NISTIR 7122

3rd Annual PKI R & D Workshop Proceedings

Krishna I. Sankar[a]
William T. Polk[b]
Nelson E. Hastings[b]

[a]Cisco Systems

[b]U. S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Information Technology Laboratory
Gaithersburg, MD 20899-8230

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

NISTIR 7122

3rd Annual PKI R & D Workshop Proceedings

Krishna I. Sankar[a]
William T. Polk[b]
Nelson E. Hastings[b]

[a]Cisco Systems

[b]U. S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Information Technology Laboratory
Gaithersburg, MD 20899-8230

September 2004

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary

TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary 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.

Foreward

NIST hosted the third annual Public Key Infrastructure (PKI) Research and Development Workshop on April 12-14, 2004. The two and a half day event brought together PKI experts from academia, industry, and government to explore the remaining challenges in deploying public key authentication and authorization technologies. These proceedings includes the 10 refereed papers, and captures the essence of the six panels and interaction at the workshop.

The first two workshops concentrated on PKI theory, architectures, human factors, and deployment. Building on this prelude, the third workshop focused on applying PKI technology to the problem of authorization, current real world interoperability issues and solutions, and retrospective views of PKI technology - taking stock of the progress and failures in PKI's evolution.

Stefan Brands' keynote presentation, which focused on the limitations of traditional authentication primitives in cross-domain environments, opened the workshop. The rest of day one explored PKI and authorization, including several techniques for delegating and sharing authorizations, a decentralized technique for X.509 based authorization for wireless networks, and a role sharing technique leveraging passwords and PKI.

Day two of the workshop featured Peter Gutmann's keynote presentation describing six "Grand Challenges" facing PKI technology, and proposing various solutions. The papers and panels which followed spanned a wide range of topics including PKI interoperability problems and solutions faced by real world PKI projects, trusted archiving, document signatures, and a retrospective look at the history of PKI.

Ken Klingenstein's keynote presentation on the necessity to interconnect a variety of security technologies beyond PKI to achieve our security goals kicked off the final day of the workshop. Day three topics included approaches and workarounds to path discovery, PKI deployment issues encountered by Johnson & Johnson, and results of the OASIS PKI survey. The workshop concluded with a panel discussion on the potential impact of the OASIS report.

The workshop also included a work-in-progress session and a birds-of-a-feather session during the evenings at the workshop hotel. Attendees included presenters from the United Kingdom, Canada, New Zealand, and Japan. Due to the success of this event, a fourth workshop is planned for 2005.

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

This page has been left intentionally blank.

2004 PKI R&D Workshop

Gaithersburg, Maryland USA

April 12-14, 2004

http://middleware.internet2.edu/pki04/

(Pre-proceedings were distributed at the workshop)

SUMMARIES

Provided by Ben Chinowsky, Internet2

Workshop Summary 1

NIH-EDUCAUSE PKI Interoperability Project: Phase Three
Document Signatures
Johnson & Johnson's Use of Public Key Technology
How to Build a PKI that Works
Controlled and Dynamic Delegation of Rights
Approaches to Certificate Path Discovery
Non-Intrusive Cross-Domain Identity Management
Which PKI Approach for Which Application Domain
A New and Improved Unified Field Theory of Trust
The PKI Action Plan: Will it make a Difference?

Works in Progress Session 11

REFERRED PAPERS

Role Sharing in Password-Enabled PKI 13

Xunhua Wang *James Madison University*
Samuel Redwine *James Madison University*

Greenpass: Decentralized, PKI-based Authorization for Wireless LANs 26

Nicholas C. Goffee *Dartmouth College*
Sung Hoon Kim *Dartmouth College*
Sean Smith *Dartmouth College*
Punch Taylor *Dartmouth College*
Meiyuan Zhao *Dartmouth College*
John Marchesini *Dartmouth College*

X.509 Proxy Certificates for Dynamic Delegation **42**

Von Welch — *NCSA, University of Illinois*
Ian Foster — *Argonne National Lab/University of Chicago*
Carl Kesselman — *University of Southern California*
Olle Mulmo — *Royal Institute of Technology, Sweden*
Laura Pearlman — *University of Southern California*
Steve Tuecke — *Argonne National Laboratory*
Jarek Gawor — *Argonne National Laboratory*
Sam Meder — *University of Chicago*
Frank Siebenlist — *Argonne National Laboratory*

Experiences of establishing trust in a distributed system operated by mutually distrusting parties **59**

Scott Crawford — *Enterprise Management Associates*
David Chadwick — *University of Salford*

Trusted Archiving **71**

Santosh Chokhani — *Orion Security Solutions*
Carl Wallace — *Orion Security Solutions*

PKI: Ten Years Later **80**

Carlisle Adams — *University of Ottawa*
Mike Just — *Treasury Board of Canada*

An Examination of Asserted PKI Issues and Proposed Alternatives **96**

John Linn — *RSA Laboratories*
Marc Branchaud — *RSA Security Inc.*

Private Revocation Test using Oblivious Membership Evaluation Protocol **110**

Hiroaki Kikuchi — *Tokai University*

"Dynamic Bridge" Concept Paper **119**

Ken Stillson — *Mitretek Systems*

Identifying and Overcoming Obstacles to PKI Deployment and Usage **131**

Stephen R. Hanna — *Sun Microsystems, Inc.*
Jean Pawluk — *Inovant*

Organizers

General Chair: Ken Klingenstein, University of Colorado
Program Chair: Krishna Sankar, Cisco Systems
Steering Committee Chair: Neal McBurnett, Internet2
Local Arrangements Chair: Nelson Hastings, NIST
Scribe: Ben Chinowsky, Internet2

Program Committee

Peter Alterman, *NIH*
Bill Burr, *NIST*
Yassir Elley, *Sun Microsystems*
Carl Ellison, *Microsoft*
Stephen Farrell, *Trinity College Dublin*
Richard Guida, *Johnson & Johnson*
Peter Honeyman, *University of Michigan*
Russ Housley, *Vigil Security LLC*
Ken Klingenstein, *University of Colorado*
Olga Kornievskaia, *University of Michigan*
Neal McBurnett, *Internet2*
Clifford Neuman, *USC*
Eric Norman, *University of Wisconsin*
Tim Polk, *NIST*
Ravi Sandhu, *George Mason University; NSD Security*
Krishna Sankar (Chair), *Cisco Systems*
Jeff Schiller, *MIT*
Frank Siebenlist, *Argonne National Laboratory*
Sean Smith, *Dartmouth College*
Michael Wiener, *Cryptographic Clarity*

Archival Sites

PKI 2003:	http://middleware.internet2.edu/pki03
PKI 2002:	http://www.cs.dartmouth.edu/~pki02

This page has been left intentionally blank.

3rd Annual PKI R&D Workshop Summary

Ben Chinowsky, *Internet2*

The workshop announcement listed the goals of this gathering as:

1. **Explore the current state of public key technology** in different domains including web services, grid technologies, authentication systems et. al. in academia & research, government and industry.
2. **Share & discuss lessons learned and scenarios** from vendors and practitioners on current deployments.
3. Provide a forum for leading security researchers to **explore the issues relevant to the PKI space** in areas of security management, identity, trust, policy, authentication and authorization.

This summary groups workshop sessions according to which of these goals was their primary concern, although many sessions addressed more than one.

Surveying Deployments

Dr. Susan Zevin, Acting Director of the Information Technology Laboratory at NIST, opened the meeting by noting some Federal PKI highlights. The Federal Bridge Certification Authority now connects six Federal agencies. The Department of Defense now requires contractors to obtain PKI credentials for email and authentication to DoD web sites. Several countries and international associations, including Canada, Australia, and NATO, are negotiating to connect to the Federal PKI. NIST is a global leader in smartcards and biometrics and their integration with PKI.

A panel discussion with Peter Alterman, Deb Blanchard, Russ Weiser, and Scott Rea discussed the **NIH-EDUCAUSE PKI Interoperability Project: Phase Three.** This project has been largely driven by the Government Paperwork Elimination Act; in order for virtual paperwork not to become just as much of a hassle as the physical paperwork it replaces, reducing the number of certificates each person needs ("reusability") is essential. While this is still at the technology-demonstration stage — a production implementation has additional, expensive, datacenter requirements — various agencies including GSA and HHS are adopting elements for production use. This uptake in the federal environment is what this seed project is all about. The panelists' project report describes progress to date in great detail.

The use of **Document Signatures** in land-ownership transactions in Canada was also the subject of a panel discussion. Attorney and former crypto engineer Randy Sabett compared physical and digital signatures, and in particular explored the issue of digital signatures being held to higher standards than physical ones. John Landwehr, from Adobe, described how Acrobat supports signatures from both the author and the user of a form, in order to guard against spoofing and data modification respectively; there has been strong customer demand for this. The centerpiece of this panel was Ron Usher's description of the application of the tools described by Landwehr to a real-world situation that raised many of the legal issues described by Sabett: moving the documentation of Canadian land-ownership transactions to an electronic format. Forgery of paper documents has been a big problem in the Canadian land-tenure system; this and the need for greater efficiency were the principal drivers of the move to secure electronic documentation. Usher described his philosophy as PKE, with the E standing for Enough: "usually what we really need is public-key cryptography," with infrastructure to be added only as needed. Usher's company, Juricert, was launched by the Law Society of Canada to implement this approach. Lawyers, not

landowners, are the ones who sign the documents in the Canadian system, so it's only they who need certificates. On the other hand, Usher observed that lawyers tend to be very conservative about process. One key to user acceptance of the switch to electronic transactions is to make the form look as much like the paper version as possible. This is a main reason for choosing Acrobat (though a TIFF image is the permanent legal record). The new system provides an "astounding improvement" in transaction time. The government had been re-keying information keyed and printed by lawyers; this system eliminates the keystroking — a big win for the cash-strapped government. The benefits have prevailed over the lawyers' conservatism: the new system has handled $400 million (Canadian) in offers and ownership transfers in the few weeks it has been in operation.

Rich Guida offered an overview of **Johnson & Johnson's Use of Public Key Technology.** The J&J PKI is enterprise-directory-centric — a certificate subscriber *must* be in the enterprise directory (which is an internally LDAP-accessible AD forest). Guida stressed the importance of providing proper training for helpdesk personnel and providing enough helpdesk resources. J&J produced a one-page document distilling what users need to know to use the PKI — what tokens are for, where you use them, what to do when asked for a passphrase, etc. — and found that users often wouldn't read even this, but would instead call the helpdesk for even the most basic questions. On the other hand, J&J was able to do most configuration and credential preparation independently of the users. Guida also noted that while it has taken significant effort to get users to treat their USB tokens as a must-have item like their car keys or J&J badge, "the beauty of using the token is starting to catch on." Users particularly appreciate having a single passphrase that doesn't have to be complex or be changed every 90 days. USB tokens were chosen over smartcards only because of the ubiquity of USB ports; Guida expects a move to multifunction smartcards (e.g., used for building access also) over time. Standardization on 2048-bit keys will help drive the transition.

David Chadwick related **Experiences of establishing trust in a distributed system operated by mutually distrusting parties.** The mutually distrusting parties in question are national governments involved in a worldwide effort to monitor production of environmental contaminants capable of doing harm across international borders. Currently about 100 of 300 monitoring sites are sending signed messages to a data collection center. Every message must be signed by a threshold number of the mutually distrusting parties; this m-out-of-n principle is used wherever possible. Chadwick noted that human factors have been a major focus in both deployment and operation.

There were also two presentations relating experiences using PKI for the specific tasks of delegation and archiving.

Von Welch reviewed the use of **X.509 Proxy Certificates for Dynamic Delegation.** Proxy certificates were first prototyped in 1998 and were standardized in PKIX earlier this year; an RFC is imminent. Proxy certificates are part of the Globus toolkit and are now widely used in scientific testbeds in many countries. There are three authorization models: identity-based authorization (i.e., impersonation), restricted delegation of rights, and attribute assertions without delegation; most implementation experience has been with the first of these. The users seem pleased; their main complaint is that certificates exist as files on the local machine.

In the **Trusted Archiving** session, Santosh Chokhani and Carl Wallace described a proof-of-concept trusted archive that they built for the US Marine Corps. The approach taken was refreshed timestamps, with RFC 3161 rules used to verify that the correct data was stored. Chokhani called the group's attention to LTANS, an IETF working group formed for trusted archive standards.

Drawing Lessons

Two sessions were devoted primarily to this goal.

Peter Gutmann keynoted on **How to build a PKI that works.** After presenting an entertaining catalogue of PKI disasters, Gutmann offered a list of six "Grand Challenges" for PKI, along with proposed approaches to meeting those challenges.

1. Challenge: key lookup. Response: "the Web is the Public File." In its simplest form, this would mean putting a certificate on your home page and letting people find it with Google; while he's not promoting this, Gutmann noted it would still be better than anything currently available. His more serious proposal is "http glue + anything you want"; pretty much any database now supports the Web, many with surprisingly little effort. See http://www.ietf.cnri.reston.va.us/internet-drafts/draft-ietf-pkix-certstore-http-07.txt.
2. Challenge: enrollment. Response: make it transparent. Gutmann quoted Bo Leuf: "the vast majority of users detest anything they must configure and tweak." The norm when trying to get a certificate issued is to be subjected to pages and pages of hassle; there is a persistent myth that this is inherent in the process of certificate issuance. By contrast Gutmann cited the ISP model: you call the ISP with a credit card, they give you a username and password, you use them, DHCP does the rest. We need to remember that our PKI-enabled applications only have to be as secure as the best non-PKI alternative. More on this "plug-and-play" approach to PKI is in http://www.cs.auckland.ac.nz/~pgut001/pubs/usenix03.pdf.
3. Challenge: validity checking. Response: Gutmann outlined an approach based on querying hashes submitted by users; this puts the work on the client.
4. Challenge: user identification. Response: Distinguished Names "provide the illusion of order" but create chaos. Gutmann used a variety of examples of this to argue for treating Distinguished Names as meaningless bit strings, and using binary compare for name comparisons.
5. Challenge: no quality control. "Some of the stuff out there is truly shocking." Again Gutmann provided a rich variety of examples. Response: Create "brands" and test procedures to become brand-certified (e.g., S/MIME testing under RSADSI); against these brands, test the basics only (lookup, verification and basicConstraints/keyUsage enforcement); make sure that users know that while software certified to the brand will work, software not so certified could do anything.
6. Challenge: implementer/user apathy. E.g., never updating CRLs, but checking against them anyway in order to formally meet requirements. Response: "Make the right way the only way to do it."

Gutmann's slides for the workshop (124 of them) develop his proposed approach in detail; he also provides crypto libraries to support it (see http://www.cs.auckland.ac.nz/~pgut001/cryptlib/).

