate authorization database. That database is maintained by some person or group, labeled A in the figure. A user, U, requests access by web page, since A and U are probably in different states if not countries and have never met one another and are not likely ever to meet one another.

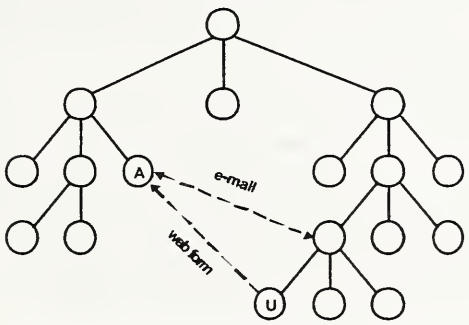


Figure 4: Central Authorization DB for VPN Access

If A were simply to enter U in the database in response to the web form, then there is no security to speak of in the system. So, A looks in the corporate central employee database to find U's manager and sends an e-mail, asking if U should be allowed VPN access. When the answer comes back in the affirmative, A enters U in the authorization database and U has VPN access.

There are at least two problems with this mechanism:

1. A sends an e-mail to someone whose name is very much like the name listed in the employee database as being U's manager. Thanks to the John Wilson problem, that does not mean that A sends an e-mail to U's manager.
2. The mechanism as described above implicitly grants every manager in the company the power to grant VPN access. Correction of that limitation would greatly complicate the database administration process.

In the next section, we address these problems.

10.2 VPN Access via Delegated Direct Authorization

In Figure 5, we accomplish the same function, but by authorization certificate and delegation of authorization. The organization or person, A, responsible for the ACL of the machine(s) that enforce VPN access, enters a public key into that ACL, as the head of a tree of certificates to be empowered to have VPN access. Person A then uses the matching private key to grant authorization certificates to his or her manager. That authorization flows, by authorization certificate, up the organization chart to the CEO and from there down the entire organization, but only into those groups where VPN access makes sense. In particular, as shown by the heavy lines, it flows from A to U and therefore has the same effect as the process shown in Figure 4.

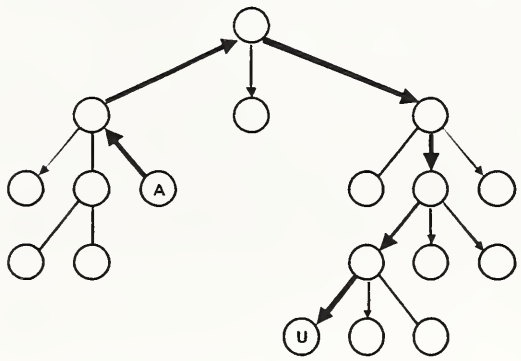


Figure 5: VPN Access by Direct Authorization

The process of Figure 5 has some distinct advantages over that of Figure 4:

1. Each grant of authorization is between two people who work together and therefore can authenticate one another biometrically, in person. Names are not used in the process, so there is no security flaw from the John Wilson problem.
2. Each grantor of authorization is in a position to know better than anyone else whether the grantee should receive that grant of authorization.
3. These decisions – of authentication and authorization – are made with almost no effort. No investigation is required.
4. The work that used to be done by A is now distributed around the company, although it is miniscule at each place a decision is made. This frees A to do other, more interesting

work. That, in turn, saves money for the corporation.

So, this process both saves money and increases security of the administration of the authorization process.

11 Cross-company B2B P.O. Example

The example of the previous section dealt with operations within a single company that had a single PKI. We now address a pair of companies that want to do electronic purchase orders, with orders automatically processed by computers in company A when they are signed by authorized keys (keyholders) within company B. Each company has its own, independent PKI.

11.1 B2B via Central DB

In Figure 6, we build a structure analogous to Figure 4. The employees of Company B that should be authorized to sign electronic purchase orders are shown in gray, while there is one person (or group) in Company A that maintains the ACL on the machines Company A uses to process purchase orders automatically.

The purchasing agents must request, somehow, to be added to the ACL, and the maintainer of the ACL needs to verify the propriety of each such request. This request goes from company B to company A. The verification of that request is a dialog initiated by the responsible parties in company A.

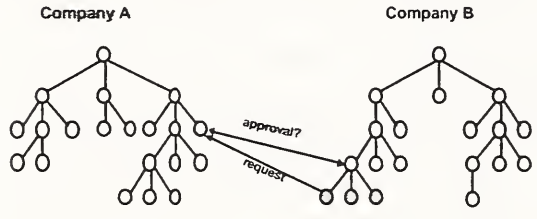


Figure 6: B2B via PKI and Authorization DB

11.1.1 Bridging of PKIs

The first thing we observe is that for ID's issued by Company B's PKI to be usable within Company A, we need to bridge the two PKIs, either with a bridge CA or by adding each PKI root to ACLs in the applications on both sides. However, when we bridge the two PKIs, we make the John Wilson problem worse for both.

1. It is made worse just by having more people under the same namespace. This leads to more name collisions and more mistakes.
2. It is possible that name uniqueness is violated. Company A could have been very careful to have only one "John Q. Wilson" and Company B could have been very careful to have only one "John Q. Wilson", but after the bridge, there are two. What is missing is some entity that would control the issuing of names within companies A and B, before they decide to bridge their PKIs. There is no such entity today, and the experience of ICANN (The Internet Corporation for Assigned Names and Numbers and other Top Level Domain efforts) suggests that no such entity will ever exist.

11.1.2 Employee Data

In the process of Figure 4, the maintainer of the ACL consulted the central employee database to find the party to contact to get approval of the request for authorization. Company A does not need the entire employee database of Company B, but it does need enough of that database (or remote access to a view of that subset) to permit it to make the proper authorization decisions.

This kind of data, especially linked to names, is traditionally considered confidential by companies. A special exemption would have to be made in this case. Meanwhile, the data that company A needs would have to be made available under strict access controls, and the authorization database for those access controls becomes an additional problem to address. This way leads to uncontrolled recursion.

11.2 B2B via Delegated Authorization

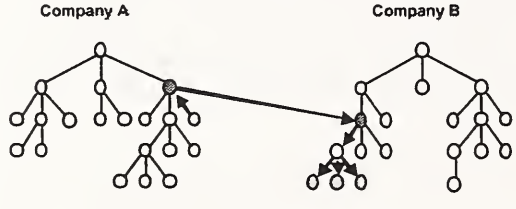


Figure 7: B2B by Delegated Authorization

In Figure 7, we show the same B2B process, but by delegated authorization rather than authorization database and ID PKI.

In this figure, we introduce a new node color (darker gray) to stand for the executives of the two companies who meet to decide to form the business relationship. These executives exist already and perform this function. Two companies do not spontaneously decide to do business with each other. There is a period of investigation and decision-making before that decision is made. The decision is usually sealed with a contract and the contract is signed by individuals of the two companies. These meetings might be electronically intermediated, but they are meetings of people rather than of computers.

In Figure 7, the permission to delegate the authorization to have purchase orders accepted and processed automatically is granted from the person or group that maintains the gate keeping machines in Company A to the executive in Company A who is going to sign that B2B contract. After the signing of that contract, the executive from A grants the executive from B the power to authorize such purchase orders. The executive from B takes that authorization back to Company B and delegates it to the purchasing group manager who certifies the individual purchasing agents within her group.

Note that this process:

1. does not use a bridge CA, so it saves that expense,
2. does not use names, so there is no John Wilson problem,
3. does not require either company to access the other company's confidential employee data,
4. does offer improved security, just as we saw in Figure 5.

12 The AND Effect of ID PKI

There are those who claim that doing authorization computation via the combination of ACL and ID cert is important because it gives you a logical AND of two conditions: the authorization and key validity. The assumption there is that a valid ID cert does more than name the keyholder. It also represents certain security conditions. It attests to the key itself not having been revoked and might also attest to the keyholder's continued employment.

This is valuable functionality. However, the use of an ID instrument for these other characteristics is not the best system design. What if some application cares about key compromise but not about continued employment? This mechanism does not allow the application designer to separate those three attributes of a key: ID, non-revoked status and continued employment. It also does not allow the application designer to specify the AND of other functions, without loading those onto the ID instrument as well.

A cleaner design is to use an explicit logical-AND and specify the conditions individually, each with its own certificate (chain). Each of these attributes can be bound to a key by an authorization certificate, with the certificate issued by the proper authority. That is, a 24x7 key loss reporting service might be in charge of providing online validity information of the non-revoked status of a key while a corporate HR office might provide information about continued employment. These attributes do not require any ID. They can be bound directly to a key. By contrast, loading all of these attributes into an ID certificate by side effect requires the ID certificate issuer to be the authority on all of those attributes – something that may be difficult to achieve, organizationally.

[Note that SPKI/SDSI [7] includes a construct called the "threshold subject" that permits expression of such "AND" conditions in ACL entries or certificates. The code that implements threshold subjects is available in [1].]

13 Conclusions

This paper makes the case that there are fundamental problems with the original ID-based notion of a PKI, in that it fails to take account of certain realities (such as human limitations). Instead, we can use delegated, distributed authorization, which does not suffer from those fundamental problems. Two examples of the use of distributed authorization were given, in brief, but there are a great many other examples. The reader is encouraged to try applying these techniques to other problems, as was done in [3].

14 References

[1] CDSA: http://developer.intel.com/ial/security - source code and documentation, including a full implementation of SPKI and SDSI certificate reduction. This link leads to the open source repository for that code.

[2] Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976.

[3] Steve Dohrmann and Carl Ellison, "Public Key Support for Collaborative Groups", Internet2 PKI Workshop, April 2002.

[4] Armando Fox and Eric A. Brewer, "Harvest, Yield, and Scalable Tolerant Systems", Proceedings HotOS-VII, 1999

[5] Kohnfelder, Loren M., "Towards a Practical Public-key Cryptosystem", MIT S.B. Thesis, May 1978.

[6] SDSI: http://theory.lcs.mit.edu/~cis/sdsi.html

[7] SPKI: http://TheWorld.com/~cme/html/spki.html

Report: EDUCAUSE – NIH PKI Interoperability Pilot Project

Peter Alterman, Russel Weiser, Michael Gettes, Kenneth Stillson, Deborah Blanchard, James Fisher, Robert Brentrup, Eric Norman

Background

Under mandate to adopt broad electronic business methods by October 2003, Federal Agencies are working hard to figure out ways to put their business on-line in a way that is secure. A leading contender to make e-government secure and trustworthy is public key cryptography. At the same time, far-sighted institutions of higher education have been busy deploying PKIs and issuing digital certificates to their faculties and staffs to enable secure, electronic business with the government and with each other. These institutions wish to use their locally-issued digital credentials to do electronic business with the government securely. The NIH, in turn, wishes to be able to rely on business partner-issued digital credentials, thereby avoiding the cost and administrative burden of issuing and managing electronic credentials. NIH and EDUCAUSE jointly constructed a PKI interoperability pilot project that demonstrated the ability of the Federal Government to receive electronic forms signed with digital certificates issued by institutions of higher education.

Description of Project

In order to address this situation, NIH and EDUCAUSE conceived a research project that would demonstrate a simplified approach to submitting digitally signed electronic grant applications to NIH. Although the project used an electronic grant form, in reality any form could have been used; the point being that the project's approach is applicable to any electronic form or file. The explicit goals of the interoperability project were to:

- Receive grant applications as digital forms signed with two different, validated, digital certificates each (an NIH business process requirement);
- Use digital certificates issued by three (later changed to five) participating academic institutions;
- Demonstrate interoperability among different CA vendors' products, including PKI service providers.

A key consideration in the design was that NIH would be a relying party with respect to the digital credentials used to sign the electronic grant applications. This is important for several reasons. For privacy and resources reasons, NIH would like to avoid issuing digital credentials to individuals and institutions. Experience trying to maintain an up-to-date, accurate inventory of research faculty and staff has demonstrated to NIH the futility of a government-centric, centralized approach to issuing and maintaining credentials of faculty engaged in government-sponsored biomedical and biobehavioral research. On the other hand, academic institutions have a much easier time of keeping track of their faculty and graduate students – so long as they wish to continue to receive paychecks.

Many academic institutions are in the process of deploying PKIs and issuing digital certificates to faculty, staff and students to facilitate e-business on campus, and these schools have voiced a clear desire to use their locally issued digital credentials for doing business with the Federal government. Thus, the logical design plan was to encourage deployment of institutional PKIs.

To support the work of the project, NIH and EDUCAUSE contracted with Digital Signature Trust (DST) and Mitretek Systems to complete key portions of the work. Fundamental work resolving directory issues was done by Georgetown University.

NIH provided the participating institutions with a Microsoft Word Template version of the *PHS-398, Application for Research Grant* form, to be used as the model for this pilot. The form was made available for download at an NIH web site. (Although not selected by any participant, a PDF version of the PHS-398 was made available to all institutions for the pilot.) This was done to provide the institutions with an electronic document that could be manipulated locally by common desktop software applications. Desktop signing of the Word templates was accomplished using Assured Office (now ProSigner) software, a Microsoft Office Suite

plug-in and standalone application developed by E-Lock (now Lexign). ProSigner, however, only works on the Microsoft Windows platform.

Phase One of the project incorporated the following assumptions and features:

- **A form that could be shared between the Principal Investigator and the Authorized Official of Record (AOR) at the research institution.** The PHS-398 is completed by PIs and the AORs, also known as Institutional Representatives (IR) in recognition of the fact that NIH funds institutions, not individuals. The form must allow for completion by multiple users, although only one of these users will submit the form to NIH.

- **A form that could be digitally signed with multiple digital signatures.** Both the PI and an IR sign the PHS 398. Both digital signatures need to be validated, that is, checked to verify they are good, when the form is submitted to NIH. The PI is typically part of a research operation of an organization. The institutional representative is an administrator, typically called the Authorized Official of Record (AOR) or IR. The two may be hundreds or thousands of miles apart. Bringing these people into a room at a single moment is often not feasible. Further, the AOR or IR may be handling numerous forms at a single time, related to many different investigators.

- **A form that could be completed with virtually no additional software requirements for the PI and IR/AOR.** In order to allow for maximum scalability, the team decided that the adopted solution should have as small a client footprint as possible, not only because of difficulties in downloading and installing products, but also because Information Technology (IT) departments are averse to installation of software that is not part of the standard configuration supported by the Institution's IT environment. This concern arises from added cost and support (which also translates to cost) requirements.

- **A form that could utilize commercial-off-the-shelf (COTS) digital signing products.** Based on our analysis of COTS digital signing software, the product that we recommended, E-Lock Web-Signer (now Lexign ProSigner), would sign not only portable document format (PDF) files, but also generally any other file type. Due to the number of users participating in this pilot, it was more cost effective to use the per-user-priced Assured Office (ProSigner) rather than the recommended Web-Signer, which is priced on a server basis.

  Research into the capabilities of Adobe Acrobat reader revealed that the reader software supported verification of signatures, but did not support digital signing or digital certificate validation natively. Additionally, Adobe Acrobat software, as distributed by the manufacturer, requires additional software plug-ins to be added to the desktop to allow it to function with PKI certificates that would be applicable to the project requirements. By using a COTS product that worked correctly with any file format, including Word templates, a separate plug-in for Adobe did not need to be created.

- **Form could be digitally signed and sent as an email attachment, requiring no changes to the NIH mail server.** In order to best meet the needs of the constituents of the pilot, e.g., the research institutions and NIH, the Word template needed to be completed, digitally signed, and emailed as an attachment to the NIH OER recipient. This allows for easier submission of the form, requiring no changes to the NIH email server or to current database or web servers. Furthermore, it greatly simplified the submission process for the institutions. The fact that their email systems logged the sending of the message as proof of date and time of submission was a serendipitous extra benefit.

PKI Bridges

To allow NIH to successfully validate the digital certificates affixed to the electronic grant applications, EDUCAUSE deployed a Higher Education Bridge Certification Authority (HEBCA) prototype structurally similar to the Federal Bridge Certification Authority (FBCA) prototype. With the support and approval of the Federal PKI Steering Committee, which included a generous grant, the two bridges were cross-to-certified and currently interoperate at the test level of assurance. Participating institutions' PKIs cross-certified with the Higher Ed Bridge while a proxy NIH CA cross-certified with the Federal Bridge. Thus, a trust path was created between NIH and the institutions

through the bridge-bridge infrastructure created to support the project.

Trust path discovery and validation for the bridge infrastructure model required use of specialized software. Mitretek Systems modified the Certificate Arbitration Module (CAM) originally created for the GSA Access Certificates for Electronic Services (ACES) program (an umbrella contract mechanism allowing the Government a acquire a broad range of PKI services) and added DAVE. The CAM/DAVE became the validation service used by Assured Office to validate the digital signatures affixed to the completed MS Word templates. How this worked will be explained further on in this paper.

Significant issues were encountered in attempting to link the different directories that supported the institutional PKIs. To resolve them successfully, the team found it necessary to use an Internet 2-supported "registry of directories," described below, developed by Michael Gettes of Georgetown University.