The other big "lessons learned" compilation was Carlisle Adams' presentation on **PKI: Ten Years Later.** Adams dates PKI from 1993 and the beginnings of X.509 dissemination. Three big lessons from those ten years are:

- As Gutmann detailed, PKI is hard to implement.
- User issues are key.

- The principal lesson of the many deployment issues is the need for many choices in the various PKI technology areas. In each of the six principal areas of PKI technology — authority, issuance, termination, anchor, private key, validation — the last ten years have increased the number of choices from one to several. The key now is to use this large toolkit for real-world deployments.

There were three particularly interesting exchanges in the Q&A portion of Adams' session:

- Adams' reference to Ellison and Schneier's "10 Risks of PKI" as the best-known compilation of criticisms of PKI (along with Gutmann's, which is more deployment-oriented) prompted Ellison to point out that he himself is now a critic of that paper. (See http://world.std.com/~cme/html/spki.html for links to this paper, and to the CACM Inside Risks columns derived from it, which Ellison considers to be better written.) Ellison noted that he and Schneier were directing their fire primarily at the marketing literature around PKI, not at PKI technology itself; he recommended his paper from PKI02 (http://www.cs.dartmouth.edu/~pki02/Ellison/paper.pdf) as a substitute. Schneier, however, still pushes the "10 Risks" paper.
- In response to David Chadwick's observation that DNS demonstrates the viability of a global namespace, Ellison predicted that political forces will never allow a global namespace to happen again. Owning the world's namespace gives you tremendous power; it happened the first time because nobody noticed until it was already established, and because it was created by technical people who weren't out for political power.
- Eric Norman suggested that the one thing that's remained constant over the last ten years is the keypair. Adams replied that not even that has remained constant — once people thought everyone just needed one keypair or maybe two (for signing and encryption); now there's a general acknowledgment that everyone will need multiple keypairs.

Identifying Tasks

The bulk of the sessions at PKI04 were devoted to identifying and prioritizing tasks needed to move PKI forward. The two main themes that emerged from the previously described sessions — 1) human factors and 2) letting practical needs drive technology choices rather than vice versa — were dominant here as well.

Six sessions addressed directions for specific technical areas.

In the **Controlled and Dynamic Delegation of Rights** panel, participants put forward various tools for addressing this problem. Moderator Frank Siebenlist presented on Grid needs for delegation of rights; he believes that industry will face similar requirements in two or three years. Carl Ellison argued that when rights are delegated it is vital that the act of delegation be performed by the authority on the rights being delegated, rather than by the party that happens to control some authorization database. More generally, Ellison stressed that the user is part of the security protocol; Ellison's work on procedures designed to take proper account of this fact ("ceremonies") is documented in http://www.upnp.org/download/standardizeddcps/UPnPSecurityCeremonies_1_0secure.pdf. Ravi Pandya presented XrML 2.x as a general-purpose policy language, not the narrow DRM language it's often seen as (XrML 1.2 was much more limited). Kent Seamons presented TrustBuilder (http://isrl.cs.byu.edu/projects.html), an architecture for automated trust negotiation based on gradual disclosure of credentials. Seamons noted that this is a growing field; William Winsborough is another key person working in this area.

In the discussion, Steve Hanna asked why there had been no presentation on XACML; Frank Siebenlist, who's on the XACML TC, noted that XACML has no delegation capability, though there are plans to add this. Carl Ellison related his experiences with SPKI/SDSI to his current involvement with XrML: lack of industrial-strength toolkits and marketing are the main reasons SPKI hasn't deployed; this in turn is due to SPKI's lack of CAs precluding anyone from making money from it. But, XrML has all the power of SPKI/SDSI and more, and it's backed by Microsoft. Pandya added that the core of XrML is pretty much final, and that toolkits are in the works. Microsoft is committed to getting organizations like Globus to take it up and work it to its full broad potential. Information on the IPR status of XrML is at http://www.xrml.org/faq.asp.

Ken Stillson of Mitretek presented a **"Dynamic Bridge" Concept Paper.** Stillson observed that the path-discovery process scales very poorly and is brittle: path discovery has no sense of direction, and taking a wrong turn can lead to a wild goose chase. "Namespaces aren't organized in a way that facilitates a routing protocol." The Dynamic Bridge provides a means of consolidating paths so that intermediate nodes no longer make you have to guess. There is substantial overlap between these ideas and work on shortening certificate chains done by Radia Perlman at Sun. Mahantesh Halappanavar noted that he and his co-authors have also published work along similar lines. Mitretek owns the patents on the Dynamic Bridge concept, but has no intent to assert patent protection. They are looking to start a discussion on possibilities for implementation; contact stillson@mitretek.org if you are interested.

Stillson's talk was followed by a panel discussion on **Approaches to Certificate Path Discovery.** Peter Hesse reviewed the basic PKI structures that path discovery must deal with, describing them as all meshes, just of different shapes. Path building has not yet been addressed by IETF standards, but an informational Internet-Draft (I-D) is in the works. Steve Hanna explored analogies for path building. Is it graph theory? Only if you download all the certificates in the world. Is it navigation? Sort of. Really it's like building a deck — going out and getting things, then repeatedly running back for things you forgot, is most of the work. So, work with what you've got, keep scraps, collect tools ahead of time, and work carefully. The common theological issue of the right direction in which to build paths needs to be answered accordingly: "it depends." Meeting in the middle is also an option, particularly appropriate for bridged topologies. Hanna suggested that more research is needed: test different path-discovery modules with different topologies, and try to find the best algorithms for particular sets of circumstances. This would make a great master's thesis and could generate dozens of papers. Matt Cooper summarized the approaches he's taken in writing three pathbuilding modules, and shared test results quantifying the usefulness of various simplifications such as pruning and disallowing loops through CAs. He also stressed the importance of doing as much validation as you can in the process of doing discovery. Ken Stillson stressed that in addition to the tasks of path discovery and path validation there is also the task of object location — as there is no global directory, even if you know the Distinguished Name (DN), you don't necessarily know how to get the certificate, so you end up having to implement a bunch of different access methods.

Hesse then moderated a discussion:
What is the goal when discovering paths? The consensus here was that (as Hanna put it) "any path is a good path." Cooper observed that it's likely that the first path you find is the intended path even if it's not valid, so that path should be reported to the user. It's also important to be able to specify a timeout: e.g. users only want it to take a few seconds for email, and a search that takes more than five minutes is very unlikely to succeed.
Is path discovery best done on the client or on the server? There appears to be a consensus that the answer here is the same as the answer to the forward vs. backward issue — "it depends" — though Stillson pointed out that audit requirements may dictate doing path discovery on the server.

What are your recommendations for PKI architects?

- Hanna: Send "a bag of certs" to the end entity via S/MIME or SSL; use name constraints in cross certificates; avoid high fan-out/fan-in.
- Stillson: Take advantage of the documents coming out of NIST. These include recommendations drawn from trying to get the bridge to work, in particular on certificate profiles, directory structure, and path discovery requirements.
- Cooper: Use name constraints; put data in the directory where it belongs.
- Hesse: Make sure your keyIDs match; use the authorityInformationAccess field.

Who has the obligation *to do path discovery?* The only consensus on this appears to be that it is an important unresolved question. Stillson noted a related question: Who's liable if a valid path tells me to do something I shouldn't?

What can be learned from PGP? Hesse observed that PGP doesn't really have a discovery mechanism; the user needs to know somebody it trusts, then build a local copy of the PKI that it cares about. On the other hand, Stillson cited the trust scoring system in PGP as having relevance. Neal McBurnett pointed the group to statistics on the PGP web of trust and links to path-building services at http://bcn.boulder.co.us/~neal/pgpstat/.

Steve Hanna wrapped up the path-discovery session by asking all with sample PKI topologies to send them to him (shanna@funk.com) for testing. Anyone interested in further research on path discovery and validation should also contact him.

Nicholas Goffee presented **Greenpass: Decentralized, PKI-based Authorization for Wireless LANs.** This project is driven by guests wanting access to Dartmouth's wireless network. Greenpass uses a SPKI/SDSI authorization certificate to bind authorizations to a public key; the delegation process makes use of a "visual fingerprint" assigned to a guest and verified by the delegator before signing the certificate. The certificate chain gets stored as a cookie on the guest's machine so the guest can reauthorize without repeating the introduction process. A pilot deployment is in the works.

Xunhua Wang presented a method for **Role Sharing in Password-Enabled PKI.** Roles are preferred to individuals as the subject of security because they are more permanent and because security policies are concerned with roles, not individuals. The principal advantage of the proposed approach is its lightweightness: users need passwords only, not smartcards or segments of the private key.

Hiroaki Kikuchi outlined a **Private Revocation Test using Oblivious Membership Evaluation Protocol.** In the course of certificate status checking, OCSP servers learn the relationship between the certificate holder and certificate checker. There is a privacy issue here; the proposal outlines an "oblivious membership test" to address this.

Another six sessions were specifically devoted to identifying key issues and next steps for PKI as a whole.

Stefan Brands outlined a comprehensive heterodox approach to making use of public-key cryptography: **Non-Intrusive Cross-Domain Identity Management.** In Brands' view, the Achilles heel of X.509 is its fundamental incompatibility with privacy: public keys are "strongly authenticated 'super-SSNs'". Brands pointed out the shortcomings of various proposed solutions to the privacy problem within the X.509 framework: pseudonyms and roles, attribute certificates, per-domain CAs and certificates, and federated identity management. Instead, "new authN primitives" are required. Brands' alternative, called Digital Credentials, is based on twenty years of research by dozens of

academics, starting with David Chaum's work in the 1980s. The features of Digital Credentials include "sliders" for privacy and security, selective disclosure/hiding of attributes, unlinkability, and a variety of choices along the identity-pseudonymity-anonymity spectrum. Digital Credentials are patent-protected, but Brands stressed that this is only so that he can secure the investments needed to drive real-world deployments. Brands is willing to make the technology available where doing so does not conflict with this goal; contact him if you have ideas for collaboration. Brands' ideas are developed at length in his book, *Rethinking Public Key Infrastructures: Building in Privacy.*

John Linn of RSA offered **An Examination of Asserted PKI Issues and Proposed Alternatives.** Linn's proposed altematives are more along the lines of new ways of using X.509: Identity-Based Encryption and related approaches; online trusted third parties; distributed computation; altemative approaches to validation (hash trees in particular); key servers; and privacy protection via pseudonyms and attribute certs. Linn also noted that "you can't have full success until you've had partial success first," and that choices such as hierarchical vs. nonhierarchical PKIs — once matters of ideological controversy — are now matters of pragmatic adaptation to circumstances.

In a panel discussion on the question **Which PKI Approach for Which Application Domain?,** Peter Alterman, Carl Ellison, and Russ Weiser explored some of the specifics of this latter point. The theme of PKI not being a one-size-fits-all technology, but rather a technology that needs to be custom-tailored to a huge variety of real-world situations, has become steadily more prominent over the last couple of years, and the contrast between this session and the "Dueling Theologies" session at PKI02 (http://www.cs.dartmouth.edu/~pki02/theologies.shtml) illustrates this nicely. Ellison stated his continuing belief in the importance of local naming — not so much to avoid name collisions, which can be addressed by domain component (dc) naming, but in order to provide a means of "the relying party knowing who this is." The relying party needs to be able to use a name it assigns — a name it can remember — for a trusted entity. Ellison claims that SPKI/SDSI and XrML can do everything needed here; X.509 might work if the environment is constrained accordingly. Rich Guida (the other dueling theologian from PKI02) observed that there's increasing recognition that if you want to join a namespace, you have to choose between changing your naming or not joining; conflicts should be resolved at join-time. The problem is that you still have to have a way of knowing who others really are, what they call themselves; relying on entities to attest to the identity of others is inescapable.

Guida suggested that doctors, for instance, would never bother to assign a local name for every patient with whom they'd need to securely exchange information. This led into a discussion of PKI in medical scenarios more generally. Peter Gutmann observed that doctors confronted with PKI usually just sign in at the start of the day and let everyone else use the machine. Doctors rightly don't want anything in the way of their work; you have to design around the fact that they see any kind of security as an impediment. PDAs that transmit certificates to the network, and short-range RFIDs, were suggested as approaches to security in emergency rooms and similar settings. Guida suggested that PKI will be used a lot more in the context of medical research and clinical trials, where there isn't the "get the certificate vs. restart the patient's heart" problem, but where there is a strong need to ensure data authenticity, integrity and confidentiality. Another possible application is finding out if a suspected drug-of-abuse-seeking patient has been to other clinics. Ellison pointed out that this use case requires an aggregator, but — contrary to common perception — doesn't require X.509, or any other particular variety of PKI. No global name for the patient is needed; what matters is that the aggregator have one, and only one, key for each patient.

Ken Klingenstein keynoted on **A New and Improved Unified Field Theory of Trust.** Klingenstein identified three spheres in which individuals require trust: personal, work, and transactions where extremely high assurance is needed (often transactions involving the government). For each of these, there is a type of trust relationship which is usually appropriate: peer-to-peer, federations, and hierarchical PKI, respectively. Virtual organizations cut across these boundaries and thereby represent an additional challenge. Klingenstein described P2P trust as "a bedrock of human existence;" expressing it in electronic form is therefore necessary. It's also hard, although PGP, webs of trust, and X.509 proxy certificates have made some progress. Federations are getting deployed; Merck has a large and noteworthy deployment. Klingenstein noted that the federation structure for InCommon will be per-nation, as attitudes and practices for security are nation- and culture-specific. InCommon is hoping to set up a virtuous circle between the use of trust and the strengthening of trust. Klingenstein also offered an overview of recent developments and ongoing projects such as Stanford's Signet, Penn State's LionShare, and Internet2's own Shibboleth, setting them in the context of his unified field theory, and noted four looming issues he expects to be prominent in his talk next year: inter-federation issues, virtual organizations over P2P trust, federated and progressive (growing trust levels) PKI, and middleware diagnostics.

Jean Pawluk, representing the OASIS PKI Technical Committee and PKI Action Plan coauthor Steve Hanna, presented on **Identifying and Overcoming Obstacles to PKI Deployment and Usage.** While the Technical Committee's research identified numerous obstacles to PKI deployment, the top four (Software Applications Don't Support It; Costs Too High; PKI Poorly Understood; Too Much Focus on Technology, Not Enough On Need) accounted for half the total points survey respondents assigned to indicate relative importance. The PKI Action Plan's five action items are:

- *Develop Application Guidelines for PKI Use.* This is of particular importance for the three most popular applications: document signing, secure email, and ecommerce, in that order.
- *Increase Testing to Improve Interoperability.* Again, the focus needs to be on the top three applications. Pawluk noted that smartcard implementations in particular are very vendor-dependent. She also noted the need to coordinate work so we don't have proliferating standards, which is a huge problem — bad experiences with this give people "a jaundiced view" of standards in general.
- *Ask Application Vendors What They Need.*
- *Gather and Supplement Educational Materials on PKI.* Pawluk stressed the near-complete absence of user understanding — most users have no understanding of PKI beyond "secret codes."
- *Explore Ways to Lower Costs.* Disseminating best practices is of particular interest here.