Interoperability

In addition to brokering trust among discrete PKIs, the Federal and Higher Education bridges also supported Certificate Authority (CA) product interoperability. The University of Alabama at Birmingham used the DST TrustID certificate service (RSA technology); the University of Wisconsin-Madison used the Netscape iPlanet CA and Dartmouth College used the Entrust CA. The University of California Office of the President and the University of Texas – Houston Health Science Center used the VeriSign On-Site CA service. (The latter has not yet been demonstrated to operate successfully in the pilot, but is expected to be operational shortly.)

By using interoperating bridges, the overall number of cross-certifications required within the community of interest was reduced. Policy mapping decisions were offloaded to the Bridge policy authorities. This model allowed disparate PKI communities to be "bridged" together. Its disadvantages were also evident: liability issues arose by offloading policy mapping functions to a Bridge policy authority; it was heavily dependent on a distributed directory system that was vulnerable to failure in a number of locations. Certificate path construction was complex, and there were disparities between the underlying directories, e.g., X.500 vs. LDAP. If proper certificate constraints were not used, then security issues were destined to erode the trust in the infrastructure. Depending on the policies of the Bridge Policy Authority, peer-to-peer cross-certification of CAs still could be required.

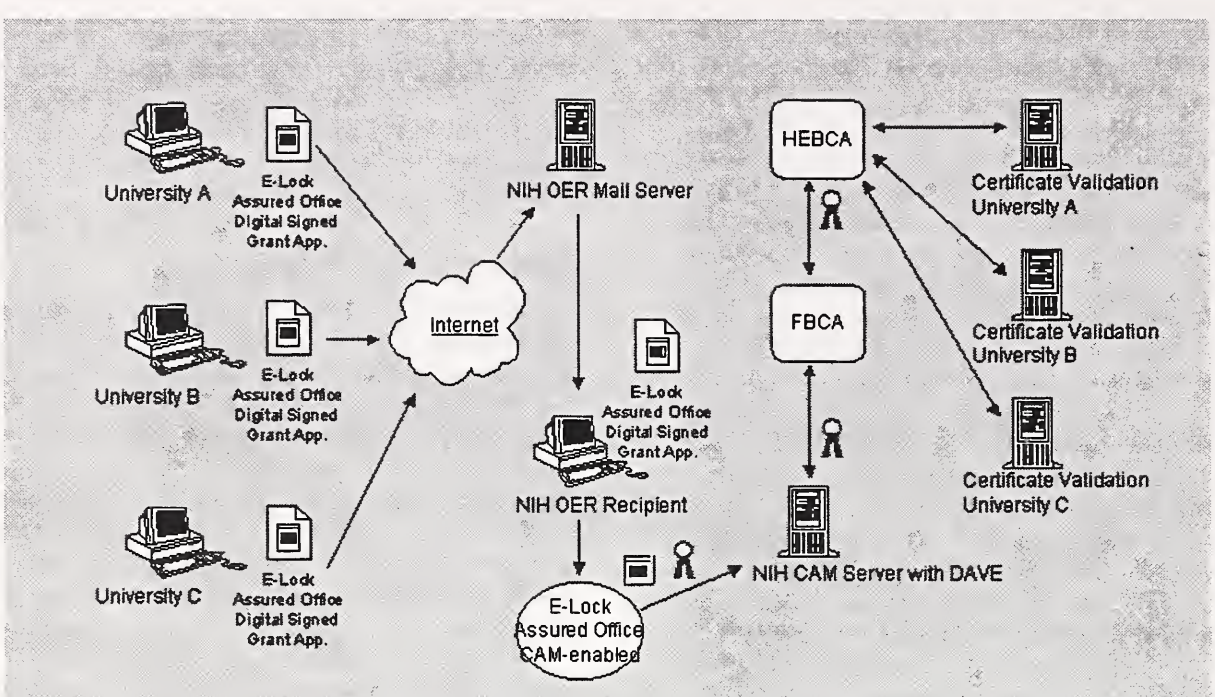
University CA Issues

As part of this project, university participants utilized their own CA software. The University of Wisconsin, for one, utilized the iPlanet CMS as its CA for university personnel certificates. This was one of the most challenging experiences – especially for the directory services. Their CA came integrated with the iPlanet LDAP directory in its default configuration, which assumed the CA would be used for an enterprise PKI in which users existed within the directory prior to obtaining the end entity certificate.

Because of this assumption, cross certifying with the HEBCA took some effort, specifically obtaining a PKCS#10 certificate request of the University of Wisconsin's root. This was found to be written as a file, instead of provided to the administrator. The publication of the cross certificate pair to the iPlanet directory had to be performed manually. The iPlanet software came with the *CertificationAuthority* object class and included *CrossCertificatePair* as one of the attributes. Using the *LDAPModify* command from the command line, the CrossCertificatePair could be published the directory

The Certificate Arbitration Module (CAM)

The CAM is an application-level router that efficiently and consistently routes certificates from relying party programs to the issuing certificate authorities (CAs) for validation. By interfacing directly with the CAM, a relying party application can interact seamlessly with multiple CAs. CAM is also flexible; it allows RSA-based certificates to be validated with the Certification Authority. The CAM runs as a separate process within the agency's security domain, allowing the agency to manage the resources and controls necessary to support the validation processing at the enterprise level. Applications interact with the CAM through a simple validation API that communicates over TCP/IP or by using a Microsoft ActiveX control.



Phase 2 of the NIH-EDUCAUSE Interoperability Pilot Project with FBCA and HEBCA

When a digital signature and the corresponding signer's certificate are presented to a PKI-aware application and the application does not recognize it, the application submits the certificate to the CAM. The CAM parses the certificate, verifies that it has not expired and checks to see that the certificate issuer trusted by the application. The CAM then either uses stored instructions or looks at the Authority Information Access (AIA) extension within the certificate to obtain the location of the OCSP validation service cited by the issuing CA. The CAM then builds an OCSP request, digitally signs it with a certificate issued to the CAM, and submits it to the OCSP server for validation.

When DAVE is incorporated, the issuing CA no longer needs to be known *a priori* (via configuration) and trusted by the CAM. Instead, DAVE's trust anchor is known *a priori*, and DAVE performs the steps of trust path discovery and validation, the latter typically via Certificate Revocation Lists (CRLs).

The CAM *Validate Request* message contains three parameters: a message type, an Application ID string, and the DER-encoded certificate to validate. CAM then performs certificate validation on behalf of the application and returns a response message back to the application. The *Validation Response* message contains five parameters: message type, certificate status, an ACES profile check code (not used in this project), an ASCII representation of the parsed certificate, and the binary digitally-signed validation response message received by the CAM from the CA's validation service.

As the application-to-CAM communication utilizes TCP/IP, an Intranet (or Internet) connection must exist between the application and the CAM. The validation request response messages are transmitted in "Little Endian" byte order, so applications integrating with the CAM must take this into account and translate the messages if they are not running on a non-Intel platform. The NIH and many of the academic institutions used Intel platforms, so this was not an issue for them during the pilot project, but it was noted that a significant Macintosh users are part of the NIH client base.

The CAM receives the signed OCSP response from the issuing CA's Responder, verifies the signature, and parses the response to obtain the certificate status. The CAM logs the response (providing an audit trail) and packages the status along with additional information in the *Validation Response* message, as discussed above. While the

functionality of each CAM is limited to a single security domain, it is also ideal for a one-stop gateway or portal architecture.

Enabling applications to utilize the CAM for validation is a fairly straightforward task. Several key points must be taken into consideration, though (See CAM Communications Specifications - Version 3.1.0 at http://cam.mitretek.org/cam):

The original design requirements assumed that the CAM and the application are running in the same security domain, that is, the protocol between the application and the CAM itself were not currently authenticated:

- The CAM server runs on a Microsoft NT 4.0 or Windows 2000 platform;
- The CAM utilizes TCP/IP to transport the validation request, responses to and from the CAM;
- The CAM trust model, when not extended by DAVE, is that the CAM is authoritative; only certificates issued from a CA explicitly trusted by the CAM are validated, hence applications have no need for further validation.

CAM Implementation

To date, the CAM has been deployed successfully in a number of instances within the Federal Government. Although not in broad use today, this growth trend should continue over time. Examples:

1. The SSA is in the third year of its "Annual Wage Reporting" (AWR) pilot and the second year of utilizing the CAM as a signature validation service for electronic AWR filings. This year's pilot includes the use of a simplified signing control, "simple sign" to calculate the signature hash, sign the signature hash, and submit the filing to the SSA services. There, the signature is validated through the CAM validation server. Not only is SSA accepting signatures through the ACES program, it has added the State of Washington PKI as a trusted issuer within their CAM trust list;
2. FEMA utilizes the CAM validation service in several programs; first, to provide certificate-based access control to several critical databases available to emergency personal during disasters; second (deployed since the September 11th attacks), a government assistance program for local government agencies that are applying for FEMA assistance. This application allows electronic submission of grant applications as well as certificate-based access to check on the status of the application by the applicant;
3. NIST has developed an electronic grant application submission and review workflow to support its research grants program. This program utilizes both ACES and NIST-issued certificates and handles signature validation via the CAM;
4. NTIS has enabled its labor union wage reporting system, utilizing CAM for signature validation of union officials when union wage reports are filed with the NTIS servers. The reports are then accepted and the information fields verified and fed into the Agency's back-end workflow system;
5. The EPA ran a pilot, "CDX," that enabled digital signing of pollution reports by reporting agencies and businesses. The program has recently incorporated a full-blown reporting exchange that includes the digital signatures, submitted reports, and their validation at the point of acceptance.

Discovery And Validation Engine (DAVE)

DAVE is an open-source software package that provides X.509 certificate trust path discovery and validation services as a TCP/IP accessible Microsoft Windows NT/2000 service. DAVE may be used as an add-on to the CAM, extending CAM-enabled applications to hierarchical and cross-certified PKI domains.

Configuration settings for DAVE include:

- A certificate corresponding to the "trust anchor." All trust-paths end at this "most trusted CA;"
- An LDAP server name and port to use for retrieval of certificates and CRLs and/or ARLs.

The incoming request protocol used by DAVE is the same as that used by the CAM. Starting with CAM version 3.6a, the "CAM-linking" and "default CA" capabilities may be used to defer validation to DAVE for CAs not specifically listed on the CAM trust list. The outgoing request protocol for certificate path discovery and for CRL retrieval is LDAP, both for certificate path discovery and CRL retrieval. OCSP-based validation may be added at a later time. CAM already provides OCSP support, but only for directly trusted CAs, not ones located by path discovery.

DAVE applies multiple techniques to construct the certificate path. When the location of the issuer's certificate is given in the AIA field of the certificate in question, DAVE contacts that specified LDAP or X.500 directory directly. When explicit locations are not conveyed in the AIA field, or when a complete trust path has not yet been constructed, DAVE switches to a second technique, issuing LDAP "read" requests to its default LDAP server which, in turn, discovers and queries the correct directories. Such discovery is made by way of hierarchical CA certificates and cross-certificates. The explicit steps taken are: (1) read the issuer field from the certificate in question and call this the target domain name (DN), and (2) do an LDAP read for the target DN, asking for the return of both all *cACertificate* and *crossCertificatPair* attribute values.

This places two requirements on the directory infrastructure DAVE utilizes:

1. PKI objects (certificates, cross-certificates, and CRLs / ARLs) must be properly stored in a part of the Directory Information Tree (DIT) with a DN equal to the subject field of the object(s);
2. The LDAP server to which DAVE connects must know of and be able to retrieve any intermediate certificates or CRLs / ARLs along the constructible paths. This generally implies directory chaining agreements or an LDAP referral arrangement.

Internally, much of DAVE's functionality is provided by other open-source packages:

- The Certificate Management Library (CML) v2 provides path construction logic and certificate validation functions;
- Crypto++ provides cryptographic functions for signature verification;
- Netscape LDAP SDK DLL (in object form; no source available) provides referral-enabled LDAP client functions;
- S/MIME Freeware Library (SFL) provides MIME processing functions, and an abstraction for Crypto++;
- Certificate Arbitrator Module (CAM) code is taken from CAM for NT service abstraction and basic core library functions that provide thread safety, safe memory allocation, logging, etc.

DAVE Status

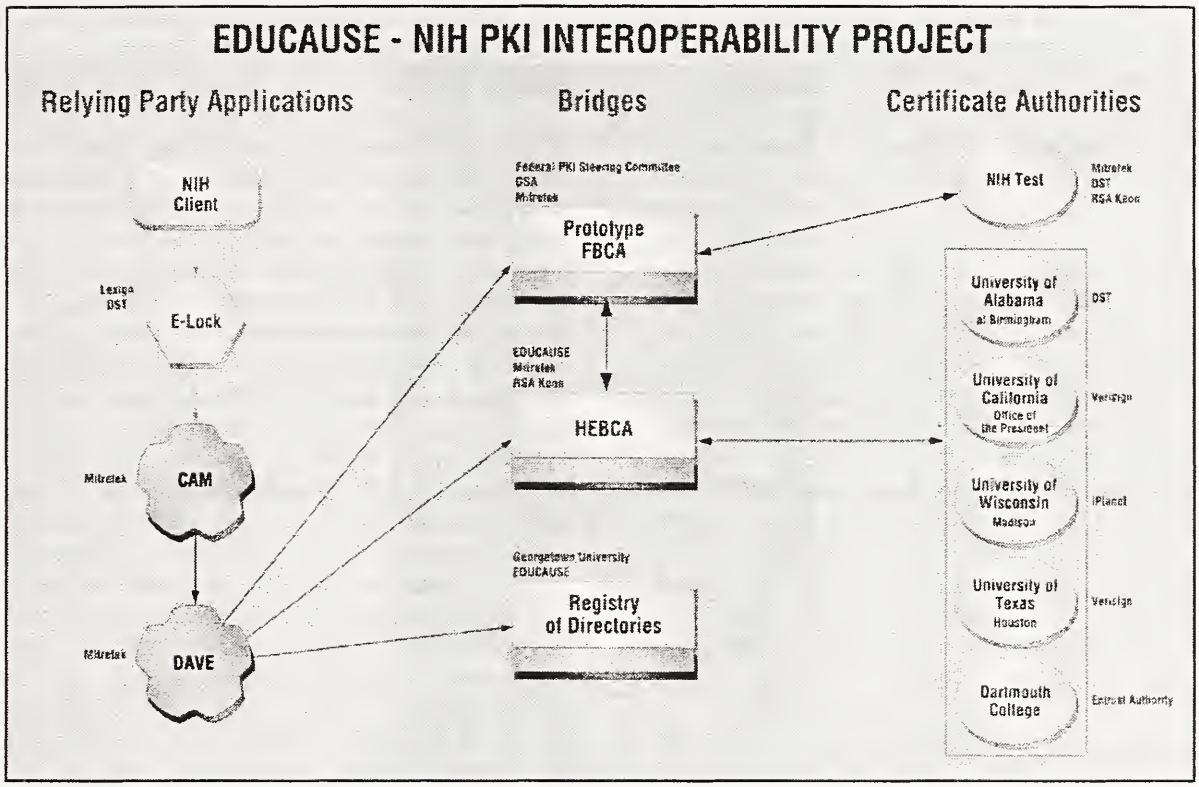
Initial development of DAVE is completed, and the source-code will be freely available shortly. DAVE has been tested in a number of trust topologies, with a variety of certificates issued by different CA product vendors.

Interoperability Pilot Test Environment

NIH is a participant in the Federal Bridge CA (FBCA) prototype and has a CA cross-certified with the FBCA prototype. The universities are participants in the Higher Education Bridge CA (HEBCA) prototype and their CAs are cross-certified with HEBCA. When a certificate is validated in this test environment, it demonstrates a trust path that traverses hierarchical and cross-certificate-based PKI domains, multiple bridges, multiple CA product vendors, and both LDAP networking mechanisms, directory chaining agreements for the FBCA, and an LDAP referral-based directory networking for the universities.

The pilot project test environment pictured above involved two users at three of the universities sending "dual signed" grant request forms using certificates issued by their respective CAs (DST/RSA, iPlanet, Entrust). The digitally signed forms were sent as attachments via standard e-mail to a user at the National Institutes of Health (NIH).

The NIH user received the e-mail message and used the CAM-enabled Lexign ProSigner application to validate the attached, signed form. ProSigner was configured to contact NIH's CAM, which contained a single-item trust list, deferring validation to DAVE. DAVE was configured with NIH's self-signed CA as it's trust anchor, and an LDAP meta-directory (referral-based) as its LDAP starting-point. On an initial run, this system was able to validate both signatures on the form within 20 seconds. On a second test run, when DAVE had automatically cached the certificates of the path, validation took place in under 5 seconds.



Pilot Project Description, highlighting positioning of CAM and DAVE in the trust discovery path

Directory Overview

Currently, the FBCA environment relies heavily on the use of X.500 directory standards to facilitate path discovery and path processing. This is partially due to the Federal Government's extensive experience with X.500 directories. Although the FBCA does utilized the LDAP v3 protocol as the primary protocol to the bridge directory, another X.500 based protocol is utilized to connect transparently to a distributed mesh of directories. Certificates that make up a full path may reside in external directories that are connected to the bridge directory transparently. The FBCA environment relies on the X.500 DSP protocol to chain automatically to the external distributed directories to retrieve the CA certificates, CRLs, and ARLs that are needed to perform path processing. The DSP protocol is managed through the use of 'Chaining Agreements' that manage authentication and retrieval of attributes and values that reside on these external directories. This environment has been tested in small scale by the FBCA with several directory and CA products.