As in other sessions, prominent themes of the discussion were that technology is a much smaller part of the problem than understanding the business needs of PKI implementers and selecting tools accordingly, and that when this is done, PKI can thrive. Bill Burr observed that the math in PKI is so cool that we try to bring everything up to its standard; instead we need to figure out how people can use PKI without understanding any of the esoteric details. Rich Guida noted that he sometimes feels like he and all the people who talk about the death of PKI dwell on "different planets;" in the pharmaceutical sector in particular, the use of PKI is "blossoming." Pawluk encouraged the group to get involved in the work of implementing the PKI Action Plan, and noted that the OASIS PKI Technical Committee that's driving it (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=PKI) usually meets via telephone.

This session was followed by a panel discussion focused on the theme: **The PKI Action Plan: Will it make a difference?** The consensus appears to be "yes, if...", with the ifs being a little different for each presenter. Sean Smith put forward his own top-three list of PKI commandments: 3) follow real-world trust flows, 2) pay proper attention to human factors, and 1) keep the goals of using PKI in mind. John Linn observed that a key question is whether deployment barriers are in PKI itself or in understanding what it can do. Most documentation is little-used and needs to be radically simplified. Linn also stressed the importance of building in reusability across applications. Lieutenant Commander Thomas Winnenberg, chief engineer for the DoD PKI, observed that the DoD PKI has been groundbreaking in that there was no ROI concern, allowing the project to be driven by an understanding of the need for vendor-neutrality and versatility in security functions. Their big problems have been around certificate issuance, but the DoD PKI now has about four million certificates in circulation. Winnenberg stressed that the focus has to be on infrastructure — relying parties are looking to PKI for a wide variety of needs, so implementations must abstract from applications. This makes "Ask Application Vendors What They Need" a key element of the PKI Action Plan. Tim Polk stressed the importance of an iterative process of application and revision of the Action Plan. Coordination will be key (in particular liaison work with groups that don't join OASIS), as will expansion of the action items into specific, concrete tasks.

Panelist Steve Hanna asked the group for further thoughts on coordination mechanisms. Tim Polk suggested making sure that IETF meeting minutes make it to the right groups; and more generally, pushing minutes of the various groups involved out to each other, rather than relying on everyone to check up on everyone else's work. Hanna suggested that liaisons also be set up between OASIS and similar groups elsewhere in the world. Hanna also asked for thoughts on how to achieve the universally-cited goal of keeping deployment efforts focused on needs rather than technology, therefore simpler ("brass-plated," as Polk put it) whenever possible. Focusing on business needs and ROI, reusability of infrastructure across applications, and applications that make it hard for the user to do the wrong thing, were all suggested here. Russ Weiser noted that often applications are something like "I have to sign something once a year;" he suggested implementing things like this in parallel with things where security need not be as stringent but that have to be done often, like submitting a timesheet. The idea is to pick the low-hanging fruit to further usability, without worrying too much about security. With respect to reusability, Polk noted that he's become a fan of the badge/token/certificate combo — if users can't get into the building without it, they'll have it with them, and then they can use it for other things. Polk also noted that NIST has been working on a PKIX test suite and client requirements for path validation; watch http://csrc.nist.gov.

Conclusions

Clearly, PKI is not dead. Throughout the workshop, presenters noted the contrast between the prevailing gloomy mood and the success of their own projects. The two overarching conclusions appear to be:

1) *Understanding and educating users is centrally important.* In particular, it is crucial a) to identify the smallest possible set of things that users need to know — the things that are inherent in the nature of PKI, b) to build systems that don't require users to know more than those things, and c) to find effective ways to teach them those things.

2) *The specifics of any particular PKI deployment should be driven by real needs, and should be only as heavyweight as necessary.* The Juricert deployment is exemplary here: it was driven by the need to stop paper forgeries, avoid re-keying, and improve transaction time, and was informed by a philosophy of "PKE" — Public Key Enough.

It was in the light of this consensus that the group met to consider the future of the PKI R&D Workshop itself.

Whither PKI0x?

There was broad agreement on keeping the current R&D focus of the workshop, with particular emphases following from the conclusions above: more on human factors, and more on using the many tools available to support a wide variety of needs and architectures. With respect to the latter, attendees would like to have more of a vendor presence at the meeting — application vendors in particular. The idea would be for the PKI0x community to get a better idea of what it can do to help vendors implement the perspective developed in the course of the workshops; ideally this would become a year-round dialogue. The group would also like to hear more about international experiences and concems, e.g. a European report on deploying a national ID card. Finally, there was agreement that the execution of the workshop needs to be tightened up: getting proceedings out more quickly and making them more visible, and publicizing the workshop more widely and further in advance.

Work in Progress Session

Ben Chinowsky, *Internet2*

Public Key Infrastructure (X.509) Library [libpkix]

Steve Hanna, Sun Microsystems

Steve presented libpkix (http://libpkix.sourceforge.net), an extensible C library for building and validating cert paths. They are looking for project participants. Various research questions are involved; one of particular importance is how you limit the amount of effort expended on pathbuilding. The Mozilla developers are very interested in this work.

The Bear Project

Sean Smith, Dartmouth College

Sean discussed Bear (http://www.cs.dartmouth.edu/~sws/abstracts/msmw03.shtml). This work is designed to address the question, "why should you trust computing that happens somewhere else?" For example, why should I trust a Shibboleth attribute authority to be giving my attributes only to the right people? The client wants to only have to provide a cert; the server doesn't want to spend money, and wants easy maintenance and good performance. The IBM 4758 doesn't solve the server problems, as it's expensive and awkward to code for. The Bear project attempts to provide 4758-like functionality on a standard machine equipped with version 1.1b of the TCPA/TCG TPM (e.g., many IBM NetVistas). Bear is now running with OpenCA in the lab; the code is at http://enforcer.sourceforge.net. Smith noted some weaknesses: Bear is probably vulnerable to power analysis; unprotected systems between the client and server will create vulnerabilities; and there are probably holes in the OS code. AEGIS could address some of these weaknesses. A revised and updated Bear paper will appear in ACSAC in December 2004.

Domain Name System Security (DNSSEC) Update

Sam Weiler, SPARTA

Finally, Sam followed up last year's WIP session on DNSSEC with a discussion of what it will take to motivate DNSSEC deployment. Security is expensive to implement, and Weiler pointed out that with security you're basically "buying brittleness" anyway. Three positions were expressed:

- Mary Thompson was the most optimistic, predicting that people will adopt DNSSEC once the technology is mature.
- Steve Hanna suggested spam as a driver, using DNSSEC to authenticate senders or MTAs. Also, some communities (e.g. ISPs) might be interested in authenticating messages to know that they're coming from within the community.
- John Linn suggested that as security is largely about assurance that information is really accurate, well-publicized DNS spoofing might get people appropriately worried. This position found the greatest resonance in the group as a whole. No one knows of any good data on how much DNS hijacking there is; it was observed that security on the web has been driven forward because there have been attacks on web sites that have cost people money, and these attacks have been well-publicized. With DNS, the attacks have probably happened, but they have not been publicized. Neal McBurnett noted that one tactic that's been used to raise security awareness is to listen to the network at a conference and publicize all the passwords discovered; maybe we need to do something similar with DNS, using a tool such as dnsspoof.

Role Sharing in Password-Enabled PKI

Xunhua Wang[†] Samuel Redwine
Commonwealth Information Security Center &
Department of Computer Science
James Madison University
Harrisonburg, VA 22807 USA
{*wangxx, redwinst*}*@jmu.edu*

Abstract

Password-enabled PKI schemes simplify the management of end users' private keys by storing them in password-protected form on a centralized on-line server. Under such schemes an end user needs only remember his password and can access his private key from anywhere the centralized server is available. Existing password-enabled PKI schemes are based on the single-user model where a private key is owned by one user. In this article, we present mechanisms to support role sharing in password-enabled PKI. In our schemes, using passwords only, a group of users share the privileges of a role through sharing the private key of that role. We first develop a hybrid password-enabled PKI scheme, which supports both easy password change and misuse monitoring. Then, based on this hybrid and existing password-enabled PKI schemes, we give password-enabled role sharing schemes for both threshold access structures that require a threshold number of these users to execute the shared role and more general access structures that allow more flexible role sharing policies.

Keywords: Role sharing, Password-enabled PKI, PAKE

1 Introduction

In a password-enabled PKI scheme [,], the private key of an end user is not stored on a smart-card or on the user's laptop. Instead, it is protected by a password chosen by the user and stored on a centralized online server. Compared to the conventional smartcard-based PKI approach, password-enabled PKI is a lightweight solution and enjoys high usability: no smartcard reader is required; an end user needs only remember his password and can roam anywhere the centralized server is available.

There are two different approaches for password-enabled PKI, *virtual soft token* [,] and *virtual smartcard* [ ]. In the virtual soft token PKI [,], a password is used to encrypt the private key of a public/private key pair and the encrypted private key is stored on a centralized server. With his password, a user can remotely authenticate himself to the server, establish an authenticated and cryptographically strong session key (thus, a secure connection) with the server, download the encrypted private key via the secure channel, decrypt it and use the private key as in the conventional PKI activities. The first step of this approach authenticates a user before he can download a password-encrypted private key and the second step establishes a session key to protect the subsequent downloading of the password-encrypted private key from the off-line

[†]A part of this work has been supported by a Cisco CIAG grant.

dictionary attack [ ]. These two steps can be accomplished by a *password-authenticated key exchange (PAKE)* protocol [, ,].

In the virtual smartcard PKI [ ], an end user's private key is split into two parts, a *human memorizable password* and a *server component.* The end user holds the password and the server component is stored on a server. Like in the virtual soft token, to use his private key, a user of the virtual smartcard PKI first runs the PAKE protocol with the server to have mutual authentication and establish a secure channel. Then, the user applies his password to a message (either a message to be digitally signed or a ciphertext to be decrypted) and sends the partial result to the server over the secure channel. The server combines its own partial result, computed from the corresponding server component and the message, with the user's partial result to generate the final result. The major difference between virtual soft token schemes and virtual smartcard schemes is that, in virtual smartcard schemes, every cryptographic operation (such as digital signature and decryption) requires the cooperation of the centralized server while in virtual soft token schemes an end user can do many cryptographic operations as he wants after securely downloading his private key.

The problem. All existing password-enabled PKI schemes [, ,] are based on the one-user-one-private-key model and are essentially *single user-oriented.* That is, they do not support multiple-users-one-private-key and thus do not support *role sharing.*

A *role*, in the access control community, is defined as a basic semantic unit to describe the authority and responsibility that users of that role assume [ ]. A good organization-wide access control decision is often based on roles (for example, president of AOL), instead of any specific individual user, as users may change over time while roles change less frequently. In many role-based access control models [,], a user assigned to a specific role is implicitly granted all the privileges of that role. However, as pointed out in [ ], within an organization, the responsibility of a role is not always assumed by any single individual but sometimes is shared among a group of users of that organization. To execute the role privileges, a subgroup of these users are required to agree on the action. In this way, power abuse by a single user or a small coalition of these users can be prevented and the principle of separation of duty can be guaranteed. We consider the case where a public/private key pair, called *role public/private key pair*, is affiliated with the role. The role public key is used by external users to encrypt messages intended for this role or verify messages digitally signed by the role private key. For users external to the role, what they see is the role itself and the users assigned to this role are invisible to them. Possibly, when collectively executing the role privileges, the users sharing a role do not necessarily trust each other.

Based on the above observation, in this article, we explore password-enabled PKI schemes to support role sharing, which are called *password-enabled role sharing PKI.*

Our contribution. We first propose a new password-enabled PKI scheme, called *hybrid password-enabled PKI*, which is later extended to support role sharing. Compared to existing password-enabled PKI schemes [, ,], this hybrid scheme allows server administrators to perform instant revocation of a user's public key and to monitor user PKI activities for misuse detection and, at the same time, it supports user password change very well. (Previous password-enabled PKI schemes support only one of these two features.) Then, we propose *password-enabled role sharing PKI* schemes for both threshold access structure and general access structure. In the scheme for threshold access structure, which is called *threshold password-enabled role sharing PKI*, a group of (say n) users are assigned to a role (and thus share the role private key), each with his favorite password and nothing else, and a threshold (say, t, $t \leq n$) of them are required to cooperate to execute the role privileges without reconstructing the shared role private key at any single location. Like the traditional single user-oriented password-enabled PKI schemes, our architecture adopts a central server, where password-protected credentials are stored. A subset of users fewer than t, together with the centralized server, will not be able to use the role private key directly. In

Table 1: Comparison of our work with previous research

Single User-oriented Password-enabled PKI		Password-enabled Role Sharing PKI	
Name	**Property**	**Name**	**Property**
Virtual soft token	easy password change	Threshold virtual soft token‡	easy password change
		Type-1 password-enabled role sharing PKI for general access structure‡	
Virtual smartcard	misuse detection		
Hybrid password-enabled PKI‡	easy password change & misuse detection	Threshold hybrid password-enabled PKI‡	easy password change & misuse detection
		Type-2 password-enabled role sharing PKI for general access structure‡	

this article we also propose password-enabled role sharing schemes to support more general access structure, in which more flexible role sharing policies are allowed.

It should be noted that our role-sharing schemes are different from the voting-based role sharing approach, where a *fully trusted* centralized server checks the votes from a subgroup of users and, if a certain condition is met, executes the role privilege. In our schemes, the centralized server is not fully trusted and the server itself alone cannot directly execute the role. Thus, neither the central server administrator nor an attacker who has successfully compromised the server can assume the role directly. Table 1 gives comparison between this work (marked with ‡) and previous research. The first two columns of table 1 give the single user-oriented password-enabled PKI schemes, including the hybrid password-enabled PKI proposed in this paper, and the last two columns of the table list their corresponding extensions for role sharing.

The remainder of this article is organized as follows. Section 2 gives the related work. Section 3 presents a hybrid password-enabled PKI scheme. Section 4 discusses some principles for designing role sharing password-enabled PKI schemes. Section 5 presents our role sharing password-enabled PKI schemes for threshold access structure and 6 gives our role sharing password-enabled PKI schemes for general access structures. In Section 7 we discuss some operational and performance issues. Concluding remarks are given in Section 8.

2 Related Work

Desmedt [ ] first proposed the concept of group-oriented cryptography to allow a threshold number of users sharing a group private key. In all the threshold cryptography schemes, including those threshold RSA [, , , , ,] and threshold DSS [, ,] schemes, each user of the role is assigned one or more *long* random secret shares of the role private key. Since most human being are not good at memorizing long random secrets and smart-cards have not been widely used yet, so far these threshold cryptography schemes have only been used in machine-oriented applications [, ,], not people-oriented systems. In contrast, the schemes explored in this article are password-based and thus, people-oriented.

Ganesan [ ] first introduced passwords into the 2-out-of-2 threshold RSA [ ] and used it to enhance the Kerberos system. We notice that this enhancement, like [ ], is still single-user oriented and does not support role sharing.

Using a 2-out-of-2 threshold RSA scheme Boneh et al. [ ] proposed an architecture for fast public key revocation. In their architecture is a semi-trusted mediator (SEM) who can monitor a user's

PKI activities. Compared to this scheme, our hybrid password-enabled PKI scheme (presented in Section 3) is password-enabled and thus enjoys better usability.

3 A New Password-Enabled PKI Scheme

The virtual soft token PKI scheme proposed in [30] allows its users to change their password easily. However, the administrators of its centralized server cannot monitor users' PKI activities as a user can perform many PKI operations after downloading his private key. On the other hand, the virtual smartcard PKI scheme proposed in [12] allows the administrators of the centralized server to monitor user PKI activities and supports instant public key revocation. However, as observed in [28], user password change is not supported very well as it is computation intensive.

In this section, we propose a hybrid password-enabled PKI scheme that supports both PKI activity monitoring and simple password change. The essential idea behind the hybrid password-enabled PKI scheme is that an additive 2-out-of-2 secret sharing is performed on the user's private key first and one of the two resulting shares, called the *server component*, is assigned to the centralized server. The other share, called the *client component*, is assigned to the user and is encrypted with the user's password and stored on the centralized server. When the user needs to use his private key, he securely downloads the password-encrypted key share, as done in virtual soft token, decrypts the key share and uses it to compute a partial result. To get the final result, the user needs the cooperation of the centralized server, which uses the server component as in the virtual smartcard scheme. Thus, this hybrid password-enabled PKI scheme is similar to the virtual smartcard scheme [12] in that the centralized server is also assigned a component of the user's private key; on the other hand, it is also similar to the virtual soft token [30] in that a user needs to download a password-encrypted credential to perform the client-side computation, which makes password change simpler.

Below we give the details of the RSA-type hybrid password-enabled PKI scheme. The same idea can be used to build DSA-type hybrid password-enabled PKI scheme but it is more complicated [28].

RSA-type hybrid password-enabled PKI Assume that Alice is a user of the hybrid password-enabled PKI scheme and her RSA public key is (N, e), where $N = p \times q$, p and q are two primes. d is Alice's corresponding private key.

- Component generation. In our hybrid password-enabled PKI scheme, the centralized server picks a random r, $1 \leq r \leq \phi(N)$ where $\phi(N) = (p-1) \times (q-1)$, and computes $r' = d - r \bmod \phi(N)$. r is the server component and r' is the client component. Alice picks her favorite password $\hat{p}$ and uses it to encrypt r'. The password-encrypted result, $y = E_{\hat{p}}(r')$, is stored on the server. For Alice, the centralized server also stores a password verification data which is a value derived from $\hat{p}$ and is used by the server to run a PAKE protocol with Alice.

- Private key use. Armed with her password, $\hat{p}$, Alice runs a PAKE protocol with the centralized server and establishes a secure channel. She then securely downloads y and decrypts at the client side to recover r'. To use her private key to perform a cryptographic operation on a message m, Alice first applies r' to m to get a partial result $c_1 = m^{r'} \bmod N$. c_1 is sent to the centralized server via the secure channel and the server applies its r to m to get $c_2 = m^r \bmod N$. The final result is $c_1 \times c_2 \bmod N$, which is equivalent to applying Alice's private key on message m.

In the above process, the centralized server is required to participate in the computation, which allows the centralized server administrator to monitor users' PKI activities and do instant public key revocation. On the other hand, Alice can change her password $\hat{p}$ to another password $\bar{p}$ by downloading y, recovering r', computing $y' = E_{\bar{p}}(r)$ and sending it to the centralized server. None of these steps is computation intensive and can be simply performed.

It is worth mentioning that in this hybrid scheme, every cryptographic operation related to the private key requires interactions with the centralized server. In contrast, with a virtual soft token, a user can load his private key onto the laptop and work offline, decrypting emails and signing new messages with no further interactions with the server.

4 Design Principles for Password-enabled Role Sharing PKI

In the remainder of this paper, PU_{role} and PK_{role} are used to denote the role public key and the role private key respectively. n is the size of the group of users to share the role and we use $\mathcal{P} = \{U_1, U_2, \ldots, U_n\}$ to denote the set of the users. An *authorized subset* is defined as a subset of $\mathcal{P}$ whose users are allowed to collectively execute the role privileges and an *access structure*, Γ, for the role is the set of authorized subsets [, pages 331].

4.1 Centralized server

Besides the users to share a role, in our architecture, there is a centralized on-line server, as in the traditional single user-oriented password-enabled PKI schemes [, ,]. It is this on-line server that makes password-enabling possible. On the other hand, this on-line server is not fully trusted in the sense that role-related credentials are *not* stored in the clear on it, but protected by passwords, and the private credentials are never exposed on the server. This distinction differentiates both the traditional password-enabled PKI and our schemes from the voting-based approach where the server is fully trusted.

For each user sharing a role, after he picks a password, the centralized server also stores the corresponding password verification data (PVD) for that user.

4.2 Design principles

There are several design principles for our password-enabled role sharing schemes. Some are straightforward while others are not.

1. The role public/private key pair does not change as often as the users assigned to the role. This is the rationale for role-based access control and is also true in our role sharing password-enabled PKI schemes.

2. A user revoked from a role should *not* know the shared role private key. Nor do a small coalition of users who have been revoked from the same role and who are not in the access structure anymore. Obviously, virtual soft token [ ] does not meet this principle.

3. Users sharing a role possess passwords only and nothing more. All operations related to the role need the explicit permission of users from an authorized subset.

4. No full trust is placed on the centralized server. The server should not know the role private key. Thus, its administrators or a hacker who has compromised the server cannot execute

the role privilege in a simple way. On the other hand, using what's stored on the server, the server administrators can mount off-line dictionary attacks. This characteristics is common to all password-based schemes and can be mitigated using multiple servers [18]. We notice that *not* all passwords are vulnerable to off-line dictionary attacks. Moreover, compared to the traditional single user-oriented password-enabled PKI schemes, in our role sharing schemes, it is harder for a malicious server administrator or a hacker who has taken control the server to mount off-line dictionary attacks as multiple, instead of a single, passwords are involved.

5. Both threshold access structure and general access structure should be supported. In a threshold access structure, any subset of size not less than the threshold is an authorized subset and this access structure is commonly used. On the other hand, threshold access structure is not always applicable and sometimes more general access structure is used.

5 Threshold Password-enabled PKI

In this section, we shall present *threshold password-enabled PKI* schemes. In the following discussion, t, $t \leq n$, is the threshold. We first discuss how to add role sharing support to the virtual soft token scheme [9]. We then extend the hybrid password-enabled PKI proposed in Section 3 to support role sharing.

5.1 Threshold virtual soft token

Threshold virtual soft token is the role sharing extension of the virtual soft token scheme [9]. Threshold cryptography schemes [7, 12, 20] are used to for this purpose. We have two types of threshold virtual soft tokens, the *threshold virtual RSA soft token* for RSA-type role public/private key pair and the *threshold virtual DSA soft token* for DSA-type role key.

In a threshold virtual RSA (DSA) soft token, a role RSA (DSA) public/private key pair is first generated and then shares of the role private key, PK_{role}, are generated through a (t, n) Shamir secret sharing [27], $(s_1, s_2, \ldots, s_n) \stackrel{(t,n)}{\longleftrightarrow} PK_{role} \bmod \phi(N)^*$ where s_i are the shares. Each user U_i, $1 \leq i \leq n$, picks his password, $\hat{p}_i$, and his corresponding password verification data, PVD_i, is generated and stored on the centralized server. For each user, also stored on the centralized server is y_i, the encryption of s_i by $\hat{p}_i$ (that is, $y_i = E_{\hat{p}_i}(s_i)$).

When the role privilege needs to be executed, depending on the threshold cryptography scheme employed, users' steps vary. In our following discussions we use the threshold RSA given in [23], called Sho00, and the threshold DSA scheme given in [12, 13], called GJKR96.

Threshold virtual RSA soft token To authorize a role-related operation, t or more of the n users are required. Let m be the message to be processed by the role private key. Each participating user first uses his password to run a PAKE protocol with the centralized server and establishes a secure connection; he then securely downloads the password-encrypted key share s_i, decrypts it and computes a partial result as $c_i = m^{2\Delta s_i} \bmod N$ where $\Delta = n!$ [26]; c_i is sent back to the centralized server over the secure channel. After collecting enough partial results, the centralized server combines them into the final result. The Sho00 threshold RSA is non-interactive and thus, in the role execution, users do not need to interact with others.

*For DSA, the modulus is the DSA system parameter q.

Threshold virtual DSA soft token When t or more users want to collectively authorize a role-related operation on message m, each of them first runs a PAKE protocol to log onto the centralized server, securely downloads his y_i and decrypts it as s_i. They then use the GJKR96 threshold DSA to collectively generate a DSA signature on m . The GJKR96 threshold DSA is an interactive scheme while, in our applications, interactions between users are not desirable. Fortunately, we observe that the interactive computation (all the steps until the computation of r [, pages 70]) of the GJKR96 threshold DSA scheme are message-independent and can be pre-computed. Based on this observation, in our threshold virtual DSA soft token, we can avoid user interactions by performing the message-independent interactive computations in a *partially-protected* store-and-forward way: all the broadcast messages by user U_i are sent to the centralized server in the clear, which will be forwarded to other participating users by the server, and all the *intermediate* private messages of U_i are encrypted by U_i's password before they are sent to and stored on the server (for future use). These pre-computations need *no* input from users and can be performed, without user U_i's interventions and notices, after U_i logs into the system.

In GJKR96, b, the number of users required for a threshold DSA signature, is $(2t-1)$, not t. That is, t should satisfy that $t < \frac{n}{2}$. Therefore, in our threshold virtual DSA soft token scheme, $(2t-1)$ users are required to collectively execute the shared role.

Both the threshold virtual RSA soft token and threshold virtual DSA soft token allow a user to change his password while keeping his role private key share unchanged. To change his password, U_i uses his old password to run a PAKE protocol with the server, securely downloads the key share protected by the old password, decrypts it, re-encrypts it with his new password, and securely uploads it to the server. To change his password, the user should also notify, via the secure connection, the server of his new PVD.

In the above threshold virtual soft token schemes, although the centralized server is used as a working platform, it is not assigned a share of the role private key and does not contribute to the final result.

5.2 Threshold hybrid password-enabled PKI

In a virtual smartcard scheme [ ], the centralized server is also assigned a share of the user's private key and is required to participate in the computation when the user's private key is used. This allows an administrator of the central server to monitor the use of the user's private key and to instantly disable the user's private key if his public key is revoked. (In contrast, the virtual soft token [ ] scheme does not offer this monitoring granularity since the private key is recovered and used on the user's machine.)

It is not immediately obvious on how to extend the virtual smartcard scheme given in [ ] to support password-enabled role sharing. In a (t,n) Shamir secret sharing scheme [ ], to share a secret, at most $(t-1)$ shares can be passwords. This fact prevents us from simply extending the virtual smartcard scheme for password-enabled role sharing since, ideally, in a password-enabled role sharing scheme, all the n, not just $(t-1)$, users hold their favorite passwords only and nothing else. One might think to apply the following extension to the virtual smartcard scheme: for each combination $U_{i_1}, U_{i_2}, \ldots, U_{i_t}$, where $\{i_1, i_2, \ldots, i_t\} \subset \{1, 2, \ldots, n\}$, we can compute $d_{\{i_1,i_2,\ldots,i_t\}} = d - \hat{p_{i_1}} - \hat{p_{i_2}} - \ldots - \hat{p_{i_t}} \bmod \phi(N)$, where $\hat{p_{i_j}}$ is the password of U_{i_j}, $1 \leq j \leq t$, and store $d_{\{i_1,i_2,\ldots,i_t\}}$ on the server. In this way, any t users can cooperate with the server to collectively apply the shared role private key on a message. However, this extension has a security flaw: it stores $\binom{n}{t}$ such $d_{\{i_1,i_2,\ldots,i_t\}}$ values on the server and in some cases the server will be able to restore the role private key from them, which contradicts with our design principle 4 (see Section 4).

On the other hand, the hybrid password-enabled PKI scheme proposed in Section 3 can be extended to support role sharing for threshold access structure, which allows both monitoring granularity and easy password change. The following details are based on the RSA-type hybrid password-enabled PKI give in Section 3.

- Component generation. After the role RSA public/private key pair $(PU_{role}, PK_{role} = d)$ is generated, a random r, $1 \leq r \leq \phi(N)$, is generated and r' is computed as $r' = d - r \bmod \phi(N)$. r is the server component and is stored on the centralized server. A (t, n) Shamir secret sharing is performed on the client component r', $r' \xrightarrow{(t,n)} (s_1, s_2, \ldots, s_n) \bmod \phi(N)$ and s_i is assigned to U_i, $1 \leq i \leq n$. Each user U_i picks his password, $\hat{p}_i$, and it is used to encrypt s_i into $y_i = E_{\hat{p}_i}(s_i)$. For user U_i, y_i and a password verification data derived from $\hat{p}_i$ are stored on the centralized server.

- Private key use. When t or more users agree to apply the role's private key on a message m, each of them uses his $\hat{p}_i$ to run a PAKE protocol with the centralized server and establish a secure channel. He then securely downloads y_i and decrypts at the client side to recover s_i. He then first applies s_i to m and gets a partial result $c_{1i} = m^{s_i} \bmod N$. c_{1i} is sent to the centralized server via the secure channel. The server also applies its r to m to get $c_2 = m^r \bmod N$. After collecting enough partial results, the centralized server combines all partial results, c_{1i} and c_2 into the final result, which is exactly of the role's private key on message m.

  In the above process, the centralized server is required to participate, which allows the centralized server administrator to monitor the role's PKI activities and do instant public key revocation. On the other hand, each user can change his password $\hat{p}_i$ to another password $\bar{p}_i$ by downloading y_i, recovering s_i, computing $y_i' = E_{\bar{p}_i}(s_i)$ and sending it to the centralized server. None of these steps is computation intensive and can be simply performed.

6 Password-enabled Role Sharing for General Access Structure

In our real world not all access structures are threshold-based and sometimes more general access structures are used. For example, four users, $(U1, U_2, U_3, U_4)$, share a role and $\Gamma = \{\{U_1, U_2\}, \{U_1, U_3, U_4\}\}$ is its access structure. This access structure is not threshold: $\{U_1, U_2\}$ has two members and is allowed to execute the role while $\{U_2, U_3, U_4\}$ is not allowed although its cardinality is 3.