Directory Issues

The FBCA model presents two fundamental challenges to the development of a HEBCA world. First, the FBCA was constructed under the assumption that X.500 directory services would be used for both the bridge and the agency directories, and the location for publishing certificates (including objects containing client, CA, CRL and ARL information) would be known *a priori*. Second, using directory request chaining to resolve requests for X.509 objects which the X.500 standard supports presents difficulties for LDAP implementations, since LDAP does not have a uniform mechanism for chaining requests and not all LDAP clients understand LDAP referrals. In the Higher Education computing environment, as in the marketplace, the use of X.500 directory servers is quite limited and LDAP is the predominant directory server technology employed for enterprise-class directory-enabled services. Since directory chaining is not one of the X.500 capabilities brought forward into the LDAP specification, the project team developed techniques for getting around these limitations.

Fundamental to the Federal BCA model is the notion that a request for an object associated with a *SubjectName* (Subject or Signer) is performed directly and not by issuing search requests. An application simply calls the "getDN" function and the directory infrastructure resolves the DN for the application.

It is also important to note that without an *AIA* extension in the certificate, the issues related to chaining and locating objects become significant. Very little software makes use of *AIA*, however, DAVE and CAM both use the *AIA* extension if it is present. If an HTTP URL form is present, DAVE will bypass directory lookups and use HTTP directly. If an LDAP URI form is presented to DAVE, the module directly queries the given LDAP server for the given DN; if it is a DN-only form, DAVE queries the default LDAP server using the DN from the *AIA* field, not the DN from the issuer/subject fields. The same logic applies for CDP fields when getting CRLs.

Chaining

This paper does not attempt to describe all aspects of chaining per the X.500 specification, but simply makes note of some of the reasons for choosing the X.500 chaining methodology and presents challenges for an LDAP equivalent methodology.

What typically transpires in the BCA model is that an application receives a form or document with an affixed certificate. To validate that certificate, the CRL associated with the issuer of the certificate must be queried to see if the received certificate is still valid. The application (or an associated certificate-handling module) extracts the *Issuer Subject Name* from the certificate and requests the DN that is the Issuer *SubjectName* from a locally defined and -configured directory service. In the X.500 context, the DSA has the responsibility for performing any name mapping and for chasing down the DSA that houses the object associated with the DN. Since this involves accessing other directories, the authentication credentials are appropriately passed to other directories for proper access control to required information. This places the burden of translation and location on the DSA, and the application has to know little of the "magic behind the curtain." This "magic" is commonly referred to as "knowledge references" and there are various types to describe and implement different behaviors. One reference describes a chaining agreement between two DSAs. Another reference describes a referral, which is returned to the application to be handled as the application sees fit. From an application perspective, this is a reasonable mechanism.

In the LDAP world, however, chaining doesn't exist formally. It is relatively easy to implement a simplified version of chaining using LDAP, but there is no standard defined for the activity. In the pilot project, the application has to chase the DSA associated with an issuer DN. While applications usually call library functions, this model potentially increases the complexity for the applications, depending upon which LDAP libraries are used. In the case of the open-source *OpenLDAP* implementation, a derivative of the University of Michigan SLAPD implementation, the libraries handle referral chasing rather well. Nevertheless, for both referral and chaining, there is still work that must be done at the DSA to define knowledge references (and, of course, to test those references). Thus, in LDAP-based models, applications must know more about the process of certificate validation, calling library functions and performing the work, but this type of activity is commonplace for LDAP-enabled applications. If handled properly, the X.500 model and the LDAP model are equally transparent to the application.

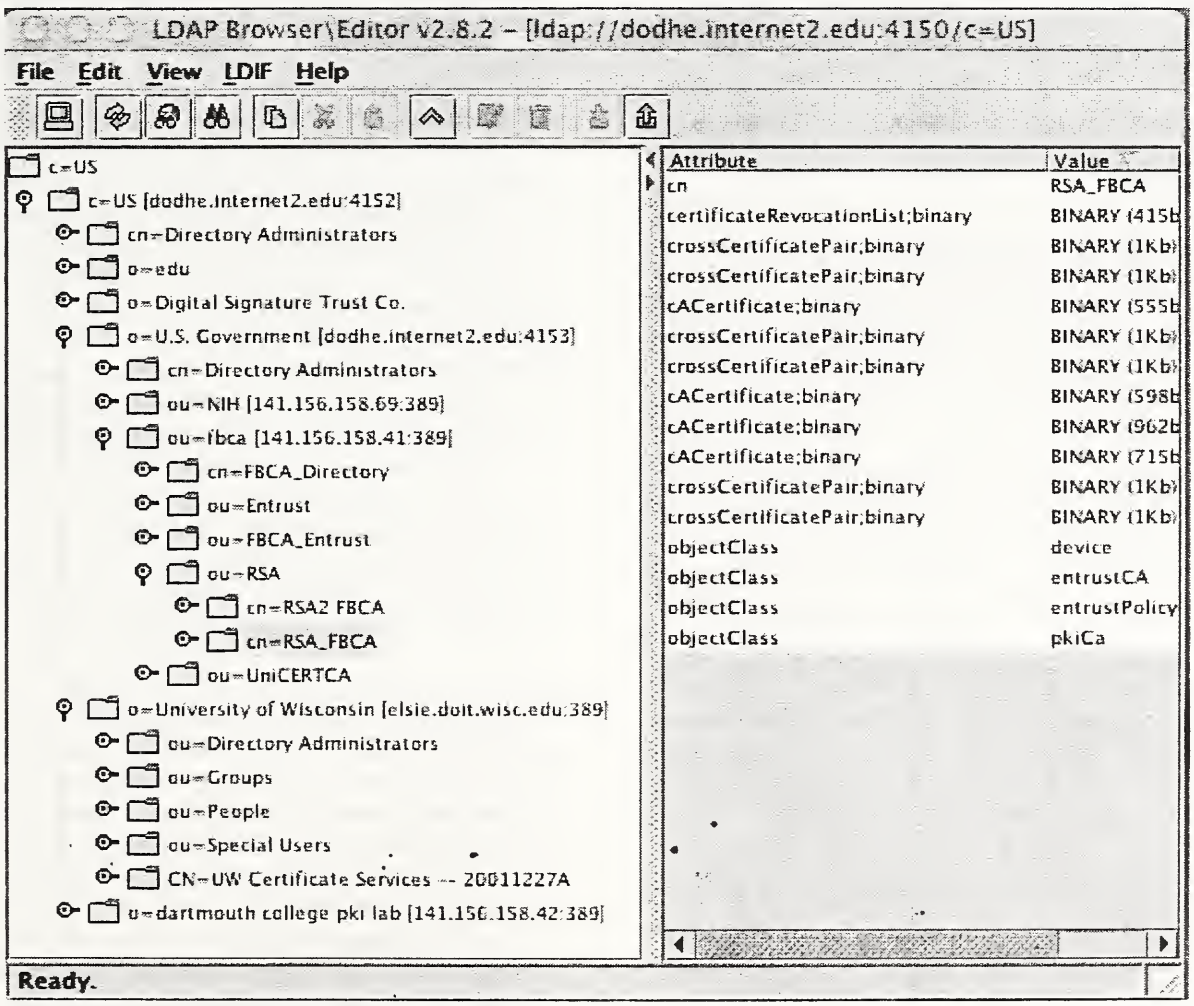
One important lesson from the FBCA work is that chaining agreements between different vendors of X.500 DSAs is quite problematic - to the point that a workaround was required for successful demonstration of the project proof of concept. Not every institution has the same Certificate Authority product or directory service product, and if they do have the same products, they might be different versions that are incompatible. This last situation particularly caused problems at the Dartmouth College PKI Lab, both with the CA and the directory (which had to be upgraded to the latest version, and even then had numerous directory chaining issues though it was an X.500 directory). Finally, the DSP protocol is time-dependent and hence two directories that are tied by chaining agreements require time synchronization in order to operate correctly.

Resolving Objects via LDAP: Registry of Directories

Given that LDAP has no inherent chaining capability, a knowledge reference service was developed that the LDAP-enabled, BCA-aware applications utilized. This service is a Registry of Directories (RoD). The RoD is an LDAP directory

utilized to provide "smart referrals" for CAs which are cross certified with the HEBCA, but which do not have X.500 directories that support the DSP chaining protocol. The RoD provides DN entries for the organization CA and an LDAP-based URI referral to the organization's directory, where the CA certificate, CRLs and ARLs actually reside. This allows DAVE to access the directory of the institution quickly and to retrieve the CA certificate, CRLs and ARLs in order to perform the path development and processing needed to bridge a trusted path with generic LDAP read and LDAP search operations. This is not much different from the FBCA concept, except that multiple directories are accessed via LDAP by the path processing software as opposed to being accessed by a single bridge directory, which then chains to the distributed directories of the participating CAs. The advantage of this is the simplicity of management of the RoD, as opposed to establishing separate chaining agreements across numerous distributed directories. This is particularly important given the sheer number of institutions, and the diversity of their infrastructures and needs.

The project created the RoD on a test system (dodhe.internet2.edu) using different ports to simulate a federated administration model of this registry. Our first implementation required the application be configured with the top of the registry service defined - or pointed to - any DSA associated with the RoD service. Each RoD DSA was configured with a superior reference, which implied that any DN requested that was not managed by the current DSA yielded a referral to the top of the RoD. The RoD figure below shows an expansion of the RoD hierarchy for this phase of the project. For each root, we configured a new RoD hierarchy. We defined two roots for this part of the project, one for c=US and one for dc=edu. Only the c=US branch is shown below, since this presents the FBCA test environment, as well as the HEBCA test environment.



Registry of Directory hierarchy for Phase Two of the pilot

Note the referrals shown in the above figure at:

c=US
o=U.S. Government
ou=NIH
ou=FBCA
o=University of Wisconsin
o-dartmouth college pki lab

The RSA_FBCA Certificate Authority was also selected in the above figure and shows the object contents to the right, revealing the CRL, *crossCertificatePair*, and *caCertificate* attributes that would be utilized in path validation and discovery. An application requesting the associated data with this object would, starting at the top, receive one referral for *c=US*, then another referral for *o=U.S. Government*, then one more referral for *ou=fbca*. The DN of this object is: *cn=RSA_FBCA, ou=fbca, o=U.S. Government, c=US.*
Referrals within the RoD service may exist at any level as appropriate for the administration of the namespace being referred. This offers flexibility to delegate administration out to the true owners of the namespace in the "global" DIT space.

Open Issues for the Registry of Directories

- Resource discovery seems to be a daunting and, as of yet, unsolved problem. Configuring client software (email clients, web servers and so on) with a local (or remote) DSA that is part of the RoD service is not desirable. Software should have a mechanism for locating the global service only if there is not a locally defined service. Using DNS SRV records and even poking at the DNS hierarchy within the local domain seem appropriate until an RoD Service SRV record is located. This will allow the starting point to be locally defined and will provide an escape route from the global hierarchy for special arrangements or alternative hierarchies depending on the commercial climate of namespace providers. DNS security is not an issue here since the objects being located will be digitally signed and will be, therefore, "self-secure" with respect to the certificate being validated.

- It is not clear which approach is better: getting an object or searching for an object. If certificates contain *AIA* extensions that lead directly to the object associated with the issuer, this is clearly the best approach. However, not all methodologies associated with *AIA* are understood by all software. If one has to locate the issuer object, then how is that accomplished? Do we search on the DN in question or simply get it? Currently, there is quite a bit of discussion within the IETF-PKIX community as to which approach is best, and even discussions regarding the representation of a certificate in a directory. Do we provide new attributes that represent the contents of certificates and search those attributes (since X.509 certificates are stored as binary blobs) or do we search using special filters and matching items that allow for searching inside the binary X.509 blobs? These questions are not yet resolved, but the FBCA model will likely have to incorporate some new set of techniques to work with new, PKI-aware applications developed in response to the results of the IETF deliberations.

- The referral URI used in the smart referrals of the RoD must be pre-escaped, meaning the URI definition rules must be adhered to such that space characters must be translated to the %20 in the URI.

- Utilizing the LDAP standard port definitions of 389 or 636 simplifies the setup, since the firewalls usually are already open for other LDAP services. The X.500 chaining agreement setup requires special ports to be opened, which can lead to time delays and further security concerns by IT staff.

- In the case of X.500 directory chaining, chaining agreements are required in both directories. This requires a coordinated effort and substantial amount of administrative time to initially setup, and test proper chaining. The LDAP referral method was found to be easily set up and tested without the need for tightly coordinated effort and without the number of schema restrictions of chaining.

- Directory availability and security are critical issues associated with the deployment of this type of PKI. There exist many issues and solutions to yield high levels of both availability and security. The Federal model advocates use of a "border directory" which is essentially a public view of data originating from internal directories or databases that likely reside behind a firewall. There are other issues associated with directory-enabled applications that also require consideration that we will not attempt to discuss here. For more information,

refer to the Internet2 Middleware Initiative web site and the LDAP-Recipe at http://middleware.internet2.edu.

Border directories are specialized directories exposed to the world that contain a partial replica of proprietary information in the enterprise directory information tree of an institution or enterprise. This allows the border directory to supply public information to the bridge environment, thereby reducing the need for directory access controls and simplifying directory administration. The concept of the border directory is part of the FBCA architectural design to provide agency-based directories that expose only information needed for the FBCA to perform the path discovery and path processing. Institutions participating in the HEBCA will probably find this same concept to be a useful data security measure. Within the FBCA, the directories and border directories may be considered critical infrastructure systems and therefore require redundancy. This adds to the setup time and testing of the X.500 chaining agreements for both the bridge directory and the border directories. The HEBCA and the participating institutions could also be considered critical systems, but it is much easier to set up and test the smart referrals in the RoD than it is to ensure redundancy on all parts of the directory architecture.

- Firewalls and access controls to the directories within the institutions and the HEBCA will always need to be considered, although the referral mechanisms of the RoD simplify these issues because of LDAP's use of standard ports 389 or 636, as mentioned above.

- Anywhere that X.500 DSP is utilized, the administration of chaining agreements will require continuous checking, as well as synchronized time supplied, adding complexity to the infrastructure.

- Referral management will require institutional administrators to be aware of changes to the local directory tree that could affect RoD smart referrals. The LDAP Browser/Editor version 2.8.2 by Jarek Gawor was utilized for the creation of the smart referrals in the RoD as the native administration interface of the directory server was found to be cumbersome.

- Dartmouth College cross certified an Entrust Authority CA with the HEBCA. The Critical Path (previously PeerLogic) X.500 directory product was used with the Entrust Authority CA in this installation. The X.500 product needed to be upgraded to version 8A3 to resolve problems with directory chaining. The cross-certification exchange of certificates did not complete properly because of a still-unresolved incompatibility in the RSA product's response to the Entrust product. This issue was worked around by manually installing the cross certificates in the Dartmouth directory. A shadow DSA was created to avoid potential issues resulting from the manual certificate storage operation. Since additional hardware was not readily available to support the shadow DSA at Dartmouth, the team initially attempted to use a non-standard port for the shadow directory's LDAP connection. The Mitretek firewall, however, was only open for port 389 traffic. To work around this issue, the shadow directory was subsequently hosted on a server inside the Mitretek firewall. In addition, the update frequency for the CRL was extended to simplify synchronization with the shadow directory.

Desktop Service – Lexign ProSigner (E-Lock Assured Office)

ProSigner is a Public Key Enabled (PKE) application, allowing any PC-based documents to be digitally signed and encrypted. ProSigner is fully integrated with Microsoft Word, Microsoft Excel, and Adobe PDF enabling users to sign and encrypt documents quickly.

- Provides ease of use through a point-and-click tool bar that integrates with Microsoft Word, Excel, and Adobe Acrobat;
- Enables document encryption, so only specified people can view the content of a document;
- Provides centralized security including signing, encryption, verification, and certificate validation;
- Manages multiple signatures and creates an audit trail of documents as they flow through the signature cycle;
- Supports any X.509 digital certificate and works seamlessly with certificates issued by Digital Signature Trust, Entrust, RSA Security, VeriSign and others;
- Policy definition, enforcement and auditing insure simple workflow requirements.

Usage

ProSigner version 4.2 was utilized as a desktop service enabling the university partners to sign the Microsoft Word template PHS-398 form. To enable the signing, NIH translated its research grant workflow rules into Lexign signing policy that defines the two signatures be applied to the PHS-398 form.

The use of ProSigner, Microsoft Word and the PHS-398 allowed the researchers to fill out the electronic grant application form offline, wherever they were located. The researchers simply utilized Word to add the pertinent information into the PHS-398 document. Once all the information was completed, the researchers used the ProSigner controls in Microsoft Word to select their personal signing certificate to sign the application, then they attached it to an email to the institutional signing authority. The signing authority then reviewed the document, verified that it was signed by the researcher, and digitally signed it with his/her own signing certificate. Once both signatures were attached to the PHS-398, it was submitted to NIH simply by attaching it to an email and sending it to the OER email server.

The NIH recipient then opened the email and opened the attached PHS-398 with WORD and ProSigner. The NIH officer's ProSigner was configured to validate all certificates against a local CAM/DAVE validation service. When the PHS-398 was opened, signature validation was requested via the *Validate* API. If the certificate was within the trust list of the CAM, then standard ACES-level OCSP validation was performed. Since the certificates were issued from CAs not in the CAM trust list, validation was passed to DAVE and its configured default CA, the FBCA - HEBCA bridge infrastructure, to perform path discovery and path processing. When both certificates were verified through the CAM/DAVE service, the NIH officer then verified all the proper information was completed for the applications and disseminated it to referral and data entry.

As mentioned before, currently, ProSigner users must mange participating institutions' root certificates since the application still needs to see them in the Microsoft certificate store as trusted CA issuers in order to operate properly, even though it is CAM-aware.

Since Entrust software uses a proprietary client-side certificate store, it was necessary for Dartmouth's PKI Lab to use the Entrust-specific version of ProSigner to sign the sample NIH PHS-398 form with Entrust-generated certificates. Other pilot project participants used the Internet Explorer version. The now-current version of Entrust supports key/certificate export to the Microsoft Crypto-API, which should allow use of the IE version of ProSigner in the future. With these issues resolved, signatures and remote verification at NIH were successful.

Outstanding Desktop Application Issues To Be Resolved

The ProSigner version 4.2 utilized in the pilot project contained several problems that were worked around and should be fixed in later versions. Following is a brief list of these problems, followed by further explanation.

1. Explicit Trust in the CAM/DAVE validation without attempting to verify the CA within the local browser root store: ProSigner has been designed so that its certificate validation supported CRLs, OCSP, and CAM validation. In the case of CRL and OCSP based validations, the explicit validation of the CA required that the issuing CA root certificate was in the local browser root store and that the certificate being validated was valid within the validity period of the issuing CA's certificate.

2. The CAM response interpretation: Currently, the CAM validation API utilized by ProSigner returns several components to the *validate* API response message. Two of these parameters are important to the operation of the bridge-bridge model: the first is the CAM status code, which is the authoritative status of the certificate being validated and the second is the binary response message received by the CAM from the CA. Traditionally, this has been an OCSP response message from the issuing CAs validation service that may be used for long-term validation or archival proof of the certificate validation.

   The addition of DAVE means that an OCSP response message is not sufficient to contain the path information and its validation response to be stored with the document, allowing for the long-term interpretation of the document signatures. The addition of another signed binary response is an issue. Also, the signed

binary response from DAVE that encapsulates the path and validation information has not been standardized to provide a clear standard for developers to utilize. Although several IETF drafts provide options into which this information may be put, they are still subject to change. This is an area that needs further development. The first viable IETF Standard RFC to defined response information that includes path and validation information should be incorporated into DAVE. Of course, determining which IETF standard is viable can be problematic.

3. The CAM's application-to-CAM API has no security provisioning built into the *validate* API. This may be a limiting factor of the CAM's acceptance as a general validation service. An unexpected finding of the interoperability pilot project was the desire of researchers to use ProSigner and the CAM/DAVE validation service across institutional boundaries. This could allow a researcher to share critical research information securely utilizing ProSigner. The recipients then would need to verify the source of the signed documents electronically and would require that a public validation service such as CAM be deployed with new APIs providing security to the educational institutions.

4. Verification and Timestamp Issues. ProSigner stores audit information along with the signed document as signatures are verified. The timestamp of the verification is associated with the signature and with the document. However, if a document is signed and verified on 4/1/2002 at 12:01AM and then again on 4/15/2002 at11: 59PM, the timestamp is set to 4/15/2002 and not the original signing and validation date o 4/1/2002. Although not a direct issue for the pilot project, long-term audit information is highly important as proof of when a valid signature is applied to a document over time. It has been suggested that an initial validation timestamp and last validated timestamp should both be associated with digitally signed documents. This would facilitate creation of a minimal long-term archive of signed documents like the PHS-398.

5. When a document that has been signed and validated with the validation response stored with the document, then the document's signature hash is broken with a debugger, ProSigner does not report a invalid hash when using offline validation. This problem was reported as a defect to Lexign and should be fixed in the next release of ProSigner.

Policy Issues

The CAs that are part of the Interoperability Project issued certificates at the test level of assurance. To do business electronically, some form of policy needs to be created that addresses trust. Within the commercial X.509 PKI community, this is understood to require creation of a Certificate Policy (CP) in RFC2527 format that formulates the policies and procedures for issuing X.509 certificates at stated levels of assertion of identity and security. It also requires creation of a Certification Practices Statement (CPS) that describes in detail how the CA is to be operated to comply with the Certificate Policy. The degree to which a certificate user can trust the binding embodied in a certificate depends on several factors. These factors include the practices followed by the certification authority (CA) in authenticating the subject; the CA's operating policy, procedures, and security controls; the subject's obligations (for example, in protecting the private key); and the stated undertakings and legal obligations of the CA (for example, warranties and limitations on liability).

Beyond the strictly formal policy and procedures requirement, however, the organization issuing digital credentials needs to develop trust policies that address the questions implicit in establishing secure electronic business processes, for example: which credentials are good enough to satisfy trust requirements for a given transaction? What must be done to satisfy the business objectives, legal requirements, and culture of the organization issuing digital certificates?

Lessons Learned

Client Applications Client applications that rely on a Bridge CA have to know how to handle the certificate of each CA in the Bridge or to rely on the server-based certificate validation. Certificate repositories may not be accessible to the client applications. Client applications tend to not be able to handle complicated certificate hierarchies that may use cross certificates. Finally, client applications must be able to utilize the policy mappings of the different CAs in the bridge. This tends to be too much processing for client applications to handle.

Applications and Certificate Path Processing Server- based applications need to be able to handle the complexities involved to support certificate path processing and validation of the trust domains. In the implementation of the HEBCA, the CAM was enhanced to use an add-on discovery and validation engine (DAVE) module to facilitate certificate path processing and to validate the trust domains.

Trusted Servers Organizations are moving towards solutions that leverage trusted servers to do the hard work associated with certificate processing, rather than have the client do all the work. Hence solutions like CAM and OCSP or even plug-in modules such as DAVE are designed to perform discovery of a certificate path for processing to be used for validation.

Cross Certification In the Bridge approaches previously mentioned, cross certification can only be obtained with self-signed root certificates. Numerous commercial PKIs are designed such that subordinate CAs within the hierarchy are designated as the trust anchor for specific policies. This leads to the need to cross certify subordinate CAs with the bridge environment.

Directory Implementations In the Bridge approaches previously mentioned X.500 directory and border directory implementations need to further embrace LDAP. As mentioned in the implementation of the HEBCA, a registry of directories and smart referrals were utilized to address interoperability across a diverse community of directory technologies.

Using a Bridge CA The cost for many agencies or institutions to operate and run their own PKI is more than these organizations can budget or afford. These organizations need to consider that ***in order to use a BCA, the agency or institution must have their own PKI.*** An organization is oftentimes best served to utilize a trust model or PKI that is already in existence, such as ACES or a trusted third party (TTP).

Areas for improvement in the current application-to-CAM communications protocol: first, the lack of security within the protocol. Although not an original design requirement of the CAM, there are now use cases where the CAM and PKI implementation would benefit by the addition of authentication and confidentiality features to allow validation of the messages sent and received across the Internet. This would protect the transactions against denial of service (DOS) attacks and against replay attacks. Second, as noted above, the TCP/IP messages between the application and CAM utilize a nonstandard packet byte ordering, that is, Microsoft byte ordering instead of standard network byte ordering. Special attention should be paid to this when integrating applications to the CAM. The CAM source AA_TEST application, which is used for initial testing of a CAM installation, is a good starting point for integrators implementing the *validate* API.

Continuing Work

As more agencies and organizations adopt and participate in the BCA approach, more work needs to be done to ensure their success. Some of the immediate needs are identified below.

- Create a cookbook or document that identifies the minimal requirements and contents of the cross certificates and the directories; Given the lessons learned and discoveries made for all the components, a cookbook or document needs to be formally written that identifies the minimal requirements for certificates, directories and applications.
- Complete the cross-certification of Dartmouth by resolving the incompatibility with the RSA Keon CA product and Entrust;
- Continue to work with Verisign to complete the cross-certification of the University of California-Office of the President and University of Texas-Houston Health Science Center;
- Split the registry of directories to enhance performance across the infrastructure;
- Analyze and determine a more general solution for DAVE to perform directory discovery. It may be advantageous for DAVE to speak OCSP, for example;
- An investigation into multiple smart referrals to provide two different URIs to verify the enablement of redundancy for critical infrastructure cases. This would include teaching DAVE to try a secondary URI if the first did not return a response. If the AIA extension were mandated for any CA that wants to operate in a bridge environment, that would be a good beginning. Then, require an RoD entry for all participants of a bridge environment so the software would look at the AIA extension or the RoD to locate the issuer.

Summary/Conclusions

Given the disparate and many PKIs that are in use within the Federal Government and within other communities of interest, research institutions and Federal Government need to begin understanding how they can best leverage and work with the PKI environments that are underway. We need to come to an understanding and agreement that there will never be a single open PKI for everything. Rather, each major industry will determine its own solution, and the other industries that have a requirement to interoperate with other industries will need to figure out how to interoperate. An example is in the credit card world. The Federal Government did not define its own credit card standard. Rather, it evolved its payment processes to include the use American Express (AMEX) cards by Federal employees. The same is true for PKI. As an example, the higher education community will define its solution, and if the higher education community and the Federal Government want to interoperate, these two diverse communities will need to determine the best method of interoperability or continue to participate in the development of the infrastructure for each community.

Acknowledgements

Grateful appreciation for their participation in the pilot project and in the preparation of this manuscript is acknowledged to: Clair Goldsmith, University of Alabama at Birmingham; Jill Gemmill, University of Alabama at Birmingham; Keith Hazelton, University of Wisconsin-Madison; Eric Norman, University of Wisconsin-Madison
Robert Brentrup, Dartmouth College; Ed Feustel, Dartmouth College; Michael Gettes, Georgetown University; David Wasley, University of California Office of the President; Bill Weems, University of Texas – Houston Health Science Center; Mark Luker, EDUCAUSE; Steve Worona, EDUCAUSE; Deb Blanchard, Digital Signature Trust; Monette Respress, Mitretek Systems; Jim Fisher, Mitretek Systems; Ken Stillson, Mitretek Systems; Russ Weiser, Digital Signature Trust; Jack Kirivong, Lexign; Andrew Lehfeldt, RSA Security; Andrew Lins, Mitretek Systems; Cheryl Jenkins, Federal Bridge Certification Authority; Judy Spencer, Chair, Federal PKI Steering Committee.

References

LDAP-Recipe: A Recipe for Configuring and Operating LDAP Directories, Michael R Gettes, Georgetown University, February 2001 & April 2002

Bridge Validation Authority, Ambarish Malpani, ValiCert, Inc., December 2001

Planning for PKI, Best Practices Guide for Deploying Public Key Infrastructure, Russ Housley, Tim Polk, John Wiley and Sons, Inc., 2001

Federal Grant Streamlining Program, Department of Health and Human Services Response to RFI-4-02-HHS-OS, Digital Signature Trust, February 2002

Final Report – Phase 1, Prepared for National Institutes of Health (NIH) Office of Extramural Research (OER), Under Contract No. GS00T99ALD0006, Digital Signature Trust, February 2002

Report of Federal Bridge Certification Authority Initiative and Demonstration Electronic Messaging Association Challenge 2000, October 2000, Mitretek Systems

PKI, Implementing and Managing E-Security, Nash, Duane, Joseph, and Brink, RSA Press, McGraw Hill, 2001

Educause Review, "A "Bridge" for Trusted Electronic Commerce," Mark A. Luker, January/February, 2002, Volume 37, Number 1

The Evolving Federal Public Key Infrastructure, Federal Public Key Infrastructure Steering Committee and Federal Chief Information Officers Council, June, 2000

Internet2 Middleware Initiative Web Site, http://middleware.internet2.edu, Middleware Architecture Committee for Education (MACE), et. al.

Experiences Establishing an Experimental International Coalition Public Key Infrastructure

By Glenn Fink *(Naval Surface Warfare Center, Dahlgren VA, finkga@nswc.navy.mil),* Shawn Raiszadeh *(Lockheed Martin Corporation, Fairfax VA, US, shawn.s.raiszadeh@lmco.com), and* Timothy Dean *(QinetiQ, Ltd., Malvern, Worcestershire, UK, tbdean@qinetiq.com)*

Abstract

Research and testing teams from the US and UK participated in joint design and testing of a Public Key Infrastructure (PKI) for international military coalition operations. We planned the design and testing in five phases from an initial PKI interoperability study through design of a second-generation PKI based on web services. Each design phase is followed by a testing and demonstration event to verify and recommend improvements to the system designed.

The paper opens with a description of the unique set of requirements an international military coalition must levy on its PKI. Next, we briefly describe each design and testing phase to give the reader a sense of context. This paper documents experiences with PKI technology that our research group had during the two most recent testing phases, II and III. We have included design and test-structure information for these two phases and highlighted our lessons-learned. We conclude with our current plans for future phases of the study. The intended audience for this paper is experienced PKI users, vendors, and researchers. We hope our findings and recommendations will be useful to the scientific community as we attempt to enable solutions complex problems through technology.

Keywords: Public Key Infrastructure, PKI, Security, International Military Coalition, Authentication, Nonrepudiation.

1.0 Introduction

The Virtual Operations Network (VON) project is an international military effort to facilitate management of naval coalitions involving forces from many nations. Teams of researchers from the UK (QinetiQ in Malvern and Portsdown West) and the US (Lockheed Martin (LM) in Fairfax, Virginia, and Naval Surface Warfare Center (NSWC) in Dahlgren, Virginia)[i] joined together to form our VON PKI research group.

1.1 Unique Requirements of Coalitions

Coalitions in operations like Desert Storm and East Timor have demonstrated that traditional solutions for communications among a diverse group of coalition partners require an unsatisfactory amount of time and effort to establish and maintain. Some of the communication problems arise from equipment and software incompatibilities. Other communication issues come from the inability to trust once communications are established. Part of the VON effort involves establishing a degree of trust to facilitate information-sharing among coalition partners that are not traditional allies or may even be traditional adversaries. This project is complicated by the dynamic nature of modern coalitions where members may join for a relatively short period of the overall operation and may change roles during the operation. Nations participating in international coalitions come from a broad spectrum of technological ability---from low-tech, third world nation-states to technological superpowers. To level the playing field, nations with technology advantages may have to provide "throw-away" PKI components and services to their disadvantaged partners. While it is likely that the US or one of its high-tech allies would host some of the coalition PKI, it is essential that any nation, including the PKI hosting nation, be able to walk away from the coalition at any time without leaving indispensable personnel or sensitive equipment behind to maintain the PKI. Any equipment that must be left behind must be highly tamper-resistant to prevent technological espionage.

The coalition PKI should be accreditable by various nations. This implies that nations can be assured that none of their national secrets will be released into any associated coalition without the nation's explicit consent. Accreditation generally requires presenting evidence that the risk is sufficiently low to make it worth the information gained. Accreditation also influences the amount of time taken to establish a coalition. The coalition PKI may be able to reduce this

delay by selecting standard or pre-approved hardware and software packages.

Because the partner nations are quite independent, a coalition PKI must have decentralized management of trust. Some partners may already have national PKIs, and most have national secret networks. Each of these partner-nations will want full access to information from the coalition PKI but tightly control the flow of national data into the coalition.

In military operations of all sorts, timely authentication and nonrepudiation are mission-critical requirements. PKI clients must be able to determine the validity of digital signatures quickly with a high degree of certainty that the status is up to date. Hardware tokens are envisioned for this application so that nonrepudiation may be more reliably achieved. For timeliness, we plan to require revocation windows of less than an hour.

The planned coalition PKI will run on shipboard platforms communicating over High Frequency (HF) or Ultra-High Frequency (UHF) radio links with extremely limited bandwidth and intermittent connectivity. In each battle group there will be one or more "gateway" ship(s) with satellite communications (SatCom) capability that will connect battle groups to the shore-based Network Operations Centers (NOCs). The NOCs may be nationally or internationally owned and will interconnect via secure, fixed links. PKI applications that cannot operate correctly under circumstances of intermittent connectivity and low bandwidth need not apply. Figure 1 is an overview of the communications concept used by VON.