In this section we discuss how to support password-enabled role sharing for general access structure. An access structure is said to be *monotone* if $B \in \Gamma$ and $B \subseteq C \subseteq \mathcal{P}$ implies $C \in \Gamma$ [ ]. We are only interested in monotone access structure here.

General access structure-oriented secret sharing — called *generalized secret sharing* — was first studied by Ito et al. [ ]. Benaloh and Leichter [ ] developed a simpler generalized secret sharing, which is called BL88 in the following discussion. It should be noted that password-enabled role sharing discussed here is more than secret sharing as we do not reconstruct a shared secret, as done in secret sharing schemes, since reconstruction leads to a single point of attack.

In the rest of this section we will give two types of password-enabled role sharing PKI schemes for general access structure. Our discussions are based on the RSA algorithm but can also be applied to DSA.

6.1 Type-1 password-enabled role sharing PKI

Type-1 password-enabled role sharing PKI for general access structure is the extension of the virtual soft token for role sharing.

- Component generation. Given a monotone access structure Γ for a role whose private key is $PK_{role} = d$, we first use the BL88 generalized secret sharing scheme to generate secret shares. Each of the n users will get one or more secret shares. Then, each of them picks his favorite password $\hat{p}_i$ and uses it to encrypt all of his secret shares. The password-encrypted secret shares, together with a password verification data derived from $\hat{p}_i$, are stored on the centralized server.
- Private key use. When an authorized subset of users want to execute the role privilege on a message m, each of them, U_i, runs a PAKE protocol to log onto the centralized server and establishes a secure connection with it. U_i then securely downloads his secret shares and applies it to m to get a partial result. The partial result is securely sent to the centralized server who combines all partial results into a final result, which is equivalent to applying the role's private key on m.

Using the above $\Gamma = \{\{U_1, U_2\}, \{U_1, U_3, U_4\}\}$ as an example, we have the following shares: d_1 is assigned to U_1, d_2 is assigned to U_2, d_3 is assigned to U_3, d_4 is assigned to U_4 where $d_1 + d_2 = d \bmod \phi(N)$ and $d_1 + d_3 + d_4 = d \bmod N$. When U_1 and U_2 agree to apply the role private key to m, U_1 computes $c_1 = m^{d_1} \bmod N$ and U_2 computes $c_2 = m^{d_2}$. After receiving c_1 and c_2, the centralized server combines them into the final result as $c = c_1 \times c_2 \bmod N = m^d \bmod N$, which is exactly the role private key on m.

In the above steps, the centralized server does not contribute to the final result and technically the step of combining partial results into the final result can be performed by any users. That is, type-1 password-enabled role sharing PKI does not provide a technical means to monitor role PKI activities on the centralized server.

6.2 Type-2 password-enabled role sharing PKI

Using the same idea of the hybrid password-enabled PKI, type-1 password-enabled role sharing PKI can be modified so that the centralized server is required to contribute for a role privilege execution. In the above component generation stage, instead of sharing d, we can first run a 2-out-of-2 additive secret sharing on d and get d' and d'', where $d = d' + d'' \bmod \phi(N)$. d' is the server component and is assigned to the centralized server. The client component, d'', is shared among the n users using the BL88 generalized secret sharing. Then each user uses his password to encrypt the shares assigned to him and stores the password-protected shares on the centralized server. When an authorized subset of users want to execute the role privilege, they compute their partial results. To get the final result, the centralized server is also required to participate and compute its own partial result. Thus, this modified scheme allows the centralized server administrators to monitor role PKI activities and is called *type-2 password-enabled role sharing PKI for general access structure.*

For general access structure, a user is likely to be assigned more than one secret shares and, in deciding which share to use, he needs to know the identities of others users of the authorized subset. This might be undesirable sometimes as it needs coordinations between the participating users. A method for U_i to avoid this interaction is to apply all of his secret shares to m to get more partial results than necessary. When the centralized server combines the partial results, only those necessary partial results will be used.

7 More Discussions

In this section we discuss some performance and operational issues.

7.1 Performance considerations

Compared to the virtual soft token [ ] and the virtual smartcard scheme [ ], the hybrid password-enabled PKI scheme does not introduce any additional significant computational cost.

7.2 Operational considerations

Password-based versus smartcard-based. Passwords are commonly used for authentication in our daily lives and support user roaming very well. Password-enabled PKI schemes integrate the roaming capability and good usability of passwords into PKI. However, in some application cases, smartcard-based solution might still be preferred due to its high-level security. For these applications, password-enabled PKI can be used as a short-term solution and the migration from password-based to smartcard-based can be made smooth.

In both threshold virtual soft token scheme and threshold hybrid password-enabled PKI scheme, an end user is assigned one or more shares of the role private key and the password-encrypted key shares are stored on the centralized server. This structure makes it easy for password and smartcards to co-exist and makes it easy to migrate from password-based to smartcard-based: users who prefer passwords can still hold their passwords and store their password-encrypted shares on the server; users who like smartcards can download their password-encrypted key shares and feed them to smartcards. This is also true for both type-1 and type-2 password-enabled role sharing PKI schemes.

Recovery from password loss. In a password-based system, a user might inadvertently lose his password to somebody else. For example, a user might use an insecure computer on which a key logging program is installed to harvest passwords. For single user-oriented password-enabled PKI, this might be disastrous. In contrast, the password-enabled role sharing PKI schemes tolerate this type of mistakes to some extent: as long as an attacker does not steal more than $(t-1)$ passwords, he will not be able to assume the shared role. The recovery from such loss is also straightforward: after a user loses his password, his old key share is disabled and any t other users who share the same role can help him get a new key share.

8 Conclusion

Conventional password-enabled PKI schemes are based on the one-private-key-one-user model and do not support role sharing. In this article we developed schemes to add role sharing to password-enabled PKI schemes. We first presented a hybrid password-enabled PKI scheme, which supports both easy password change and misuse monitoring. Then, we extended our hybrid and existing password-enabled PKI schemes to support role sharing. Our password-enabled role sharing PKI schemes support both threshold access structures and general access structures. Compared to conventional password-enabled PKI schemes, from an end user's perspective, our password-enabled role sharing PKI schemes do not occur additional significant computational cost and also tolerate end user's operational mistakes.

9 Acknowledgement

We wish to thank the anonymous reviewers for pointing us to the related work in [ ] and for other useful comments.

References

[1] B. Barak, A. Herzberg, D. Naor, and E. Shai. The proactive security toolkit and applications. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 18–27, November 2–4 1999.

[2] S. Bellovin and M. Merritt. Encrypted key exchange: password–based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[3] J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In S. Goldwasser, editor, *Advances in Cryptology - Crypto '88*, number 403 in Lecture Notes in Computer Science, pages 27–36, Berlin, 1988. Springer-Verlag.

[4] D. Boneh, X. Ding, G. Tsudik, and C. M. Wong. A methof for fast revocation of public key certificates and security capabilities. In *Proceedings of the 10th USENIX Security Symposium*, August 13–17 2001.

[5] C. Boyd. Some applications of multiple key ciphers. In C. G. Gunther, editor, *Advances in Cryptology, Proc. of Eurocrypt '88*, volume 330 of *Lecture Notes in Computer Science*, pages 455–467, Davos, Switzerland, May 1988. Springer-Verlag.

[6] Y. Desmedt. Society and group oriented cryptography: a new concept. In *Advances in Cryptology, Proc. of Crypto '87*, pages 120–127, August 16–20 1988.

[7] Y. Desmedt. Some recent research aspects of threshold cryptography. In E. Okamoto, G. Davida, and M. Mambo, editors, *Information Security*, volume 1396 of *Lecture Notes in Computer Science*, pages 158–173, September 1997. URL `http://www.cs.fsu.edu/~desmedt/ISW.pdf`.

[8] Y. Desmedt, G. Di Crescenzo, and M. Burmester. Multiplicative nonabelian sharing schemes and their application to threshold cryptography. In *Advances in Cryptology — Asiacrypt '94*, pages 21–32, November/December 1994.

[9] Y. G. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics*, 7(4):667–679, November 1994.

[10] Y. Frankel and Y. Desmedt. Parallel reliable threshold multisignature. Tech. Report TR–92–04–02, Dept. of EE & CS, Univ. of Wisconsin–Milwaukee, April 1992. ftp://ftp.cs.uwm.edu/pub/tech_reports/desmedt-rsa-threshold_92.ps.

[11] R. Ganesan. Yaksha: Augmenting Kerberos with public-key cryptography. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium*, 1995.

[12] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Advances in Cryptology — Eurocrypt '96*, pages 354–371, May 12–16 1996.

[13] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Information and Computation*, 164(1):54–84, 2001.

[14] N. Gilboa. Two party RSA key generation. In M. Wiener, editor, *Advances in Cryptology - CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 116–129, Santa Barbara, California, USA, August 1999.

[15] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. In *Proc. IEEE Global Telecommunications Conf., Globecom'87*, pages 99–102. IEEE Communications Soc. Press, 1987.

[16] D. P. Jablon. Strong password–only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, October 1996.

[17] T. Kwon. Virtual software tokens - a practical way to secure PKI roaming. In G. Davida, Y. Frankel, and O. Rees, editors, *Proceedings of the Infrastructure Security (InfraSec)*, volume 2437 of *Lecture Notes in Computer Science*, pages 288–302. Springer-Verlag, 2002.

[18] P. MacKenzie and M. Reiter. Two-party generation of DSA signatures (extended abstract). In J. Kilian, editor, *Advance in Cryptology – EUROCRYPT 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 137–154, Santa Barbara, CA, USA, August 2001. Springer.

[19] R. Morris and K. Thompson. Password security: a case history. *Communications of the ACM*, 22(11):594–597, November 1979.

[20] R. Perlman and C. Kaufman. Secure password-based protocol for downloading a private key. In *Proceedings of the ISOC Network and Distributed Systems Security Symposium*, 1999.

[21] T. Rabin. A simplified approach to threshold and proactive RSA. In *Advances in Cryptology, Proc. of Crypto'98*, pages 89–104, August 23-27 1998.

[22] R. Sandhu, M. Bellare, and R. Ganesan. Password enabled PKI: Virtual smartcards vs. virtual soft tokens. In *Proceedings of the 1st Annual PKI Research Workshop*, pages 89–96, April 2002.

[23] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and Systems Security*, 2(1):105–135, February 1999.

[24] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[25] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[26] V. Shoup. Practical threshold signatures. In *Advance in Cryptology – EUROCRYPT 2000*, pages 207–220, May 2000.

[27] D. R. Stinson. *Cryptography: Theory and Practice*. CRC, Boca Raton, 1st edition, 1995.

[28] X. Wang. Intrusion-tolerant password-enabled PKI. In *Proceedings of the 2nd Annual PKI Research Workshop*, pages 44–53, Gaithersburg, MD, USA, April 28–29 2003. Natl Inst. of Standards and Technology.

[29] X. Wang, Y. Huang, Y. Desmedt, and D. Rine. Enabling secure on-line DNS dynamic update. In *Proceedings of the* 16^{th} *Annual Computer Security Applications Conference*, pages 52–58, New Orleans, Louisiana, USA, December 11-15 2000. IEEE Computer Society Press.

[30] T. Wu. The secure remote password protocol. In *Proceedings of the 1998 Network and Distributed System Security Symposium*, pages 97–111, 1998.

[31] T. Wu, M. Malkin, and D. Boneh. Building intrusion tolerant applications. In *Proceedings of the 8th USENIX Security Symposium*, pages 79–91, 1999.

Greenpass: Decentralized, PKI-based Authorization for Wireless LANs*

Nicholas C. Goffee, Sung Hoon Kim, Sean Smith, Punch Taylor,
Meiyuan Zhao, John Marchesini

Department of Computer Science/Dartmouth PKI Lab†
Dartmouth College
Hanover, New Hampshire USA

Abstract

In Dartmouth's "Greenpass" project, we're building an experimental system to explore two levels of authorization issues in the emerging information infrastructure. On a practical level, we want to enable only authorized users to access an internal wireless network—while also permitting appropriate users to delegate internal access to external guests, and doing this all with standard client software. On a deeper level, PKI needs to be part of this emerging information infrastructure—since sharing secrets is not workable. However, the traditional approach to PKI—with a centralized hierarchy based on global names and heavy-weight X.509 certificates—has often proved cumbersome. On this level, we want to explore alternative PKI structures that might overcome these barriers.

By using SPKI/SDSI delegation on top of X.509 certificates within EAP-TLS authentication, we provide a flexible, decentralized solution to guest access that reflects real-world authorization flow, without requiring guests to download nonstandard client software. Within the "living laboratory" of Dartmouth's wireless network, this project lets us solve a real problem with wireless networking, while also experimenting with trust flows and testing the limits of current tools.

1 Introduction

Dartmouth College is currently developing *Greenpass*, a software-based solution to wireless network security in large institutions. Greenpass extends current wireless security frameworks to allow guest access to an institution's wireless network and selected internal resources (as well as to the guest's home system).

This project, which enhances EAP-TLS authentication with SPKI/SDSI-based authorization decisions, is a novel, extensible, feasible solution to an important problem.

- Our solution is **seamless.** Guests can potentially access the same access points and resources that local users can. The same authorization mechanism can apply to local users, and can also be used for application-level and wired resources.
- Our solution is also **decentralized**: it can accommodate the way that authorization really flows in large academic organizations, allowing designated individuals to delegate network access to guests.

Although we are initially targeting universities, Greenpass may apply equally well in large enterprises.

This paper. This paper provides a snapshot of the current state of our project. Section 2 reviews the problem we seek to solve, and Section 3 reviews the existing wireless security standards we build upon. Section 4 presents weaknesses in some current attempts to secure wireless networks. Section 5 presents our approach: Section 6 discusses the delegation piece, and Section 7 discusses the access decision piece. Section 9 discusses future directions, and Section 10 offers some concluding remarks.

More lengthy discussions (e.g., [Gof04, Kim04]) of this

* A previous version of this paper appeared in Dartmouth Dept. of Computer Science technical report TR2004-484

†This research has been supported in part by Cisco Corporation, the Mellon Foundation, NSF (CCR-0209144), AT&T/Internet2 and the Office for Domestic Preparedness, Department of Homeland Security (2000-DT-CX-K001). This paper does not necessarily reflect the views of the sponsors. The authors can be reached via addresses of the form `firstname.lastname@dartmouth.edu`

work will appear this Spring.[1]

2 The Problem

Wireless network access is ubiquitous at Dartmouth, and we see a future where a lack of wireless network access at a university—including access for visitors to the campus—is as unthinkable as a lack of electricity. Many institutions, however, want to restrict access to their wireless networks for several reasons: the cost of providing network bandwidth and resources; the credibility or liability hit the institution may incur should an outside adversary use the network to launch an attack or spam; the ability to block users who have not installed critical security patches; and the ability (for reasons of license as well as levels-of-protection) to restrict certain local network resources to local users.