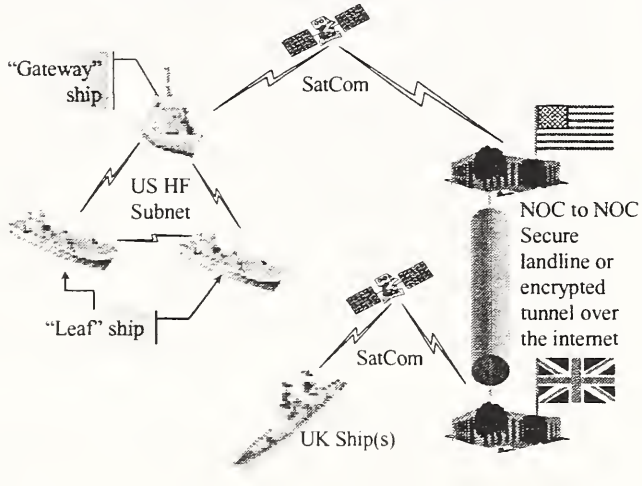


Figure 1: VON Communications Concept

1.2 The Experiments

Our research group conducted experiments with two separate PKIs during Fall 2000 and Summer 2001. During the test periods, laboratories at four geographically separated sites hosted simulated tactical platforms (ships and NOCs). The platforms were interconnected by dial-up ISDN lines simulating radio frequency (RF) transmission speeds. This effort was a step towards deploying PKI technology in a multi-national at-sea trial in 2002.

The VON PKI effort can be divided into five phases of experimentation leading toward eventual deployment in operational environments:

Phase I	Lockheed PKI Interoperability Study
Phase II	UK-US Joint PKI Interoperability Test
Phase III	UK-US Joint Proof of Concept
Phase IV	Multinational At-Sea Prototype Trial
Phase V	2nd Generation PKI: Web Services

Currently the project has completed Phase III and some initial testing for Phase IV. We will complete Phase IV during Summer 2002. This report documents experiences our joint research group had while fielding experimental PKIs during Phases II and III and will outline plans for the following phases.

Phase I was conducted by the LM team according to requirements defined by NSWC and the Office of Naval Research (ONR). LM evaluated five PKI certificate management systems (CMSs) and two Lightweight Directory Access Protocol (LDAP) directory-server products, given the requirements we had defined. The team simulated a three-nation coalition PKI using three different PKI vendors. This study is documented in [1] and is not further expanded upon here, but findings from it form the basis for the phases that followed.

Phase II testing occurred in the Fall of 2000 (September through early December), with the focal testing events conducted 13-17 November. The purpose of Phase II was to test the work done in [1] in a truly international setting. This was our first bilateral experiment in the PKI school of hard knocks. Two Certification Authorities (CAs) were set up, Netscape Certificate Management System in the US and Baltimore UniCert in the UK. We achieved limited PKI interoperability by maintaining a trusted lists of CAs in the clients. Parties successfully exchanged and verified signed and encrypted e-mail (sans attachments), and, with mixed success, visited each others' SSL-secured web pages. We also established secure network tunnels (via Internet Protocol Security (IPSec)) but used only static keying without automated enrollment via PKI.

Phase III testing was conducted in late summer of 2001 (July through August). The purpose was threefold:

1. To centralize trust management at the national level (as opposed to each user managing trust lists individually),

2. To reduce risk of component or system failure during the planned at-sea trial during Phase IV, and
3. To incorporate hardware tokens (smartcards, etc.) for end user credential storage.

Phase III testing was focused on cross-certification, exchange of S/MIME e-mail with attachments, and revocation testing. Both nations setup their own root CA (the US used Entrust, and the UK used Baltimore) and the teams cross-certified the two PKI domains. The 2001 testing period is believed to have been the first time that government/military organizations from different countries successfully established trust between independent national PKI domains using different vendor products. Participants at four separate sites exchanged, validated, and read digitally signed and encrypted email messages, proving the interoperability afforded by the coalition PKI.

The Phase IV at-sea trial will exercise the PKI configuration established and refined in earlier phases. This phase may involve more nations and will be on actual rather than simulated shipboard platforms. This phase should be completed by the end of this summer.

Phase V will incorporate the knowledge gained during the at-sea trial and attempt to define a middleware prototype that will standardize the application program interface to the coalition PKI regardless of the underlying PKI structure. This phase will rely heavily on Extensible Markup Language (XML) technologies, especially XML Key Management Specification (XKMS) and Security Assertion Markup Language (SAML). Work beyond this phase will probably involve further interfacing the coalition PKI with national PKIs and the multitudes of policy issues that arise from these interfaces.

2.0 Phase II Experiments

2.1 Objectives of Phase II

The overall objective was to set up a simulated coalition communications infrastructure and PKI to test interoperability results obtained during the study done in the previous phase. The supporting objectives of this experiment were:

1. Build a simulated RF shipboard network using ISDN links and RF simulators.
2. Establish TCP/IP (e-mail) connectivity.
3. Standup national PKIs and establish coalition trust via trust list.
4. Exchange signed and encrypted e-mail.
5. Test mutual web-server access and SSL.
6. Test publishing certificates to an LDAP directory and test remote LDAP replication.
7. Experiment with certificate issue, revocation, reissue, and CRL distribution.

2.2 Testbed Configuration for Phase II

The testbed for Phase II consisted of a wide area network (WAN) of computers using ISDN as the backbone. Figure 2 shows the coalition communication concept for this phase. Four simulated ship platforms from fictional countries: Green (San Francisco, CA, US), Red (Portsdown, UK), Blue (Dahlgren, VA, US), and Orange (Malvern, UK) communicated over simulated radio links.

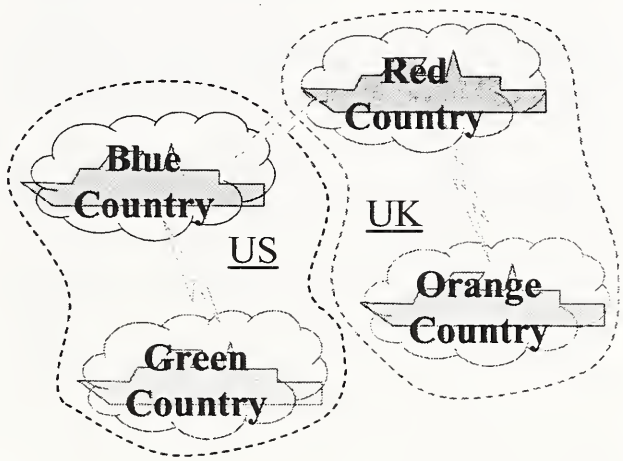


Figure 2: Phase II Coalition Structure

No NOC was used although network operations were concentrated in Red and Green. Blue and Orange were the PKI providers for the exercise. The US hosted the following services:

- CA/RA: Netscape Certificate Management System v4.1.5 (NT)
- LDAP Directory: Netscape Directory Server v4.1.5 (NT)
- SSL-compliant Web Server: iPlanet Web Server v1.0 (Solaris)
- Web Clients: Netscape Navigator Clients v4.7.5 (NT/Solaris)
- Mail Server: Netscape Messaging Server v4.1 (Solaris)
- S/MIME-compliant Mail Client: Netscape Communicator Messenger v4.7.5 (NT/Solaris)

The UK hosted the following services:

- CA/RA: Baltimore UniCERT Certificate Management System v3.0.5 (NT)
- LDAP Directory: ISOCOR Directory Server v2.3r1 (LDAP)

- SSL-compliant Web Server: MS Internet Information server v4.0 (NT)
- Role-Based Access Control: WebMACE v1.1 (NT)
- Web Clients: MS Internet Explorer v5.5 (NT)
- Mail Server: MS Exchange v5.5sp2 (NT)
- S/MIME-compliant Mail Client: MS Outlook 98 (NT)
- Firewall/Mail Guard: SWIPSY (Trusted Solaris)

2.3 Testing Conducted and Results from Phase II

We followed a pseudo-military scenario that involved a coalition forming, performing a mission, evolving, and disbanding. From the scenario and the objectives we derived the following our technical PKI events of interest. Each national CA sent the other CA its self-signed certificate for the end users to add to their trusted list. Then US CA issued "coalition" certificates to UK users and vice versa. We tested these certificates by exchanging signed and/or encrypted email and by visiting each other's secure web sites (via Secure Sockets Layer (SSL) v2.0 using both server-side and client-side authentication). After using the certificates we revoked them and attempted the same tests with the revoked certificates to make sure that revocation ended the trust relationship.

Overall success was achieved in most areas. The most notable deficiencies were caused by incomplete implementation of PKI awareness in the client applications.

2.3.1 Problems Encountered in Phase II

There were numerous bumps along the way and a few failures of minor objectives. This section is a collection of our problems grouped according to the software unit where the problems were manifested.

iPlanet Directory Server—We learned that the Directory Information Tree (DIT) structure is tightly coupled with working of the CA and other PKI servers. We originally underestimated the degree of coupling and could not publish certificates to the directory. We were forced to do several directory naming scheme reworks to make certificate publishing work.

Even after fixing the directory problems, we were unable to publish certificates from Netscape CMS via SSL to the iPlanet directory consistently. Either the directory or the CA seemed very buggy on this point. Once we got it working we dared not touch it. This behavior would not be acceptable in an operational environment.

Netscape Certificate Management System (CMS)—SSL server-to-server communications never worked for the Netscape Messaging server. Although certificate enrollment for the Messaging Server seemed to work well, the subsequent use of the certificates in SSL communications did not work. This implied no secure transfer of e-mail from one mail server to another, no secure Internet Mail Access Protocol (IMAP), and no secure access to directory data from the directory server. We were however using IPSec to bulk-encrypt all traffic so these issues were not immediate problems.

Netscape CMS seemed to be quite brittle requiring reinstallation numerous times. Simple changes (e.g., IP addresses of servers, etc) could render CMS useless until it was re-installed.

Netscape Communicator clients in general—The inability of client software to reliably check certificate status was a major problem. In Netscape Communicator's web and e-mail clients, revoked SSL server certificates would not raise any alarm until a CRL was explicitly downloaded into Communicator from Netscape CMS's end-user web interface. The button used to download a CRL to Communicator apparently is only available when visiting the client web portal of Netscape CMS. Once a CRL was downloaded into the client, revoked web site certificates generated the appropriate warning, and mail users could not use expired certificates for signing messages. All this was expected and proper, but after downloading a CRL, the client must manually reload a new CRL before the old one expires or be unable to use any SSL or S/MIME facilities. This then prevents the user from downloading a new CRL! This behavior is clearly counter-productive. Some flexibility to allow a user to participate in SSL transactions even if the local CRL has expired would be helpful. Another possibility would be to automate the CRL download process. For our application, CRL lifetimes were very short (fifteen minutes) so we were forced to ignore CRLs altogether to avoid the continual annoyance of downloading new CRLs manually.

Netscape did provide a Personal Security Manager (PSM) plug-in for its Communicator 4.73 client. This plug-in would allow the use of Online Certificate Status Protocol (OCSP) to verify certificates presented to the client. However, PSM was so buggy and caused so many crashes that we decided not to use it. Since there were at that time no other freely available OCSP-aware clients we elected not to use OCSP.

Netscape Navigator web client—Users of both national PKIs were able to register for and receive certificates from the web portal of the foreign CA, but US users were inexplicably unable to import the UK's CA chain into Navigator's trust list. Numerous creative attempts failed, although the UK was unable to duplicate the incompatibility. The reason for this

problem was never discovered and may have been caused by influences outside either the Baltimore CA or Navigator.

By default, Navigator expects the Distinguished Name (DN) of an SSL server's certificate to follow a specific format. A certificate's DN must have the common name (CN) of the server as its first element, and the CN must match the server's Domain Name System (DNS) name exactly. Using a more human-readable CN (e.g., "`CN=Stanleys Web SSL Cert`") in the certificate generated name mismatch errors in the browser every time the web site was visited. This makes maintaining a large number of certificates unwieldy because they are not readily identifiable by humans. Supporting the Subject Unique Identifier field or allowing the CN to be free form would help.

The UK certificates generated by Baltimore CA and issued to US users could not be used to sign messages or validate signatures. The problem appeared to stem from the inability of the US's Netscape clients to import the UK's trust chain. Reasons for this inability are unknown.

Role-Based Access Control (RBAC)—Hosts at all sites were able to access the native web interface of NSWC's Netscape CMS CA using SSL with mutual authentication. US users with certificates issued by the UK were able to access UK home-grown websites requiring presentation of a client certificate. But US Netscape users were unable to properly access UK pages controlled by the RBAC software, WebMACE. The reasons for this are not known. The US did not attempt to protect any of its home-grown websites via PKI because it was not immediately apparent how to implement this and testing time was limited.

Firewall and Guards—The UK deployed a coalition guard on the periphery of its national network. The purpose of the guard was to prevent leakage of sensitive information from the national network into the coalition. Unfortunately, the guard did leak e-mail addresses with names that revealed the underlying structure of the UK network (e.g., the domain name indicated which platform the user was located on). Eventually this guard would also be a PKI signature proxy. The guard would replace the signatures of individual UK users with the guard's signature so that the internals of the national PKI would be shielded from the coalition. This feature has not yet been implemented.

General PKI Instability—The US lab at SPAWAR Systems Center—SanDiego, California (SSC-SD) provided Radio Frequency (RF) link simulation for the exercise via AdTech SX-12 RF simulators installed at their site. The RF simulators were intended to provide realistic bandwidth limitations and error characteristics to emulate the HF radio and Satellite communication links that will be used in at-sea scenarios. Unfortunately, we were unable to simulate RF links in Phase II because the PKI was never stable enough to be stress tested.

2.3.2 Accomplishments of Phase II

Out-of-band resources were established for exchange of administrative data among experimenters. These resources included ftp, web, and chat servers, Voice-over-IP (both in the clear and over IPSec), and teleconference phone calls. The latter two were indispensable in overcoming the PKI and networking obstacles we encountered.

We used an IPSec encryption mesh between each of the four sites using pre-shared keys and 56 bit DES. This allowed us to assure the security of the experiments without relying exclusively on PKI.

We published certificate and user information to US and UK LDAP directories accessible to all. There were no problems with users registering or retrieving certificates, except for the US's problem attempting to import the UK's trust chain. Thus, users at all sites were able to exchange signed and encrypted email using at least US-issued certificates.

The US deployed a Network Time Protocol (NTP) server for eventual use as a trusted time server for non-repudiation. The NTP server was, however, only used to synchronize clocks in order to preserve the correct order of receipt of mail messages from all sites.

2.4 Lessons Learned in Phase II

Many general lessons were learned about the issues of PKI deployment:

- PKI interoperability was, at that time, an afterthought among vendor products we tested.
- PKI-enabled applications were rare and limited in their implementation of PKI features such as certificate status checking.
- PKI was much harder than we thought, and implementations were not at all robust. The brittleness of all the PKI implementations tested meant that they could not be relied upon for operational use at that time. We learned that the foundation of workable PKI is the directory. The format of information stored in national border directories is crucial for all parties to agree upon.
- Constant coordination was required to bring up a coalition PKI.

The state of PKI technology did improve over time as did our understanding of it. We had much more success in the next phase of experimentation.

3.0 Phase III Experiments

3.1 Objectives of Phase III

The goals of VON Phase III were threefold:

1. To centralize trust management at the national level,
2. To reduce risk of PKI component or system failure during the at-sea trial (Phase IV) by defining common minimum architecture requirements and baselining the configuration for the at-sea trial, and
3. To incorporate hardware tokens for end entities' certificate storage and presentation.

Testing was focused on cross-certification, exchange of S/MIME e-mail with attachments, and revocation testing (both end-entity and cross-certificate). Web and other services were de-emphasized in favor of solidifying the PKI itself. As the PKI evolves, we anticipate adding other services.

3.2 Testbed Configuration for Phase III

The testbed for Phase III (shown in Figure 3) simulated five platforms located at four geographically separate sites: two national NOCs, one in the US and the other in the UK, a US gateway ship and two US leaf nodes. The US NOC was physically split between two locations. The LM site provided the PKI servers in its half and NSWC provided DNS and mail servers and served as a network hub. Both US sites hosted LDAP servers for performance, redundancy, and fail-over reasons.

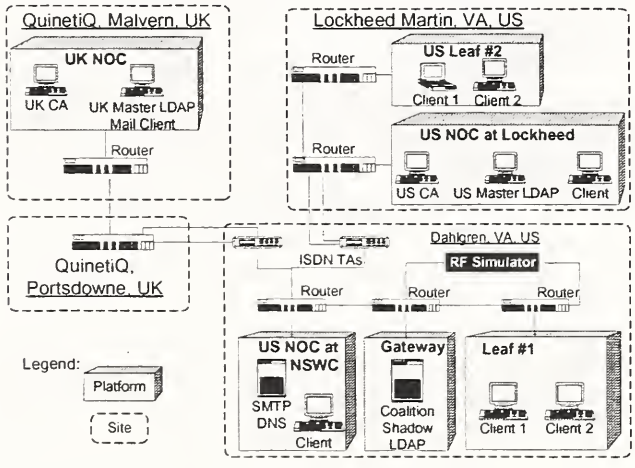


Figure 3: Testbed Configuration for Phase III

The US team developed a proposed coalition PKI architecture document [2] that specified interface standards that PKI products used in the demonstration must support to achieve the minimum acceptable level of interoperability. Only commercial PKI products were used in the demonstration. The proposal was accepted by the UK with minor changes. In particular, it was agreed that secure and trusted collaboration would be achieved by cross-certification between the US and UK CAs over a single ISDN 64 Kbps channel that emulated throughput expected during the at-sea trial in the following phase.

The configuration below was outlined in the proposal to achieve secure communications and mutual trust between US and UK systems. The boldface items represent changes from the Phase II configuration. The results of the testing confirmed this as the baseline configuration for Phase IV.

The US hosted the following services:

- CA/RA: **Entrust v5.1.1** (NT)
- LDAP Directory: Netscape Directory Server v4.1.5 (NT)
- Mail Server: Netscape Messaging Server v4.1 (Solaris)
- S/MIME-compliant Mail Client: **MS Outlook 2000 (NT) with Entrust Express plug-in**

The UK hosted the following services:

- CA/RA: Baltimore UniCERT Certificate Management System **v3.5** (NT)
- LDAP Directory: **Border: iPlanet Directory Server v4.1.5 (NT); Internal: Novell DirXML 1.0 and eDirectory**.
- Mail Server: MS Exchange v5.5sp2 (NT)
- S/MIME-compliant Mail Client: MS Outlook **2000** (NT) **with Baltimore MailSecure**.
- Mail Guard: SWIPSY (Trusted Solaris)

3.2.1 Certification Authorities

Both fielded CA products supported cross-certification as defined in RFC 2587 [3]. To ensure the security of the certificate exchange, an "out-of-band" process (voice telephone) was used to verify the thumbprint of a cross-certificate request.

Scalability problems arise when establishing and maintaining trust relationships solely via cross-certification. A total cross-certification trust model implies a mesh topology with $O(n^2)$ cross-certificates to be issued and maintained. However, we assumed that the number of relationships is manageable given our small demonstration coalition. We chose cross-certification as a potential step toward an bridge CA trust model that would require only O(n) cross-certificates.

To avoid the undesirable side-effects of transitive trust, we specified that the *pathLenConstraint* field of the Basic Constraints extension would be set to zero as described in RFC 2459 [3]. Transitive trust is indirect

trust between PKI domains that can be established either knowingly or inadvertently. For example, suppose CA_1 trusts CA_2 and CA_2 trusts CA_3. If after this CA_1 now trusts CA_3 then transitive trust exists. Transitive trust management via name constraints, etc. was not used.

Risk reduction tests conducted prior to Phase III found that a number of CA configuration options had to be agreed upon in order to ensure client application interoperability. Therefore, the CA products for both countries were required to support the following configuration:

- 160-bit SHA-1 hash for authority and subject key identifiers
- X.509v3 certificates with the following standard extensions:
  - *keyUsage*
  - *authorityKeyIdentifier*
  - *subjectKeyIdentifier*
  - *cRLDistributionPoints*
  - *subjectAltName* (containing the subject's email address per RFC 822), and
  - *basicConstraints*.
- All other extensions marked as non-critical.

The US installed its CA at the LM NOC site and published CA information including CRLs, CDPs, ARLs, and certificates to the collocated US master directory server. Likewise, the UK installed its CA at the UK NOC site and published CA information including CRLs, CDPs, ARLs, and certificates to its master directory server

The US issued two identity certificates to each US users one for encryption and another for signing. Private keys for the signing and encryption certificates were generated on smart cards; but only encryption private keys were escrowed at the CA. The UK issued certificates to its users similarly, except that they used soft tokens and did not escrow any keys.

The UK and US then exchanged copies of their respective Root CA certificate both in native format and in a PKCS #10 signing request via in-band e-mail Once exchanged, both parties verified the thumbprints of the PKCS #10s over the telephone. These tasks helped us to understand the impact of the following problem-domain issues: the effort involved in using a secure method of exchanging the PKCS#10 requests, the amount of work needed to configure cross-certification, and the time required to set-up a root CA for coalition operations.

3.2.2 Directory Service

The US and the UK agreed to standardize on the iPlanet Directory Server v4.1.5 as the border directory service implementation. The agreement to use a common directory product avoided several technical and implementation issues, most notably directory replication. Surprisingly, although iPlanet directory server v5.0 was available to us, its replication function is not compatible with version 4.x of the same product. Since the US did not have the resources to test interoperability between Entrust and the v5.0 directory, the UK decided to use the older directory server for its border directory. Directory interoperability is certainly an area where standards are lacking. Emerging standards and products for directory-to-directory interoperability such as LDAP Duplication/-Replication/Update Protocols (LDUP)), Directory Services Markup Language (DSML) and Novell's DirXML are possible solutions. The UK demonstrated the use of Novell's DirXML internally as an automated directory synchronization agent between iPlanet Directory Server v4.1.5, Microsoft Exchange and Novell eDirectory.

We used centralized-partitioned (a.k.a. hub and spoke directory) topology for our directory replication scheme. Communication between the UK and US directories occurred through the US hub and its UK replica. In a coalition environment where connectivity is sporadic and throughput limited, the hub and spoke topology was best for scalability, redundancy and manageability. Each coalition member provided a read-only directory replica of local security information to the hub directory. The hub directory provided a complete read-only replica to each spoke, thus allowing each coalition member a complete local view of the coalition. Figure 4 depicts an idealized hub and spoke directory topology in a coalition environment.

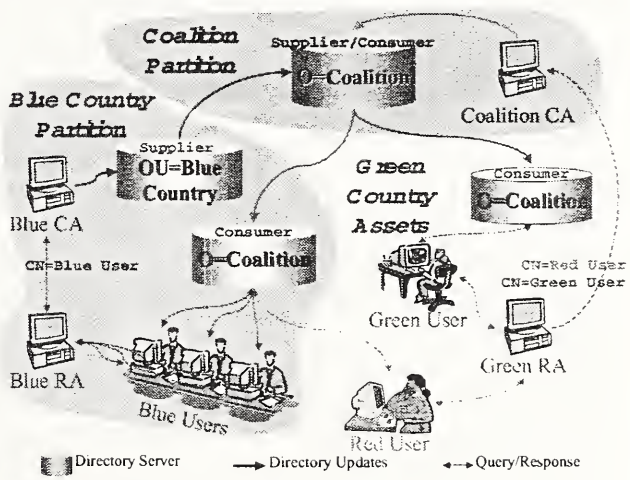


Figure 4: Hub and spoke directory topology

In the figure, Blue country supplies its own master directory information to the coalition and receives back a re-mastered copy of the entire coalition directory (including entries for Green country and the Red user).

This model allows for countries to participate without supplying a master directory or a CA/RA. Replication agreements are minimized while redundancy is preserved. Any country providing a master directory server and a coalition shadow may take over as the coalition hub in case the original hub is damaged or lost. Note that the Coalition CA in the diagram need not exist at all and the coalition directory may be hosted by any partner nation.

Our implementation of hub and spoke topology is shown in Figure 5. Both parties agreed on a directory schema including DIT, added PKI attributes, etc. The US configured two directory servers: one as a US Replication Hub (US-1), one as a US master replica (US-2). Then, the US configured a simulated gateway ship computer (US-3) as a read-only replica of the US Replication Hub (US-1). The US set up replication from US-2 to US-1 (replication path RP1); and from US-1 to US-3 (RP2)

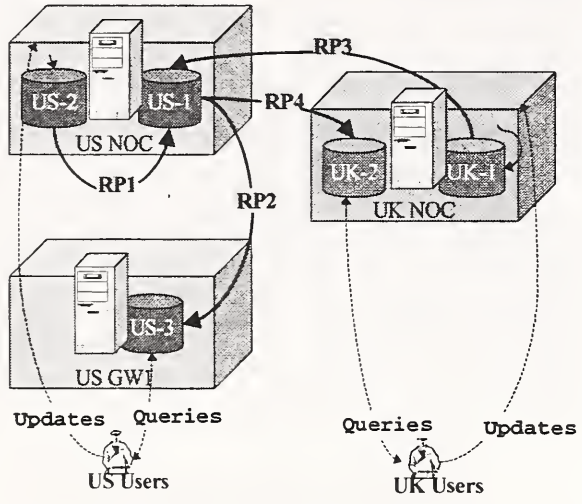


Figure 5 Coalition directory replication topology

The UK also configured two LDAP servers: one as a UK master replica (UK-1), and one as a read-only replica (UK-2) of the US Replication Hub (US-1). The UK collaborated with the US to set up replication from UK-1 to US-1 (RP3). Finally, the US collaborated with the UK to set up replication from US-1 to UK-2 (RP4).

Replication path RP2 demonstrated replication over intermittent links or unreliable connections as may happen between the Gateway ships and NOCs on the shore. Replication paths RP3 and RP4 demonstrated replication over a reliable link, as expected between the two NOCs in the following phase and in deployment.

Replication was achieved using LDAP bind IDs and passwords, rather than certificates for this phase. Replication over SSL will be used in later phases. All replication was server-initiated (push) rather than consumer initiated (pull).

3.2.3 Applications

Secure (S/MIME) email was the touchstone application used to test the Phase III coalition PKI architecture. S/MIME provides authentication and integrity via digital signatures over message hashes, and data confidentiality via encryption. Both the US and UK used Microsoft Outlook 2000 for encoding and decoding of S/MIME messages. We used plug-ins for Microsoft Outlook 2000 to provide trusted exchange of messages leveraging coalition cross-certificates. The US used the Entrust Express plug-in and the UK used Baltimore's MailSecure product for verifying trust between the cross-certified PKI domains. The plug-ins enabled Microsoft Outlook 2000 to check user certificate status by downloading Certificate Revocation Lists (CRLs) from a local directory replica.

3.3 Testing Conducted and Results from Phase III

As detailed above, before beginning testing in this phase we took pains to define minimum interoperability standards. This precaution resulted in a much smoother testing period. We tested by transmitting unsigned, signed, encrypted, and signed-encrypted e-mail messages both with and without attachments during the test phase. Our results demonstrated working path validation and discovery. We also tested revocation by sending signed e-mail between realms after revocation of a user certificate or a cross-certificate.

We used network analyzers to record email and LDAP traffic and verify system correctness. The recorded traffic was analyzed to ensure email messages were indeed digitally signed and/or encrypted when applicable. The recorded traffic was also used to ensure proper workflow for certificate validation. Figure 6 depicts the certificate validation logic the US Entrust Express client used to validate a digitally signed email message from a UK user.

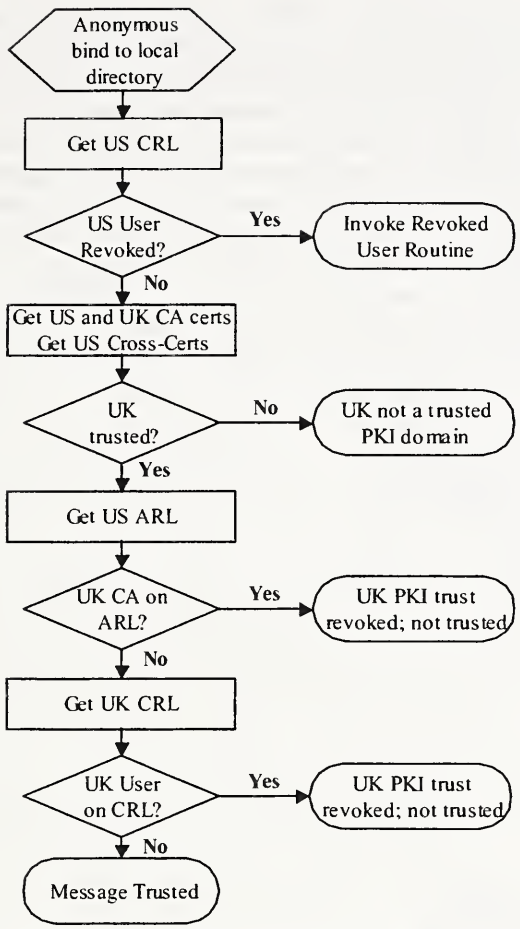


Figure 6: Client-Side Signed Email Validation

3.3.1 Problems Encountered in Phase III

This section presents major problems encountered during Phase III testing. The problems are organized according to the products where they manifested themselves. We have explained each problem to the extent of our forensic abilities, but because of the inherent complexity of PKI, formal attribution of problems is not possible. We hope these records will be useful to the vendors and to new PKI users as they field their own PKIs.

Entrust CA—Insufficient fields were present in the PKCS #10 cross-certification request from the US's Entrust CA for a correctly formatted cross-certificate to be produced by Baltimore's UniCERT CA. In particular the Subject and Authority Key Identifier fields appeared to be missing. These fields are essential in correct trust path building. Cross certification was successfully achieved using the US root self-signed certificate instead of the PKCS #10 message. The missing fields were manually added to the cross-certificate by the UK's CA operators.

Baltimore UniCERT CA–The UK found it difficult to achieve reinstallation of Baltimore's UniCERT CA without reinstalling the machine's entire operating system. It is very important to establish correct CA configuration throughout the coalition at install time.

A few other minor incompatibilities were also discovered between MailSecure and UniCERT in trust path building using cross-certificates.

Entrust RA – After revoking a user through Entrust RA, the CRL must be manually created and pushed to the directory via the Entrust RA interface in order for the latest CRL to be immediately published to the directory. Once again, a publish-and-subscribe CRL mechanism would be ideal.

Entrust Express Outlook 2000 plug-in – Outlook 2000 must be installed in "Corporate Mode" in order to support Entrust Express. Installing in "Internet Mode" produced inconsistent results and strange errors when doing signature validation.

When trying to add a user to the Entrust Address Book from a Directory Search, Entrust Express generated an error[ii]. Entrust assumes that the certificate being added to the Entrust Address Book from the LDAP Directory is an encryption certificate (e.g. the *keyUsage* value is "Key Encipherment"). Entrust does not publish digital signature certificates to the directory because they are sent in every S/MIME of digitally signed message. If the *userCertificate* attribute for a user in the directory contains multiple certificates, the first or only certificate must be the user's encryption certificate. For Entrust Express, the ideal would be for each user entry of the directory to contain only one certificate: the users' current encryption certificate. To avoid problems, any revoked certificates must be manually remove from the directory and the first certificate entry must be a valid encryption certificate.

Entrust Express was unable to validate the certificate chains with heterogeneous signature algorithms. VON's policy specified DSA key pairs, since DSA was the preferred US and UK Government algorithm. When the RSA algorithm became public VON's requirement changed to using RSA key pairs since RSA has wider usage. The UK had installed its Baltimore CA using a DSA self-signed certificate prior to the policy change and preferred not to reinstall the CA in order to comply. Instead, the UK team decided to issue all end-entity certificates with RSA key pairs and leave the self-signed root certificate alone. Unfortunately, we found during testing that Entrust Express displayed an error[iii] when opening digitally signed messages received from the UK since the sender's CA certificate public key algorithm was different from the public key algorithm used by end-entity certificates. The work-around was to ensure the same public key algorithm is used for CA and end-entity certificates. To fix this problem during the testing events, UK had to reinstall its entire CA to change the CA's self-signed certificate to use the RSA

algorithm. All UK user certificates were then issued with the RSA public key algorithm. In general we determined that the Entrust plug-in could handle homogeneous RSA or DSA algorithms all the way up the chain, but cannot validate certificates whose validation paths use mixtures of DSA and RSA signing algorithms.

iPlanet Directory Server – Occasionally, replication agreements did not result in automatic replications when the directory service in question functioned as both a supplier and a consumer of the same tree (e.g., coalition mirror directories that also replicated themselves to other directories).

iPlanet Messaging Server – The Messaging Server must be able to write to the directory root organization (e.g. "o=coalition.mil") where it pulls email-related information. Otherwise the Messaging Server will fail to start Simple Mail Transfer Protocol (SMTP) services. The Messaging server uses the root entry to store certain administrative data. If the root entry is not writeable, the SMTP service cannot start, but other services may. The US had to constrain directory replication to its Messaging Server to the "ou=United States, o=coalition.mil" subtree to work around this limitation.

Entrust & Baltimore Mail Client Plug-ins—By default, Entrust and Baltimore cache Certificate Revocation Lists (CRLs) and Authority Revocation Lists (ARLs). It was therefore necessary to restart the clients to download the latest CRL from the directory when conducting revocation tests. This is not a shortcoming; both retrieve CRLs from the directory when the most recent CRL expires. Unfortunately, we found no way to push an interim CRL containing newly revoked certificates to the clients before the next update time. Turning caching off produced excessive CRL network traffic, and caching time could not be set below four hours for Entrust because that is the minimal CRL lifetime allowed in the version of Entrust CA we were using. Our requirement for timely revocation drove this testing, and no suitable alternative could be found. OCSP was not supported by either client, and even with OCSP, our requirement to tolerate intermittent network connectivity would have limited OCSP's utility. The most satisfactory arrangement would be if there were some way to set up a CRL publish-and-subscribe mechanism where CRLs could be pushed asynchronously to clients.