Access to a wired network often depends, implicitly, on the physical boundaries of the network. Most establishments do not install courtesy network jacks on the outside walls of their buildings: a standard door, therefore, fulfills most access-control needs. Wireless network traffic, on the other hand, travels on radio waves, extending the network's physical boundaries. Access control and encryption must be designed into the link layer or higher to prevent unauthorized use and/or eavesdropping.

This future raises some challenges:

- We need to permit authorized local users to access the network.
- We also need to permit selected guests to access the network.
- We must minimize the hassle needed to grant access to guests, and we must accommodate the decentralized ways that authority really flows in large organizations.
- The security should cause little or no additional effort when regular users and guests use the network.
- The type of guests and the manner in which they are authorized will vary widely among the units within an institution.
- We must accommodate multiple client platforms.
- The solution must scale to large settings, more general access policies, and decentralized servers.
- The solution should also extend to *all* authorization—wired or wireless, network or application, guest or intra-institution.
- The solution must be robust against a wide range of failures and attacks.

We have encountered several definitions of "guest access" to a wireless network, many of which differ substantially from our own. Two basic definitions we have seen are as follows:

- **Definition 1**. The trivial solution: the network is open and all passersby, even uninvited ones, can potentially become "guests."
- **Definition 2**. Insiders can connect to a *VPN* (*virtual private network*) or to the inside of a firewall, allowing them to access private resources. Guests have basic wireless access—perhaps with a bridge to the Internet—but remain outside the firewall or VPN.

Our requirements for guest access, on the other hand, are as follows:

- **Definition 3.** We want guests to access the inside; that's the whole point. But we also need to control who becomes a "guest," and we want to permit authorization to flow the way it does in the real world: we don't want a centralized authority (or a central box and rights system purchased from a single vendor) controlling all end-user decisions.

3 Background

Wireless networking comes in two basic flavors. In the *ad hoc* approach, the wireless stations (user devices) talk to each other; in the *infrastructure* approach, wireless stations connect to access points, which usually serve as bridges to a wired network. We are primarily interested in infrastructure networking. Understanding the numerous protocols for access control in infrastructure mode requires wading through an alphabet soup of interrelated standards and drafts. We will provide a brief overview of these standards in this section; Edney and Arbaugh's recent book [EA03] explores them thoroughly.

Rudimentary access control to a WLAN could be implemented by requiring users to enter the correct *SSID* for the access point they are trying to connect to, or by accepting only clients whose MAC addresses appear on an *access-control list* (*ACL*). Both these techniques are easily defeatable, as we discuss below in Section 4.

Wired equivalent privacy (WEP) is a link-layer encryption method offered by the original IEEE 802.11 wireless Ethernet standard. WEP is based on a shared secret between

[1] As of press time, the cited theses have been published as technical reports.

the mobile device and access point. WEP has numerous flaws, which Cam-Winget et al. [CWHWW03] and Borisov et al. [BGW01] discuss in detail.

Wi-Fi Protected Access (WPA) is a stronger authentication and encryption standard released by the Wi-Fi Alliance, a consortium of vendors of 802.11-based products. WPA is, in turn, a subset of *802.11i*, the IEEE draft that standardizes future 802.11 security. WPA provides an acceptable security standard for the present, until 802.11i is finalized and becomes widely supported.

WPA and 802.11i both use *802.1x* [CS01], a general access-control mechanism for any Ethernet-based network. 802.1x generalizes the *Extensible Authentication Protocol (EAP)* [BV98], originally designed for authentication of PPP dialup sessions.

In a wireless context, 802.1x access control works as follows:

- By trying to connect to an access point, a mobile device assumes the role of *supplicant*.
- The access point establishes a connection to an *authentication server*.
- The access point (which, in 802.1x terminology, assumes the role of *authenticator*) relays messages back and forth between the supplicant and the authentication server. These relayed messages conform to the EAP packet format.
- EAP can encapsulate any of a variety of inner authentication handshakes including challenge-response schemes, password-based schemes (e.g., Cisco's LEAP), Kerberos, and PKI-based methods. The supplicant and authentication server carry out one of these handshakes.
- The authentication server decides whether the supplicant should be allowed to connect, and notifies the access point using an *EAP-Success* or *EAP-Failure* message.

EAP-TLS. Authenticating a large user space suggests the use of public-key cryptography, since that avoids the security problems of shared secrets and the scalability problems of ACLs. One public-key authentication technique permitted within EAP is *EAP-TLS* [AS99, BV98].

TLS (Transport Layer Security) is the standardized version of *SSL (Secure Sockets Layer)*, the primary means for authentication and session security on the Web. In the Web setting, the client and server want to protect their session and possibly authenticate each other. Typically, SSL/TLS allows a Web server to present an X.509 public key certificate to the client and prove knowledge of the corresponding private key. A growing number of institutions (including Dartmouth) also exploit the ability of SSL/TLS to authenticate the client: here, the client presents an X.509 certificate to the server and proves ownership of the corresponding private key. The server can use this certificate to decide whether to grant access, and what Web content to offer. An SSL/TLS handshake also permits the client and server to negotiate a cryptographic suite and establish shared secrets for session encryption and authentication.

The EAP-TLS variant within 802.1x moves this protocol into the wireless setting. Instead of a Web client, we have the supplicant; instead of the Web server, we have the access point, working in conjunction with the authentication server.

Our approach. WPA with EAP-TLS permits us to work within the existing Wi-Fi standards, but lets the supplicant and access point evaluate each other based on public key certificates and keypairs. Rather than inventing new protocols or cryptography, we plan to use this angle—the expressive power of PKI—to solve the guest authorization problem.

4 Black Hat

As part of this project, we began exploring just how easy it is to examine wireless traffic with commodity hardware and easily-available hacker tools. Right now:

- We can watch colleagues surf the Web and read their email.
- We can read the "secret" (non-broadcast) SSID for local networks.
- We can read the MAC addresses of supplicants permitted to access the network.
- We can tell our machine (Windows or Linux) to use a MAC address of our own choosing (such as one that we just sniffed).

The lessons here include the following:

- We can easily demonstrate that security solutions that depend on secret SSIDs or authenticated MAC addresses do not work.
- The current Dartmouth wireless network is far more exposed than nearly all our users realize; the paradigm shift from the wired net has substantially changed the

security and privacy picture, but social understanding (and policy) lags behind. We suspect this is true of most wireless deployments.

We conjecture that any solution that does not use cryptography derived from entity-specific keys will be susceptible to sniffing attacks and session hijacking.

5 The Overall Approach

We have already built a basic prototype of Greenpass, and are planning a series of pilots in the near future. Our prototype consists of two basic tools. The first automates the process of issuing credentials to a guest by allowing certain local users to issue *SPKI/SDSI authorization certificates* [EFL+99a, EFL+99b] to guests. The second tool is a *RADIUS (Remote Authentication Dial In User Service) server* [Rig00, RWC00, RWRS00] that carries out a standard EAP-TLS handshake for authentication, but has been modified to consult SPKI/SDSI certificates for authorization of non-local users. Neither tool requires users to have software beyond what is typically installed on a Windows laptop (covering most of our user space); other platforms need 802.1x supplicant software, which is provided with recent versions of Mac OS X and is readily available for Linux.

Authorization in real life. In the physical world, a guest gets access to a physical resource because, according to the local policy governing that resource, someone who has the power to do so said it was OK. In a simple scenario, Alice is allowed to enter a lab because she works for Dartmouth; guest Gary is allowed to come in because Alice said it was OK. Gary's authorization is decentralized (Dartmouth's President Wright doesn't know about it) and temporary (it vanishes tomorrow). More complex scenarios also arise in the wild: e.g., Gary may only have access to certain rooms, and it must be Alice (and not Bob, since he doesn't work on that project) who says OK.

For a wireless network in a large institution, the decision to grant authorization will not always be made by the same Alice—and may in fact need to reflect policies and decisions by many parties. PKI can handle this, by enabling verifiable chains of assertions.

Authorization in EAP-TLS. EAP-TLS specifies a way for a RADIUS server to *authenticate* a client, but leaves open the specification of *authorization*. Often, a RADIUS server will allow any supplicant to connect who authenticates successfully—i.e., whose certificate was signed by a CA the RADIUS server has been configured to trust. This approach does not adequately reflect real-life authorization flow as just described. Alice can see to it that Gary, her guest, obtains access to the wireless network, but she must do so by asking a central administrator to issue Gary an X.509 certificate from Dartmouth's own CA. It is possible for the RADIUS server to trust multiple CAs, such as those of certain other universities, but this option remains inflexible if a guest arrives from an institution not recognized by the existing configuration. (Another option that merits further investigation, however, is the possibility of linking RADIUS servers using more advanced trust path construction and bridging techniques. One such implementation is the Trans-European Research and Education Networking Association's (TERENA) [TER] top-level European RADIUS server, a hierarchy of RADIUS servers connecting the Netherlands, the UK, Portugal, Finland, Germany, and Croatia.)

Conceivably, a RADIUS server could perform any of a number of authorization checks between the time that a supplicant authenticates successfully and the time that the RADIUS server transmits an EAP success or failure code. In other words, we can modify a RADIUS server to base its decision on some advanced authorization scheme. Policies could be defined by policy languages such as XACML; or by signed *authorization certificates* as defined by *Keynote* [BFIK99] and its predecessor *PolicyMaker* [BFL96], by the *X.509 attribute certificate* (*AC*) standard [FH02], or by SPKI/SDSI.

SPKI/SDSI. For our Greenpass prototype, we settled on SPKI/SDSI for three main reasons: (1) it focuses specifically on the problem we are trying to solve (authorization), (2) its central paradigm of *delegation* gives us precisely the decentralized approach to guest access we desire, and (3) its lightweight syntax makes it easy to process and to code for.

SPKI/SDSI differs from a traditional X.509-based PKI in three important ways:

- SPKI/SDSI uses public keys as unique identifiers: people and other *principals* are referred to as holders of particular public keys, rather than as entities with particular names.
- A SPKI/SDSI certificate binds an authorization directly to a public key, rather than binding a name to a public key (as an X.509 certificate does) or an authorization to a name (as an ACL entry or attribute certificate typically does).
- Any person or entity, not just a dedicated CA, can potentially issue a SPKI/SDSI certificate. In particular,

the recipient of a SPKI/SDSI authorization can optionally be authorized to *delegate* his privilege to further users.

SPKI/SDSI therefore solves some of the problems with guest authorization. First, even if a guest's home organization issued him an X.509 certificate, we cannot use it to authenticate the guest (i.e., bind the guest to a unique identifier) if the issuer of the certificate is not trusted. A SPKI/SDSI authorization certificate, however, binds an authorization to a particular keyholder without an intermediate naming step: therefore, we can bind credentials to a guest's public key. The public key acts as the sole identifying information for the guest ("authentication," then, means proving ownership of the key).

Additionally, SPKI/SDSI delegation provides a straightforward way to implement guest access. Dartmouth can issue Alice, a professor, a SPKI/SDSI certificate granting her the ability to delegate access to guests.[2] If Alice invites Gary to campus to deliver a guest lecture, he will probably request access to the network. Alice can simply issue Gary a short-lived SPKI/SDSI certificate (vouching for the public key in his X.509 certificate) that grants him access to the network while he is on campus. No central administrator need be contacted to fulfill Gary's request.

Alternative approaches to authorization. Other approaches to delegated guest access are available, and each has its own balance of advantages and disadvantages.

An X.509 AC can grant a short-lived authorization to the holder of a particular X.509 identity certificate, in much the same way as we use SPKI/SDSI. Attribute certificates address the problem of authorization, but are intended to be issued by small numbers of *attribute authorities* (*AAs*); the attribute certificate profile [FH02] states that chains of ACs are complex to process and administer, and therefore recommends against using them for delegation. Doing so would amount to emulating SPKI/SDSI's functionality using attribute certificates. If standard WPA clients were able to transmit attribute certificates along with identity certificates as part of the EAP-TLS handshake, we might have chosen ACs instead of SPKI/SDSI certificates, as the former would have provided a convenient means to transmit authorization information to a RADIUS server.

Two other authorization certificate systems worth considering are *PERMIS* [COB03, PER] and X.509 *proxy certificates* [TWE+03, WFK+04]. The PERMIS system uses attribute certificates to specify roles for various users; members of some roles are able to delegate their role, or a subordinate role, to other individuals. PERMIS is worth investigating as a means of wireless guest access, although it might require modification to eliminate its reliance on global names. X.509 proxy certificates provide a different means of delegation than SPKI/SDSI does, along with a concept of temporary, relative identities that local users could provide to their guests. Proxy certificates conform closely enough to the X.509 name certificate format that they might work directly in an EAP-TLS handshake.

We also could have implemented guest access by placing temporary ACL entries in a central database. "Delegation" could be implemented by allowing authorized delegators to modify certain portions of the ACL. Ultimately, however, would like to support a "push" model of delegation where guests carry any necessary credentials and present them upon demand, allowing us to further decentralize future versions of Greenpass (see Section 9). Decentralizing authorization policies using signed certificates also eliminates the need for a closely-guarded machine on which a central ACL is stored.

We chose SPKI/SDSI because it reflects, in our minds, the most straightforward model of real-world delegation. A thorough comparison of alternative approaches would provide a worthwhile direction for future work.

6 Delegation

Assume that a new guest arrives and already holds an X.509 identity certificate containing a public key. In order to obtain wireless connectivity, the guest must obtain a SPKI certificate that conveys the privilege of wireless network access directly to his own public key.

To obtain this certificate, the guest will find a local user who can delegate to him (e.g., the person who invited him in the first place). This *delegator* must then learn the guest's public key. This step requires an information path from the guest's machine to the delegator's. Once the delegator learns the guest's key, he can issue a SPKI certificate with the guest as its subject.

We also need a way to ensure that the key the delegator authorizes to use the wireless network is really the key held by the guest. Otherwise, an adversary might inject his own public key into the communication channel between guest and delegator, tricking the delegator into authorizing the adversary instead of the intended guest. Dohrmann and Ellison describe a nearly identical problem in *introducing* the members of a collaborative group to one another [DE02]. Their

[2] In our current scheme, Alice uses an X.509 certificate, signed by the local CA, to gain access to the network herself; she must obtain a SPKI/SDSI certificate only if she needs to delegate to a guest without a locally-signed X.509 certificate.

solution was to display a *visual hash* of the public key being transferred on both the keyholder's device and the recipient's device: this allows the recipient to quickly compare the two visual images, which should appear identical if and only if the recipient received a public key value identical to the one stored on the originating device. We adopted this same approach; further details are given below.