Problems were encountered when sending e-mail messages between Entrust Express software and Baltimore's MailSecure software. Entrust Express includes the entire certificate chain with each signed message. MailSecure used the chain included in the message to perform validation instead of consulting the directory. Therefore all Entrust Express-signed messages failed to validate in MailSecure because the US-signed-by-UK cross-certificate found in the directory was never seen. Since the UK's trust of the US was documented in the cross-certificate, the US root self-signed certificate was not trusted directly. Individual user certificates could be validated after opening the messages by manually resolving trust paths back to the cross-certificate. Since it would be impossible for Entrust Express to include the correct validation chain for a UK user, a straight-forward solution would be to no longer include the validation chain in messages at all. Unfortunately, Entrust Express did not provide such a facility. Inclusion of a proper validation chain would help satisfy the intermittent network connectivity requirement, but the amount of additional data sent with each message could pose a bandwidth problem under the strain of operational use.

A number of attributes needed to be added to the UK's directory entries that were mandated by Entrust: First Name, Last Name, Common Name, User ID, Password, *mailrecipient*, *nsmessaginserveruser*, *mailbox*, *Maildeliver*, *Mailhost* to correctly process them. These were not strictly needed by the UK, but were added for compatibility reasons.

Baltimore MailSecure—MailSecure did not recognize the cross-certificates we used to establish trust because it did not use the crossCertificatePair attribute of the US CA's directory entry. As a workaround, the UK obtained the US's cross-certificate signed by the UK (labeled <<US signed by UK>> in Figure 7) and copied it into the cACertificate attribute of the US CA's directory entry. They did this in a "stub" directory copied from the real directory so as not to modify the original. They then pointed MailSecure to the stub directory as the first source for certificate path validation. When a certificate's trust chain led MailSecure to the US CA's certificate in the stub directory, the *cACertificate* attribute further referenced the UK's own CA as a superior in the trust chain. We believe this work-around does not impact the trust hierarchy. However, if the same modifications were made in the master (US) directory, all PKI enabled applications under the US's CA that use the *cACertificate* attribute would work incorrectly. Therefore the stub directories are a necessary part of the approach. Fortunately, MailSecure does allow the use of multiple directories to build validation paths. Without this capability the UK users would have had to copy their entire directory into the stub directory to make the process work.

Figure 7 shows how MailSecure searches the stub directory first to find certificates. When it needs to find the US CA certificate, it finds the appropriate entry and looks at the *cACertificate* attribute. The first value in the attribute is the <<US signed by UK>> certificate

that points to the UK CA. This feature allows MailSecure to automatically trust all US-issued certificates. The self-signed certificate remains as the second value of the attribute for compatibility purposes. However, we have found, in general, that PKI path-building clients do not look beyond the first value of an attribute.

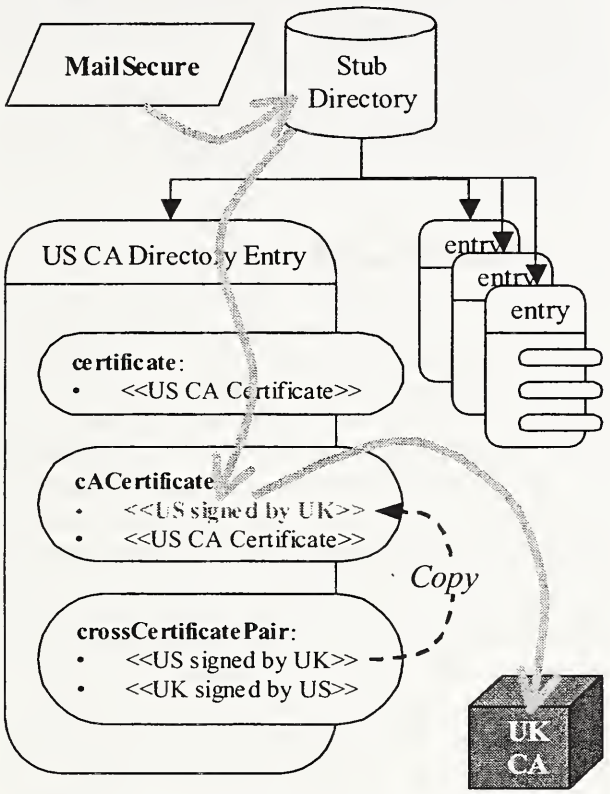


Figure 7: Using a Stub Directory with MailSecure

General PKI Problems—Some PKI products expect a country (C=) code to be the root element of all coalition DNs (after the fashion of X.500). Since VON uses the Organization (O=) code as the root, we encountered several PKI problems. For example, with no country code in the DN, the UK users were unable to generate their own keys and request certification via MailSecure. However, this was achieved at the local Registration Authority (RA) using face-to-face certification resulting in the manual transfer of user certificates to client machines. This DN restriction also means that each user may need a set of certificates for the coalition and another set for national use. The practicality of this must be considered.

RFC 2459 [3] is ambiguous in its specification of CRL Distribution Points (CDPs). Although all PKI products we used follow the standard, different legitimate interpretations resulted in incompatibilities between compliant products. We discovered that the UK's Baltimore UniCERT CDPs could be configured in a way that made them incompatible with Entrust Express, although both appeared to be following the standard. The ambiguity allowed directory locations to be resolved from UK and US certificates in incompatible ways. In order to resolve this incompatibility, we found the Issuing Distribution Point (IDP) must be set to non-critical and fully qualified CDPs must be used.

We found it necessary to use third-party utilities to confirm the correct configuration of certain pieces of software used in the trials. For example, certificate viewers and Base-64 decoding tools from the OpenSSL distribution were needed to debug problems with certificates issued by foreign CAs. We suggest that vendors include such tools in debugging suites to increase the interoperability of their software with others.

3.3.2 Accomplishments of Phase III

All participants accomplished the following during the summer 2001 test period:

1. Established network infrastructure over a private ISDN link.
   - Simulated platforms included national NOCs and several simulated ship platforms.
   - Infrastructure included both nations providing coalition e-mail and DNS servers.
2. Established nationally supplied directory services interconnected into a unified coalition directory with automatic replication between sites.
3. Set up national PKIs and cross-certified them yielding a unified coalition PKI including:
   - Directory Servers
   - Certification Authorities (CAs)
   - Registration Authorities (RAs)
   - PKI enabled e-mail clients
4. Verified the functionality of the coalition PKI via e-mail tests.
   - Conducted 48 e-mail tests (including digitally signed and/or encrypted e-mail both with and without attachments) with no unqualified failures.
   - Discovery of encryption certificates via unified coalition LDAP directory worked consistently.
5. Tested revocation of individual coalition users and cross-certificates.

3.4 Lessons Learned in Phase III

The Phase III testing identified a number of issues with the vendor products used. While all 48 email-exchange tests were successfully performed, a few of the exchanges required workarounds deemed unsuitable for

a tactical environment. These workarounds were due to PKI vendor incompatibilities. In addition a number of issues were discovered concerning the underlying network infrastructure (e.g. DNS, routing, etc), which must be resolved prior to at-sea trials. The teams will perform additional work in 2002 to get the demonstration testbed ready for at-sea trials in 2002.

Following are some logistical lessons we learned during the testing process:

- The conference telephone call was an invaluable tool that allowed problems to be solved in an efficient and timely manner. It also allowed out-of-band verification of certificate fingerprints during the cross certification process. We found using an out-of-band channel for verifying certificates and PKCS #10s to be simpler and more cost-effective than face-to-face certificate exchange.
- Detailed configuration planning in advance avoids unnecessary, lengthy reinstallations of software.
- Separating key server machines among several sites makes it more difficult to locate and rectify network configuration and other problems.
- The US found it useful to have several administrative user accounts for each nation: echo, record, and revocable. The echo user is configured so that e-mail to this user is automatically echoed to the sender. This account is useful in testing basic e-mail connectivity so that one nation can verify that another's e-mail server is responding without further coordination or specialized knowledge. The record user was used as a repository for CCs of all mail messages sent during the testing. This user's mailbox formed a complete record of all e-mail sent during the test and often served as verification that a nation actually sent a message when network congestion caused delayed delivery to the recipient. The revocable user accounts are useful for conducting revocation testing. These user's certificates are intended to be revoked for testing purposes so that other users' accounts need not be disturbed and no one's feelings get hurt!

Following are some lessons we learned about planning and managing LDAP directory servers for PKI:

- Hub and spoke replication topology worked well, allowing access to the complete coalition directory even when remote links were down. Further experiments may be needed to check that this strategy will work with high volumes of data and/or low bandwidth links.
- The e-mail address book is often separate and disconnected from the coalition directory because the directories are used for different purposes. Manually copying e-mail information into the coalition directory is a slow and error-prone method. Automatic replication between the e-mail and the coalition directories is highly desirable. . The UK successfully demonstrated Novell's dirXML product for this purpose in their testbed.
- Each nation needs to ensure that its users' entries are fully completed in the directory so that the PKI-enabled client software in use for other nations can process all users' certificates.
- The directory must be a robust product. Restarting the directory and rebooting the directory server regularly will not be satisfactory in real-time operations.

4.0 Conclusions and Future Work

The Phase IV at-sea trial will exercise the PKI configuration established and refined in earlier phases. Work is ongoing now to refine the configuration in preparation for the testing event. Several more e-mail exchange tests have been conducted, and the testing methodology has been refined to a high degree of precision. Since the test will be shipboard, a great deal of logistical matters must be considered. It normally takes over a year to determine the ships where an installation will be done, schedule a time for the ship to be in port, find a place for the installation, and verify that the installation works without negatively impacting any mission-critical systems. At this time, the logistics dominate the preparation process and the exact venue is still uncertain. This phase may involve more nations and will involve untrained users for the first time. We are prepared to collect data on both the functionality and the usability of our design from a user perspective.

In Phase V, we will seek to overcome the problems of PKI by using Extensible Markup Language (XML) and its child technologies. XML is quickly becoming the de facto standard for providing interoperability between disparate systems. XML's meteoric rise together with the momentum of Web Services may finally push PKI to deliver on its promise of universally defined trust and usability. In particular, XML standards that may be leveraged to make PKI easier to use and implement include XML Digital Signatures, XML Encryption, XML Key Management System (XKMS) and Security Assertion Markup Language (SAML). These standards may help solve problems inherent to the design of a Coalition PKI. For example, providing PKI services for nations that do not have a pre-existing PKI or the technology to establish one. With a standard web-based interface to the coalition PKI, the coalition would be able to meet nations at their level of technology and, with minimal provision, make it accessible. The coalition PKI should have a common interface that is usable in the same way by all partners regardless of the

underlying PKI provider. PKI should be a transparent part of the network infrastructure and should be usable over low-bandwidth links and on low-end workstations or mobile devices. It should allow for considerable mobility by low-end clients and be easy to set up and tear down dynamically as coalition partners come and go. Different coalition members need different access to the coalition PKI for the various roles they may play. Particularly useful is the offloading of CPU intensive PKI processes from the client to the server and making developers job of integrating PKI into applications easier. As a result, thin clients can take advantage of the strong security a full-fledged PKI provides. Using XML as a fundamental technology for PKI may allow machines to communicate in a language they already understand without a complex rollout of customized hardware and software. The issues of the online nature of these follow-on technologies will be a subject of considerable concern in this phase. We plan to contribute to the development of the standards to the benefit of all those who cannot depend on continual availability of the internet or high-bandwidth connections.

Beyond managing a single coalition, one of VON's future aims is to manage interactions among multiple, simultaneous coalitions. Each coalition must be treated as a separate "community of interest" with administrative and policy structures that are somewhat independent from those of the member nations. Additionally, there are usually multiple security levels and compartments within each community. Given n nations the potential number of communities is bounded by the expression, 2^n-1. The number of security levels and compartments is completely arbitrary and may be as complex as the coalition administration finds useful. The picture is further complicated when one considers the existence of informal ties and covert channels between nations. The rules for controlled interchange among such communities are necessarily complex and should be enabled/enforced by a coalition PKI.. This very difficult problem may not be addressable by any technological solution at all, but the goal of the VON project is to identify and implement technology that will enable at least a partial solution to problems of this sort.

In conclusion, we observe that military coalitions are often formed between partners with complex political relationships and data sharing requirements. These requirements must be underpinned by technologies that support individual identification, encryption of content for privacy purposes, data separation and access control, and non-repudiation. These will all be essential services for future network-enabled warfare operations between military allies. PKI has been shown to provide the technical underpinning for such services, and is likely to be an important part of future coalition operations. The technologies have been demonstrated practically, and are found to be reaching the state of maturity where they can be used for such purposes. Nevertheless, there are some areas where further work is required if the military is to reap maximum benefit from this young technology. In particular, policies on the use of PKI must be refined, the robustness of the technology must be determined under a variety of circumstances, and network operators must be trained in its use if it is to support coalitions of the future.

References

[1] NTA Coalition Information Technology Interoperability Final Report, 5 January 2001

[2] "Proposed AUSCANNZUKUS+ Coalition PKI Architecture for VON 2002 Pacific Demonstration," 26 June 2001, by NCAT/ Lockheed Martin M&DS and Glenn Fink, NSWC Dahlgren

[3] RFC 2587 – Internet X.509 Public Key Infrastructure LDAPv2 Schema

[4] RFC 2459 – Internet X.509 Public Key Infrastructure Certificate and CRL Profile

Biographical Data:

Glenn Fink (MS, Comp Sci) is a member of the Information Transfer Technology group at the Naval Surface Warfare Center (NSWC) in Dahlgren, Virginia. He specializes in computer and network security, especially PKI and Intrusion Detection. He has worked for the DoD for 14 years during which time he has been associated with a variety of projects mostly involving software development. He plans to leave the government in Fall 2002 and pursue a doctoral degree in Computer Science at Virginia Tech. Apart from work, his primary interest is in his family: his beautiful and intelligent wife and two sweet young children. He is the "principal" and occasional teacher for his children's home-school. His family participates in a house-based church fellowship.

Shawn Raiszadeh has worked for Lockheed Martin designing and developing cutting-edge security systems for the past three years. Shawn has worked on a number of research and development programs with the goal of creating new and innovative solutions to existing problems. Shawn has a Bachelor of Science degree in computer science from Virginia Tech.

Tim Dean, (BSc, MSc, MBCS) leads a research team and is a technical specialist in IT Security. His particular interest is in Public Key Infrastructures (PKI) and the issues associated with their practical deployment. He worked for the UK Ministry Of Defence for 14 years during which time he led teams in a variety of defence-related messaging and security projects. These included the design of new security protocols and architectures, including a key management scheme for a NATO communications network. For the last five years he has headed a research team studying network vulnerabilities and countermeasures in a military context. He now works for QinetiQ, where he is continuing his research interests. In his leisure time, he enjoys playing the piano and violin, which he uses as part of the worship group at his local Baptist Church.

---

i VON is a joint Office of Naval Research (ONR, US) and Defence Science and Technology Laboratory (DSTL, UK) In the US, ONR subcontracted its VON work to Naval Surface Warfare Center in Dahlgren, Virginia (NSWC, US); SPAWAR Systems Center in San

Diego, California (SSC-SD, US); and Lockheed Martin, Management and Data Systems, Integrated Solutions Center, eSecurity Center of Excellence in Fairfax, Virginia (LM, US). On the UK side, DSTL (formerly Defence Evaluation and Research Agency (DERA)) subcontracted work to the government-owned private company, QinetiQ Limited (QinetiQ, UK). Of these entities, our bilateral PKI research team was composed of members of QinetiQ, LM, and NSWC. To reduce confusion bred of multiple acronymns QinetiQ will be referred to as the UK team and LM and NSWC jointly will be referred to as the US team where their separate accomplishments are not significant to the context.

[ii] Entrust Express error: "(-3975) A certificate attribute for this EntrustName does not exist."

[iii] Entrust Express error:"(-4089) Signature algorithm cannot be used with given key."

Position Papers

PKI Trust Models
Position Paper for PKI Research Workshop
NIST, Gaithersburg, MD – April 24-25, 2002

Yassir Elley
Internet Security Research Group
Sun Microsystems Laboratories
yassir.elley@sun.com

One of the major obstacles to the widespread deployment of PKI is the use of poor trust models. The advantages and disadvantages of using a particular trust model need to be carefully considered before deployment. Currently, PKIs commonly use either the multi-rooted hierarchical trust model (used by web browsers) or the anarchy trust model (used by PGP). This position paper summarizes the drawbacks of these two commonly deployed trust models and advocates a third approach (known as the bottom-up trust model). For a more complete analysis, please refer to [1], authored by Radia Perlman, a member of the Internet Security Research Group at Sun Labs.

In the multi-rooted hierarchical trust model, each relying party is configured with the public keys of several well-known trusted CAs. These trusted CAs are typically associated with various security vendors and are completely trusted to vouch for anyone's public key, or to delegate authority to another CA to vouch for public keys. X.509 certificates are typically used in this model. Starting with a trusted public key, a user can attempt to build a chain (or path) of certificates to a specific target. One of the major drawbacks of this trust model is that if any of the private keys corresponding to the set of trusted public keys is compromised, the security of a vast number of entities (presumably the majority of browsers) is compromised. Even if a trusted key pair is changed for legitimate reasons (e.g. key rollover), a massive world-wide reconfiguration needs to take place. There is also the additional question of why these CAs have been granted this universal authority to vouch for anyone's public key in the first place. In the real world, trust relationships tend to be strongest at the local level and tend to dissipate as the distance between a certificate issuer and subject increases. It is unclear why a user would place greater trust in some distant CA than in a CA that is operated locally by a competent administrator. While it is possible for users to remove these universally trusted public keys, ordinary users rarely do this and are more likely to add malicious keys to this list as a result of a message box urging them to do so.