Guest interface. We chose to use a Web interface to allow a guest to introduce his public key to a delegator. Web browsers are ubiquitous: we can safely assume that any user who wishes to access our network will have a Web browser installed. This technique gives us an advantage over, e.g., infrared transfer, wired tranfer, or passing of some storage medium, all of which might be incompatible with certain client devices.

Both our delegation tool and our modified RADIUS server rely on the observation that standard X.509 certificates, and standard SSL/TLS handshakes (including EAP-TLS) perform three functions that we need:

- An X.509 certificate contains the value of its owner's public key.
- An SSL/TLS handshake *presents* an X.509 certificate (and thus the owner's public key value).
- An SSL/TLS handshake, if it succeeds, also *proves* that the authenticating party owns the private key corresponding to the subject public key in the presented X.509 certificate.

With this observation, it becomes clear that, if the guest's Web browser supports SSL/TLS client authentication, then he can present his public key value to a Web site using this functionality.

When a guest arrives he must connect to our Web application to present his existing X.509 certificate. Therefore, he must obtain *some* wireless connectivity even before he is authorized. We are experimenting with various ways to enable this by creating an "unauthorized" VLAN (for newly-arrived guests) and an "authorized" VLAN (for local users and authorized guests); we present this approach in more detail in Section 7.

When a guest connects to our Web application, he will see a welcome page helping him through the process of presenting his certificate. We handle three situations at this point:

- If the guest's Web browser presents an SSL client certificate, we allow the option of presenting it immediately.
- We also allow the guest to upload his certificate from a PEM-formatted file on his local disk. (Browser and OS keystore tools usually allow a user to export his X.509 certificate as a PEM file. Since the purpose is to transfer the certificate to another user, a PEM file typically does *not* contain the user's private key.)
- If the guest does not already have a keypair and certificate, he can connect to a "dummy" CA page (separate from the main Dartmouth CA) that lets him generate a keypair and obtain a temporary X.509 certificate. (This should not be a standard approach, because a proliferation of client keypairs impairs usability. Note that the sole purpose of the dummy CA is to get the guest a keypair—we are therefore exempt from standard CA worries such as securing the registration process and protecting the CA's private keys.)

We implemented the guest interface using simple CGI scripts served by an Apache Web server. Our installation of Apache includes *mod_ssl*, which we configure to request (but not require) SSL client authentication. (We had to set a seldom-used option in *mod_ssl* that forces Apache to accept the guest's certificate even if it was signed by an unknown CA. Our purpose here is to learn a stranger's public key, not to authenticate a known user.) Therefore, if the guest has installed a client certificate in his Web browser, it will present it to our Web server. Our CGI scripts use OpenSSL to process the guest's X.509 certificates.

The dummy CA uses the standard enrollment functionality included in Web browsers that support SSL client authentication. The guest visits the CA page and enters (possible fake) identifying information. The page includes code that, when he submits the identifying information, causes his Web browser to generate a keypair, store the private key, and submit the public key to our Web server. The dummy CA then issues a new X.509 certificate back to the guest's Web browser, which stores it in its keystore. We support both Internet Explorer and Netscape/Mozilla enrollment methods.

After the guest presents his X.509 certificate by one of the above methods, our Web server generates a visual hash of it using the *Visprint* [Gol, Joh] program. (This program tranforms the MD5 hash of an object into an image using IFS fractals.)

After the guest uploads his certificate using one of the above methods, our Web server stores it in a temporary repository from which the delegator can retrieve it.

Delegator interface. A delegator first visits the same Web server as the guest, and searches for the guest's X.509 certificate by entering pieces of identifying information such as

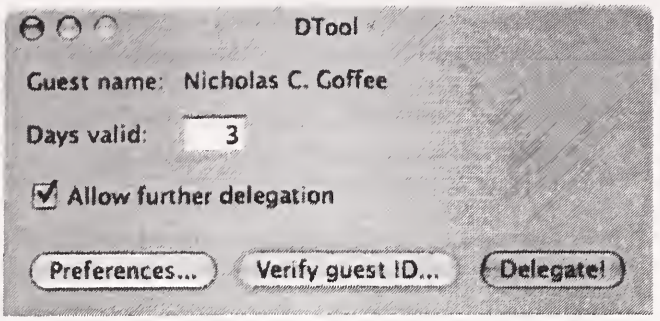


Figure 1: A screenshot of our delegator tool (a trusted Java applet, shown running under Mac OS 10.3). Before delegation, the delegator has to verify the identity of the guest's public key using a visual hash comparison. Also notice the inputs for validity interval and whether or not to allow further delegation.

the guest's name and organization. After this step, the delegator verifies the certificate's authenticity (by comparing visual hashes) and constructs and signs a SPKI certificate.

Signing a SPKI certificate is problematic, because it requires access to the delegator's private key. A private key must be well-protected so that adversaries cannot use it to sign data that did not actually originate from the owner. Software usually signs data of a very specific type (email, Word documents, authentication challenges, certificates) to prevent misuse of the key.

We therefore needed to build a special software tool for signing SPKI certificates. We considered a number of alternative ways to implement this, including a custom application which delegators would have to download, but for the prototype, we settled on using a trusted Java applet (screenshot shown in Figure 1). Trusted applets are hashed and signed by an entity that the user of the applet trusts, ensuring that the applet has not been modified to do anything the signing entity did not intend. Sun's Java plugin for Web browsers, by default, gives trusted applets greater privileges than standard applets, including the ability to access the local filesystem on the client machine. (It is not unreasonable to have local users install a trust root certificate for the applet signer ahead of time.) Our applet can, therefore, load the delegator's private key from a local file[3] and, after prompting for a password to decrypt the key, use it to sign a SPKI certificate.

Our Web server generates a page with a reference to the delegation applet, and provides the guest's PEM-encoded certificate as an argument to the applet. The applet uses standard Java cryptography functionality to extract the public key from this certificate, and uses a Java SPKI/SDSI library from MIT [Mor98] to construct and sign a SPKI certificate that delegates wireless access privileges to the guest. The applet allows the delegator to specify a validity interval for the new certificate and choose whether or not the recipient should be able to delegate further. We have ported the Visprint code from C to Java so we can build the visual hash verification step into the applet as well.

[3]The applet prompts the delegator to choose an appropriate keystore file the first time it is run, and saves its location to a local preferences file for future signing sessions. We currently support PKCS12 keystore files. In the future, we would like to support various platforms' OS keystores.

7 Making the Decision

We now consider the process by which our modified RADIUS decides whether to admit users.

7.1 The decision process

Local users. In the initial case, local users show authorization (via EAP-TLS) by proving knowledge of a private key matching an X.509 identity certificate issued by the local CA. Once the TLS handshake succeeds, the supplicant is granted access. On most platforms, the supplicant must choose which certificate to submit only on the first successful attempt; the machine will remember which certificate to use on subsequent attempts, making the authentication process transparent to local users.

Guests. Authorized guests also authenticate via EAP-TLS using an X.509 certificate. (In this case, "authentication" consists only of proving knowledge of the private key, since we cannot trust the certificate's naming information.) The RADIUS server uses a different process, however, to decide whether the user is authorized. It must find a valid SPKI/SDSI certificate chain originating from a principal it trusts that ultimately grants access privileges to the supplicant's public key.

In preliminary sketches, we also involved the delegator's X.509 certificate, but that does not seem to be necessary. As a consequence, the delegator doesn't necessarily need to have a centrally-issued X.509 identity certificate; we consider this further in Section 9.

The algorithm. Putting it all together, the modified RADIUS server follows the following procedure, illustrated by the flowchart in Figure 2:

- The supplicant initiaties an EAP-TLS authentication handshake.

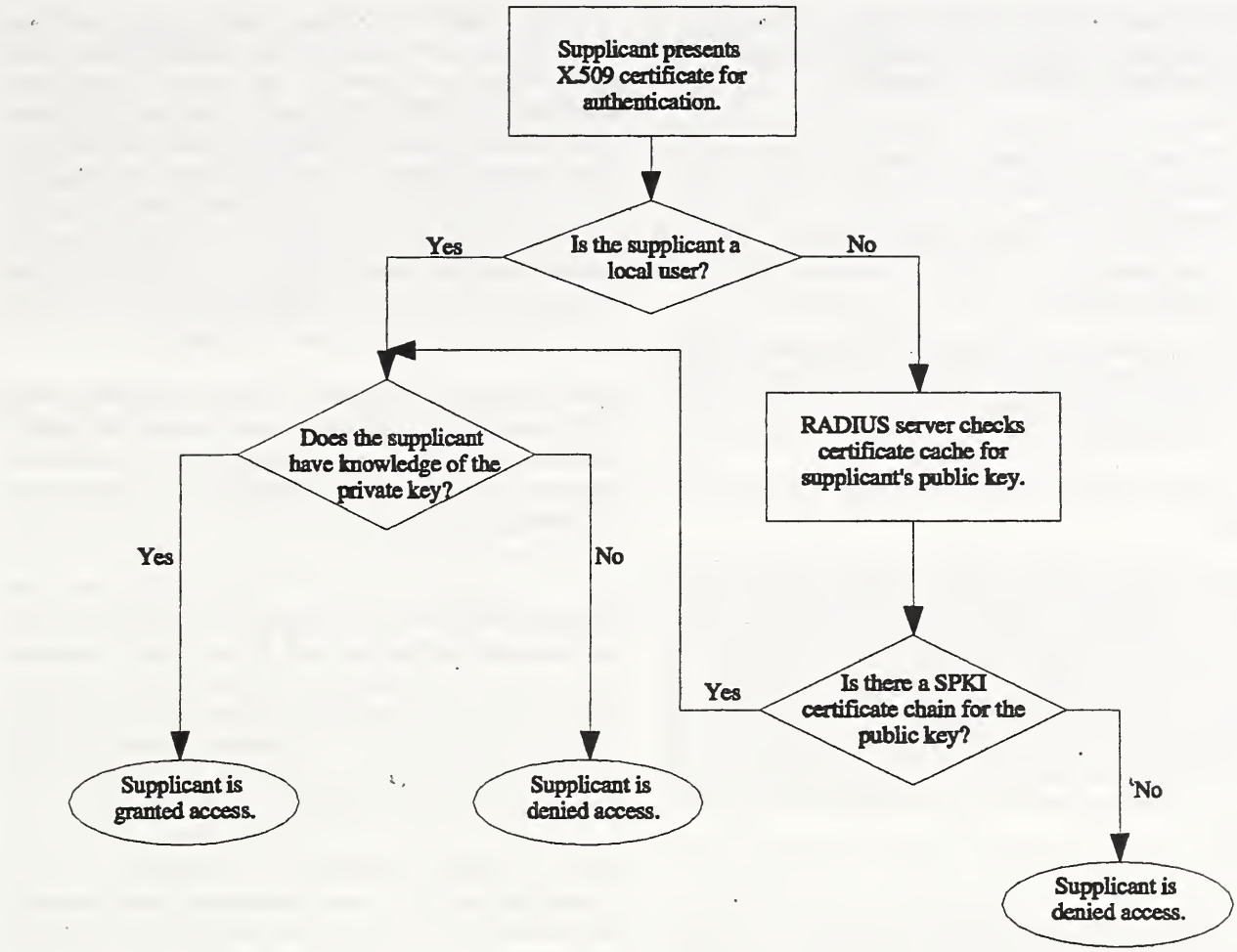


Figure 2: Decision flowchart used by the RADIUS server. If the supplicant is a local Dartmouth user (i.e., presents an X.509 certificate issued by the Dartmouth CA), then the supplicant only needs to prove knowledge of the private key associated with the certificate. Otherwise, if the supplicant is a guest, the RADIUS server checks for a SPKI certificate chain vouching for the supplicant's public key.

- If the supplicant cannot present an identity certificate, we shunt them to a special VLAN on which the supplicant can only connect to our delegation tool's "welcome" page.
- If the supplicant *can* present an identity certificate, we then evaluate it as follows:
  - If the certificate is valid and issued by the local CA, then we accept it.
  - Otherwise, if we can obtain and verify a valid SPKI/SDSI chain supporting it, we accept it.
  - Otherwise, we reject the certificate and shunt the supplicant to our "welcome" page.
- If we accept the certificate, and the supplicant proceeds to prove knowledge of the private key, then we let him in.
- Otherwise, we shunt the supplicant to our "welcome" page.

This procedures modifies standard EAP-TLS implementations only by changing how the server decides to accept a given supplicant certificate.

Getting the certificates ("pull" approach). To carry out the guest user case, the RADIUS server needs to know the X.509 identity certificate, the public key of whatever source-of-authority SPKI/SDSI chains will originate from, and the SPKI/SDSI certificate chain itself. EAP-TLS gives us the first, and we can build in the second. But how do we find the relevant SPKI/SDSI certificates?

One solution would be to have the delegation process leave the authorization certificates in a reliable, available directory where servers can access them; since the data is self-validating, maintenance of this directory should be automatic. When the RADIUS server needs to verify a guest's SPKI/SDSI credentials, it can "pull" up the credentials it re-

quires from the directory. We can organize these certificates as a forest: guest authorization certificates are children of the delegation certificates that signed them.

- The source-of-authority tool needs to write new delegator certificates to this directory.
- The delegator tool needs to read delegator certificates from this directory, and write new guest authorization certificates back.
- The RADIUS server needs to be able to ask for delegator-authorization chains whose leaves speak about a given public key.

The directory itself can perform time-driven checks for expiration.

Our initial implementation used the "pull" approach just described: SPKI/SDSI certificates were maintained in a cache that the RADIUS server can query via XML-RPC. The RADIUS server queries the cache about a particular public key; the cache itself finds a chain, if it exists, verifies it, and returns it. (To make our prototype more secure, we need to use authenticated XML-RPC messages or move the decision procedure onto the same machine as the RADIUS server.)

Getting the certificates ("push" approach). The centralized solution above is somewhat unsatisfying, because it introduces a centralized component (even if this component does not have significant security requirements). It would be slicker to find a way for the delegator and guest themselves to carry around the necessary certificates, since the necessary information paths will exist. When necessary, the guest can "push" the necessary credentials to the RADIUS server for validation.

We note that HTTP cookies provide most of the functionality we need. (We will add a message to the guest welcome page notifying users of what browser features will need to be enabled, including cookies and Java, in order to use our services.)

- The delegator will be interacting with the source-of-authority signing tool when their delegation certificate is created; the delegation certificate could be saved at the delegator machine as a cookie.
- At delegation time, both the delegator and the guest will be interacting with the delegation tool. The tool can read the delegator's certificate chain as a cookie, concatenate it with the new authorizatiion certificate, and then store the resulting chain as a cookie in the guest's Web browser.