The anarchy trust model employed by PGP also uses pre-configured public keys that are completely trusted to vouch for other keys. However, these trusted public keys are typically those of close friends who the user trusts to serve as introducers to other users. This trust model addresses the reality that local trust is often stronger than distant trust. However, this model requires the user to completely trust the initial trusted public keys AND to completely trust any public keys vouched for by the initial public keys. The multi-rooted hierarchical trust model is better in this regard, because X.509 certificates allow a certificate issuer to place various constraints (e.g. name constraints, policy constraints, path length constraints) into the certificate limiting the sorts of certificates the subject is trusted to issue. Additionally, the anarchy trust model does not scale well beyond relatively small communities. Chains of certificates can be arbitrarily long and the absence of constraints on certificates can make the problem of constructing a certification path intractable.

The bottom-up trust model that we advocate incorporates the advantages of the multi-rooted hierarchical and anarchy trust models, while avoiding their disadvantages. A relying party is configured with a single trusted public key, which is usually the public key of a local organizational CA, thus providing the advantage of local trust. The bottom-up trust model assumes a hierarchical namespace and uses the properties of that namespace to efficiently construct paths from a trusted public key to a target. Chains start at the bottom (with your trusted CA), traverse up the namespace as often as necessary, cross over to a namespace ancestor of the target (if necessary) and then down to the target. Another advantage of the bottom-up trust model is that it can be deployed incrementally within a workgroup or an organization and does not require users to obtain (and pay for) certificates from an outside organization. If the CAs in your organization are managed well, keys of outside entities that are compromised will have no effect on intra-organizational security because the certificate chain between two users within the organization will never include a key of an entity outside the organization.

Finally, if a CA's public key is compromised, there is no need to re-configure all the machines in the world. Only the users who have that CA's public key configured (typically those in a particular workgroup or organization) need to be re-configured.

Rather than using specific namespace heuristics, the bottom-up trust model can be realized by using the name constraints present in X.509 certificates to constrain the search space when constructing paths. This adds flexibility to the model and allows standard X.509 certificate processing software to be employed. One result that we reported in a paper at NDSS '01 is that building paths from the trust anchor to the target is more effective for general trust models than building from the target to the trust anchor. [2] Building from the trust anchor allows the path to be validated while it is being built, enabling the quick rejection of paths that fail to validate. In addition to adopting a bottom-up trust model, software must be deployed that can build complex certification paths. To that end, our research group has contributed to the development of the Java™ Certification Path API [3] and reference implementation, which has now been released as part of JDK 1.4.

Additional Information

[1] R. Perlman, "Overview of PKI Trust Models," *IEEE Network*, Nov/Dec 1999.
[2] Y. Elley, A. Anderson, S. Hanna, S. Mullan, R. Perlman, S. Proctor, "Building Certification Paths: Forward vs. Reverse," *Network and Distributed System Security Symposium Conference Proceedings (NDSS '01),* 2001.
[3] http://java.sun.com/j2se/1.4/docs/guide/security/certpath/CertPathProgGuide.html

PKI Position Paper: How Things Look From The Trenches*

William F. Flanigan, Jr., Ph.D.
Information Assurance Chief
william.flanigan@mail.dss.mil

Deborah M. Mitchell
PKI Program Manager
deborah.mitchell@mail.dss.mil

U.S. Department of Defense
Defense Security Service
881 Elkridge Landing Road
Suite 140
Linthicum, MD 21090

As Public Key Infrastructure (PKI) technology becomes an aging teenager, it appears problematic that significant stability/maturity will be achieved as it moves into its early twenties. What follows is a brief description of some of the principal problems, pitfalls, and perils of PKI as viewed from a day-to-day operational perspective. They have not been prioritized. We present these issues with the hopeful expectation that this newly-organized Workshop will thoughtfully examine them and, where appropriate, either provide solutions directly or serve as a means to help achieve attenuating processes and mechanisms.

1. Cost. Not only are "gold-plated" PKIs to be aggressively avoided [1], but "brass-plated" and even "pot-metal" PKIs often remain prohibitively expensive to lease, buy or do-it-yourself. Large and equity-rich organizations such as governments and the institutions they support (for example, college and university systems), the financial community, well-endowed academic centers, and, in general, the Fortune 100 seem to be unique in their ability to mount post-pilot PKIs. It appears that this is because these entities can not only afford PKI, but are not necessarily subject to the same level of return-on-investment (ROI) scrutiny (or any ROI scrutiny) faced by the rest of the .com's, .gov's, .edu's, and .org's on the planet [2]. There are also PKI life-cycle maintenance/technology-migration costs (often "hidden" or not fully comprehended initially) that will be incurred at the organizational-PKI level as well as for subcomponents (such as corporate departments and governmental offices/agencies) utilizing the "free" organizational PKI utility. These hidden costs can prove to be formidable, and may preclude the continued use of the PKI. What can be done to mitigate this cost problem? Are most/all PKI costs directly related to the needless complexity of PKI technology?

2. Needless Complexity. PKI is anything but a simple, user-friendly technology. Part of the complication seems to stem from digital ancestor worship of closed-door-generated standards going back to at least the mid 80's. However (and ironically), the complexity also would appear to be due to the current open-standards process.

* This is an unofficial position paper submitted to the First Annual PKI Research Workshop. It does not represent the views, policies or positions of the U.S. Department of Defense or the Defense Security Service. The authors have a combined 6-7 years of experience in participating in Internet standards bodies as well as in designing, building, implementing, and employing large scale COTS-based PKIs.

The outputs of Internet standards bodies are primarily generated by de facto "professional" protocol writers. As is the case with all professions, these individuals speak a unique, esoteric language. Further, protocols are usually crafted by committees whose members have a multiplicity of conflicting agendas. The result seems to be protocols so convoluted and obtuse that vendor implementation is difficult/impossible and costly (the latter may discriminate against new and emerging innovative enterprises). Further, product protocol compliance is no guarantee of product interoperability (or even out-of-the-box full functionality). What needs to be addressed to reduce/remove this peril? Do PKI protocols call out for a hefty application of the KISS principal?

3. End-User Needless Confrontation. End-user awareness, training and education is unrelenting, and often self defeating. Extensive hand holding is required with end users to get them set up and started using this security mechanism. Once implemented, users continue to find it easier to just turn PKI off rather than to try to figure out (or remember) what actions they need to take to use it. This is compounded with the roll out of new PKI releases and versions. Virtually nothing about PKI is totally transparent (or even opaque) to the end user. What corrective actions could/should the PKI community pursue to eliminate this pitfall? Must this situation continue unabated? Is the lack of PK-enabled, commercial-off-the-shelf (COTS) applications that painlessly foster (or "force") the utilization of PKI part of the problem?

4. Certificate Revocation Lists (CRLs). PKI provides level-of-assurance trust interoperability between end users and machines with certificate-revocation status comprising a large part of that trust. The CRL concept requires the replying party to not only validate a subscriber's signature certificate, but also to validate the certificates of the subscriber's signing CA plus the signature certificates of all the intermediaries. As to the latter (and ignoring local caching), it's not inconceivable that a relying party may need to check the revocation status of the certificate used by a validation authority to sign a path validation guarantee who, in turn, may need to check the revocation status of the certificate used by a responder to sign a delta CRL who, in turn, may need to check the revocation status of the certificate used by a CRL distribution agent to sign the latest, distributed CRL who, in turn, may need to check the revocation status of the certificate used by the CA to sign the CRL. With multiple PKIs, the process grows in complexity. What measures and mechanisms could/should the PKI community be designing/testing to mitigate/eliminate this problem? Is certificate-revocation status based on CRLs a needless (and expensive) digital goat rope?

5. Configuration Management of Self-Signed Certificates. Browsers now come preloaded with up to 100 self-signing certification authorities (CAs) and roots which are automatically trusted by the client. Who are these entities? Some could be your competitors or adversaries. Intruder addresses and certificates inserted into a secure message thread could be

automatically trusted. The same may hold for person-in-the-middle Web sites exhibiting "authenticated" certificates and running SSL/TLS. How can this peril be countered or minimized, at what level(s), and will it be scalable? Should this issue be elevated to the level of a cyber terrorist threat?

6. Key Escrow and Recovery. It is essential for businesses and organizations to be able to access encrypted data in the event that something happens to an employee or the employee's cryptographic module. Some PKI products provide limited support for key escrow and recovery, but not (yet) for third-party certificates. It would seem prudent that escrow and recovery policies and practices evolve and migrate in full and complete synchronization with available technologies (building a policy field is no guarantee that the open-standards, COTS vendors will come!). The legal ramifications, complexities, and costs are also directly related to what technical procedures, processes, and devices are available (and utilized). Further, there are not insignificant life-cycle issues to contend with. How will data, audit trails, and electronic records in general required to be retained in an encrypted state for extended periods of time (due to legal requirements) continue to be protected as well as made accessible to ever-changing parties with access rights and privileges? Can this problem be adequately addressed using open-standards, COTS-based products? Like the profiles and checklists that have been formulated and adopted for certificates, CRLs, policies, and practices by the PKI community, is a key escrow and recovery profile and checklist also called for?

7. Registration Agent (RA) as Super-user Inside Intruder. In most/all PKIs, the RA would seem to have unlimited power with unquestionable authority. By virtue of their role, RAs must be trusted, but what about the insider threat? The RA is not just limited to making changes in the organization or business they support; they can, in fact, effect changes for any objects under the root and signing CAs of the PKI. How can this peril be prevented or at least managed effectively? Can corrective action be expected from open-standards, COTS-based PKIs?

8. Cross Certification (Trust Inter-operability) In The Client. Passing the key-management buck to the client is probably tantamount to zero key management. End users are likely to just click through the manual trust sequence when each new PKI is presented. Checking multiple (or any) CRLs is also pretty unlikely if not impossible both technically (at least for now) and psychologically (forever?). What processes or techniques might eliminate this pitfall for, say, S/MIME? Could a pragmatic solution (in terms of cost and scalability) consist of reducing the subscriber's key management to just two key sets, the mail-server's certificates and the subscriber's certificates?

References

[1] T. Polk. Panel comments. *COTS-Based PKI: The Hard Questions Persist.* RSA Conference 2002, February 2002.

[2] W. Price. Panel comments. *COTS-Based PKI: The Hard Questions Persist.* RSA Conference 2002, February 2002.

Novel Schemes for Certificate Management in Public-Key Infrastructure[1]

Ravi Mukkamala
Department of Computer Science
Old Dominion University
Norfolk, Virginia 23529-0162
mukka@cs.odu.edu

Abstract

Efficient and timely distribution of certificate revocation information is the biggest challenge currently facing the providers of Public-key Infrastructure (PKI). All of the current schemes, including the Certificate Revocation List (CRL) and its variants, place a considerable processing, communication, and storage overhead on the infrastructure elements (e.g., Certification Authorities (CAs) and its repositories) as well as the relying parties. We think that the concepts of *active certificates* and *recertification* would greatly improve the current situation. An active certificate is one that not only contains static data but also executable code. This concept also gives rise to several possibilities in using digital certificates for authentication, authorization, access control, and privilege management. With recertification, a certificate needs to be recertified periodically during its lifetime. This additional step is expected to reduce the size of the revocation lists drastically and thereby make the process of validation more efficient. In addition, it may make it possible to offer several qualities of service to a relying party that are not possible in the current system. The PKI research group at the Old Dominion University is currently investigating these concepts in much more depth to investigate their feasibility and utility in real-world applications.

Detailed Position Statement

In PKI, a certification authority (CA) accepts requests for certificates and issues the same after verifying the authenticity of the user provided information. Basically, it plays the role of a *trusted third party* (TTP), certifying the identity of one party to another. While the primary intent of a digital certificate is to assure a relying party that a public-key indeed belongs to the purported owner, it is now being used for other purposes such as authenticating other attributes of a certificate holder and even for access control.

When a PKI certificate is issued, it is expected to be in use for its entire validity period. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Traditional method of managing certificate revocation is through CRLs as specified in X.509. Here, a CA issues a CRL periodically and posts it to a repository (or a directory service). The CRL includes all unexpired certificates issued by the CA that have been revoked. Each CRL includes a *nextUpdate* field that specifies the time of the next CRL issuance. A relying party requiring certificate status information, that does not already have an unexpired

[1] The work is supported in part by a grant from Commonwealth Information Security Center (CISC) at the James Madison University, Virginia, USA.

CRL, retrieves current CRL from the repository. Several variants of CRL schemes have also been suggested.

However, both PKI researchers and practitioners have identified several shortcomings of the CRL and its variants. First, they are expensive to distribute. Second, they involve expensive storage and validation costs at the relying parties (e.g., service providers). Third, they provide only negative information (i.e., a certificate is not revoked) instead of positive confirmation. Fourth, they place a considerable burden on a relying party to verify a user's certificate. Fifth, they contain substantial redundant information (e.g., consecutively published CRLs would have more than 99% of redundancy).

We (myself and a group of graduate students) are currently investigating two mechanisms to solve the current problems in certificate management: active certificates and recertification.

Active Certificates

Current digital certificates are passive---they are simply a stream of bits (or bytes) of data. They are not executables. Whenever a certificate-holder needs a service, he/she submits the certificate to a relying party (service provider). The relying party is now responsible for validating the certificate. This often involves contacting a chain of certificate authorities and processing several CRLs. This process is both resource and time consuming. Often the relying party spends time in locating the CA or other repositories. Some time, a relying party may not have the required bandwidth, the desired storage, or the processing power to do such validations. Since a relying party is more interested in expending resources for its own service rather than validation of certificates, we need to find an alternate way.

Our solution to the problem is active certificates. This term is coined by us and is new to the PKI world. According to our definition, *an active certificate is one that contains not only the necessary data but also an executable code*. In other words, it is similar to an applet or servlet in Java terminology. Now that a certificate is an executable, several opportunities exist for its use. For example, we can now shift the burden of verification and proof of validation on the certificate-holder instead of a relying party. In addition, instead of simply using a certificate for authentication, we can extend its usage for authorization (e.g., a line-of-credit of $4,000 granted by a certificate), access control, and privilege management. It also may be helpful in more efficient management of certificate revocation. In this context, we are currently investigating the following issues.

1. How should the active certificates be implemented in the current technology?
2. How can the certificates be used for authorization? How should the authorizations change as a certificate is being used? How to prevent duplicate authorizations being created?
3. How to revoke privileges or authorizations?
4. What security and trust concerns are introduced due to active certificates that are capable of modifying themselves?
5. What types of newer domains of applications can the active certificates be used that were not even considered with the current static certificates?

Recertification

The primary impetus for introducing the recertification concept comes from the following observations:

- *It is more efficient for a CA to issue certificates with long validity periods.* Since there is a considerable overhead involved in issuing a certificate, it is more economical to issue long-life certificates.

- *The information about a revoked certificate needs to be maintained and distributed until its expiration time.* In other words, longer the lifetime of a revoked certificate, longer is the period of maintaining its status by a CA or a repository. So a CRL, for example, keeps maintaining a revoked certificate on its list until it expires. A longer CRL is expensive (processing cost) to prepare (at CA), expensive (communication cost) to distribute to repositories, expensive (communication cost) for relying parties to copy from the repositories, and expensive (processing cost) for the relying parties to search when users submit requests.

The concept of recertification aims to combine the benefits of long-life certificates for an issuer with the benefits of short-lived certificates for revocation. The main idea is to initially issue a certificate for the normal period of duration (e.g., 1 or 2 years) and then require the certificate-holder (or user) to get the certificate recertified at certain intervals during its lifetime. A relying party not only looks for the lifetime of a certificate but also for its recertification at the time of verification. To reduce the load on the certificate issuer (e.g., CA), the recertification task is assigned to a different entity called the recertification authority (RCA). Certainly, RCA should have been delegated this authority by a CA, say by issuing an attribute certificate to this effect. Typically, RCA does not have to be as trusted and secure as a CA, since it does not originate the certificate but only recertifies it. However, the CA should certify it so the relying parties can trust its actions.

In this context, we are currently investigating the following aspects of recertification.

1. Suggest schemes for efficient implementation of recertification concept.
2. Identify the trust and security concerns of introducing recertification.
3. Evaluate the performance improvements due to the introduction of recertification and thereby reducing the length of Cross. Similarly, evaluate the increase in load due to frequent renewals of certificates. A detailed cost-benefit analysis is necessary if the idea is to be accepted by the PKI community.
4. Identify application domains where recertification would be most suitable. Identify those where recertification would be too expensive or not appropriate.

In summary, we feel that introducing the active certificates and recertification will greatly extend the range of applications that the PKI can be used. It will also offer several qualities of services for both relying parties and users that are currently not possible.