The only remaining question would be how to get this cookie from the guest machine to the RADIUS server, when an authorized guest connects. One approach would be to add a short-term SPKI/SDSI store to the RADIUS server. When deciding whether to accept an X.509 certificate not issued by the Dartmouth CA, the server looks in this store for a SPKI/SDSI certificate chain for the public key in this X.509 cert. If none can be found, the supplicant is routed to a special Web page, that will pick up the guest's certificate chain from an HTTP cookie (this step requires that the guest have a browser running) and save it in the store.[4]

In this decentralized approach, it also might make sense to have the delegation tool save newly created SPKI/SDSI chains in the short-term store at the RADIUS server, since the guest will likely want to use the network immediately after being delegated to.

Changing VLANs. We now have two scenarios—when first receiving delegation, and in the above decentralized store approach—where a supplicant will be connected through the access point to the special VLAN, but will want to then get re-connected to the standard network. In both scenarios, the guest will be interacting with the Web server we have set up on the special VLAN.

One way to handle this would be for our server to display a page telling the guest how to cause their machine to drop the network and re-associate. However, this is not satisfying, from a usability perspective.

Instead, it would be nice to have our server (and backend system) cause this action automatically. One approach would be to use the administrative interface provided by the access point. For example, the Cisco 350 access point (that we're experimenting with) permits an administrator, by a password-authenticated Web connection, to dis-associate a specific supplicant (after which the supplicant re-initializes the network connection, and tries EAP-TLS again). We could write a daemon to perform this task, when it receives an authenticated request from our backend server. The server needs to know *which* access point the supplicant is associated with; however, in both scenarios, the RADIUS server has recently seen the supplicant MAC and access point IP address, since it told the access point to route this supplicant down the special VLAN. If nothing else, we can cache this information in a short-term store that the daemon can query.

We plan to explore other approaches here as well.

[4] As this paper goes to press, we have successfully implemented the cookie-based approach suggested here.

7.2 Executing the decision

On the server side, we are currently using FreeRadius version 0.9.2, running on a Dell P4 workstation running Red Hat 9, and an Apache Web server running on another Dell P4 workstation running Red Hat 9. We're testing with a Cisco 350 access point, with a Cisco Catalyst 2900 series XL switch and a hub to connect the two machines running the RADIUS server and Web server.

Setup. In our prototype, we have the access point configured to provide two different SSIDs. The broadcast SSID is called "Guest user" and authentication is not needed. It associates all users onto VLAN 2, the guest VLAN. The SSID "Dartmouth user" is not broadcast, and requires EAP authentication. Supplicants who pass EAP authentication are associated to this SSID on VLAN 1, the native VLAN that has access to the whole network. (We will abbreviate these designations as V_1 and V_2 in the following discussion.)

Our VLAN configuration is illustrated in Figure 3. The RADIUS server is connected to V_1 on the switch and the Web server is connected to V_2. The access point, connected to V_1, is configured to query the RADIUS server for user authentication. The hub connects the two machines and allows them to communicate to one another through the resulting private connection. In the future, we will obtain a router capable of VLAN trunking, which will allow the Web server to exist on both VLANs; this will eliminate the need for the private connection through a hub.[5]

Configuration. The EAP-TLS module of FreeRADIUS uses OpenSSL to execute the SSL/TLS handshake between the supplicant and the RADIUS server. After we changed the appropriate FreeRADIUS configuration files to enable EAP-TLS authentication and linked it with the OpenSSL libraries [Sul02], the RADIUS server was ready to accept EAP-TLS authentication attempts. The client file was configured to only accept requests sent from an access point (called a *network authentication server*, *NAS*, in the RADIUS protocol) with a Dartmouth IP address and the user file was set to only allow EAP (in our case EAP-TLS) authentication for all users, placing the user on V_1 if successfully authenticated. A shared secret between the RADIUS server and the NAS secures communication between these two components.

In order to use EAP-TLS authentication, the RADIUS server needs a trusted root CA so that it knows which certificates to accept. The RADIUS server also needs its own server certificate and key pair issued by the trusted root CA for authenticating itself to the supplicant in the handshake process. Local users are given a key pair and issued client certificates signed by the trusted root CA. OpenSSL can be used to generate key pairs, create a root certificate, and issue server and client certificates [Ros03]. Once the RADIUS server has a trusted root CA to refer to, it can handle authentication requests from the access point.

We modified the RADIUS server code to link with XML-RPC libraries we installed on the same machine. These libraries allow the RADIUS server to commmunicate with the cache, mentioned above, that stores SPKI/SDSI certificates and searches for chains authorizing a given principal to connect.

The decision process. The RADIUS server idles and waits for packets. When it receives an EAP Access-Request packet, it checks to see if the NAS that sent the packet is recognized and the shared secret is correct. If so, then it looks at the packet and sees what type of authentication is used. Since the SSID is configured to require EAP authentication, the RADIUS server should only receive EAP authentication requests from the NAS.

Once the EAP-TLS module is done executing, the decision to accept or reject the supplicant has already been made and is packed into the response packet. Thus it is necessary to intercept the EAP-TLS module before a reject decision is made to accomodate any modifications to the decision process.

Our modification determines if there is an error code returned by reading the supplicant's certificate. For example, the most common case would be the certificate is issued by an unrecognized CA. Once the validity checks are finished, we read the resulting error code to see if the validation passed or failed. If it passed, then the certificate presumably was issued by the known CA and the supplicant has provided knowledge of the corresponding private key. If the handshake failed due to an unrecognized CA, however, we use XML-RPC to query the Java SDSI library code about the public key of the X.509 certificate provided. The library uses the SPKI/SDSI certificate chain discovery algorithm proposed by Clark et al. [CEE+01]. If the Java code finds a valid SPKI certificate chain vouching for the supplicant, then we accept the supplicant and the EAP-TLS module returns an accept code. If such a SPKI/SDSI certificate chain cannot be found, then the user is rejected. Once graceful VLAN switching is implemented, the unauthorized guest will be placed on V_2 and see a Web browser window with instructions for obtaining guest access.

[5] We recently revised our setup to include a Cisco 2600 series router to handle VLAN trunking. The revised setup also uses newer models of the switch (Cisco 3550 series) and access point (Cisco 1100 series).

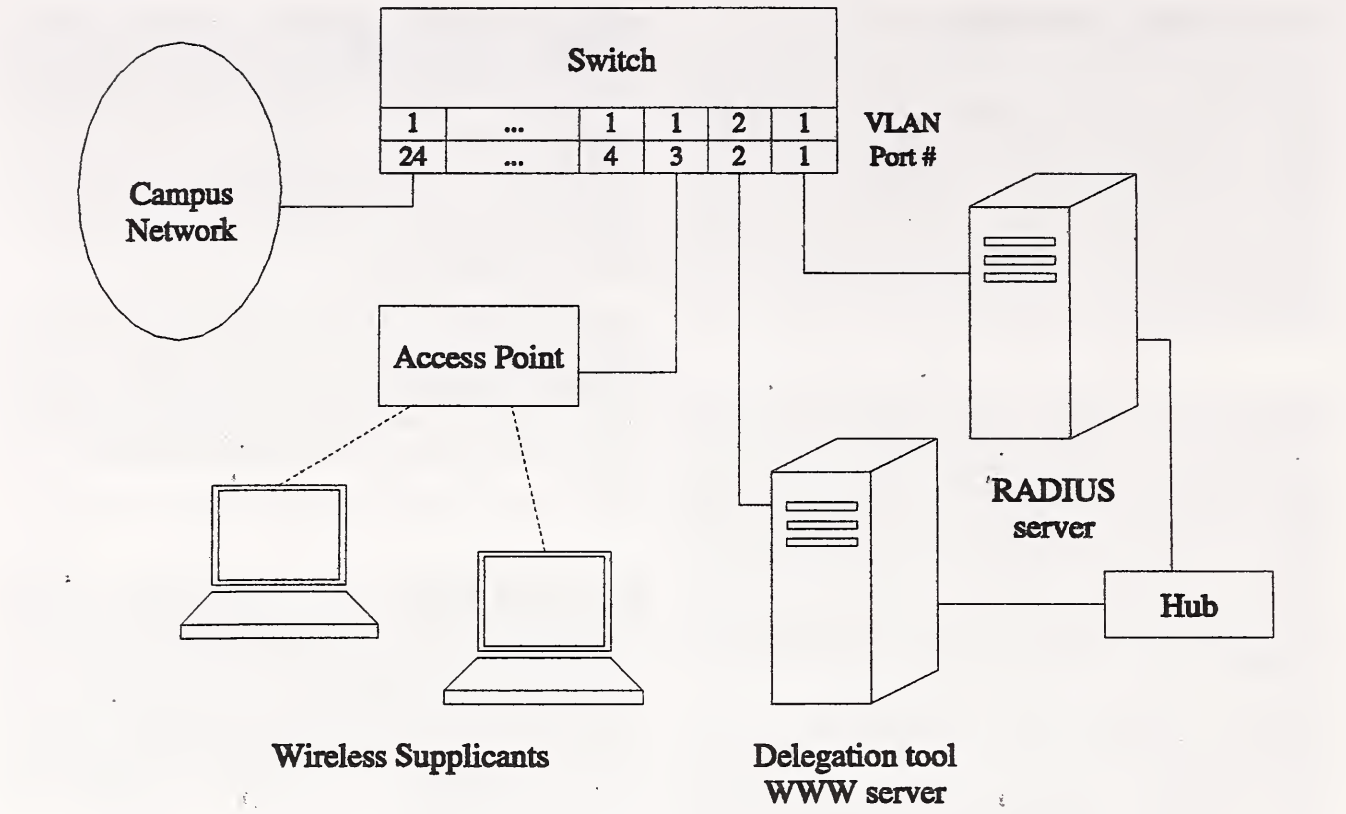


Figure 3: The setup of the Greenpass prototype. The switch is configured to associate VLANs with physical port numbers. The Web server is the only element currently housed on VLAN 2. Eventually, VLAN trunking will be used to communicate between the RADIUS server and the web server, eliminating the need for the private connection that exists between the two.

8 Related Work

Balfanz et al. [BDS+03] propose using secret keys to let wireless parties authenticate. We've already noted related work [DE02] in the "introduction" problem between two devices.

In the SPKI/SDSI space, the Geronimo project at MIT [Cla01, May00] uses SPKI/SDSI to control objects on an Apache Web server. The project uses a custom authorization protocol, with an Apache module handling the server side of the protocol and a Netscape plug-in handling the client side. The protocol can be tunneled inside an SSL channel for further protection; the authors also considered replacing X.509 with SPKI/SDSI within SSL. Howell and Kotz [HK00] describe a similar SPKI/SDSI-based access-control system for Web content as part of their *Snowflake* authorization architecture. Koponen et al. [KNRP00] propose having an Internet cafe operator interact with a customer via infrared, and then having that customer authenticate to the local access point via a SPKI/SDSI certificate; however, this work does not use standard tools and institution-scale authentication servers. Eronen and Nikander [EN01] describe several SPKI/SDSI-based enhancements to both authorization and authentication in Jini, a Java-based distributed computing environment.

Canovas and Gomez [CG02] describe a distributed management system for SPKI/SDSI name certificates and authorization certificates. The system contains name authorities (NAs) and authorization authorities (AAs) from which entities can request name and authorization certificates, including certificates which permit the entity to make requests of further NAs and RAs. The system takes advantage of both name certificates that define groups (i.e., roles) and authorization certificates that grant permissions to either groups or individual entities.

9 Future Directions

Initially, we plan to "take the duct tape" off of our current prototype, and try it in a more extensive pilot. Beyond this initial work, we also hope to expand in several directions.

No PKI. We note that our approach could also accommodate the scenario where *all* users are "guests" with no keypairs—in theory, obviating the need for an X.509 identity PKI for the local population. For example, if an institution already has a way of authenticating users, then they could use a modified delegator tool that:

- authenticates the delegator (via the legacy method)
- sees that the delegator has a self-signed certificate (like our guest tool does)
- then signs a SPKI/SDSI delegator certificate for this public key (like our delegator tool does).

In some sense, the division between the X.509 PKI and the delegated users is arbitrary. It would be interesting to explore the implications of dividing the population in other ways than users versus guests (perhaps "permanent Dartmouth staff" versus "students likely to lose their expensive smart-card dongles while skiing").

Not just the network. Many types of digital services use X.509 identity certificates as the basis for authentication and authorization. For example, at Dartmouth, we're migrating many current Web-based information services to use X.509 and client-side SSL/TLS. In the Greenpass pilot, we're adding flexibility to wireless access by extending X.509/TLS with SPKI/SDSI. This same PKI approach can work for networked applications that expect X.509, such as our Web-based services.

In the second phase, we will extend the Greenpass infrastructure to construct a single tool that allows delegation of authorization to networked applications as well as to the network itself.

Not just EAP-TLS. Some colleagues insist that *virtual private networks* (*VPNs*) with client-side keypairs are the proper way to secure wireless networks. In theory, our scheme should work just as well there. In the second phase, we plan to try this.[6]

Alternative approaches to hash verification. An attacker could potentially abuse our delegator applet if the delegator chooses to skip the fingerprint-verification step. Visual fingerprints are designed to discourage users from skipping crucial verification steps: it is faster and less painful to compare two visual fingerprints than to compare hashes represented as hexadecimal strings. We must devise either a method that ensures the delegator cannot skip this step,[7] or a method that takes humans out of the loop entirely. Balfanz et al. [BSSW02] suggest an introduction phase based on a *location-limited channel*; this approach might allow us to eliminate human interaction from the introduction phase in the future. We are also considering alternative models of fingerprint verification: for example, using PGPfone's [PGP] mapping of hash values to word lists would allow introduction to take place over the phone as well as in person.

Other threats. We designed our prototype around the concept that a user's public key is his or her online identity; as a result, delegators authenticate to Web servers and the RADIUS server using the same keypair (identity) as they use to sign SPKI/SDSDI certificates for guests. A weakly designed authentication protocol—one that requires a user to sign an unstructured random challenge using his private key—could be exploited by a malicious server. Specifically, the server might present a "random" challenge that actually contains a SPKI/SDSI certificate or its hash: a delegator could then be tricked into signing that certificate by authenticating to the malicious server, whose owner might use the resulting signature to obtain unauthorized access to the wireless LAN or some other resource. We therefore need to consider whether the TLS and SSL client authentication handshakes are vulnerable to such an attack.

The TLS 1.0 [AD99] and SSL version 3.0 [FKK96][8] handshakes appear to be immune to this attack due to the format of the MAC that the client signs in order to authenticate (at least when using an RSA keypair: see below). In both protocols, the MAC that is signed is a *concatenation* of both the MD5 and SHA-1 hashes of the values in question. The SPKI/SDSI certificate format [EFL+99a] defines signatures using *either* an MD5 or a SHA-1 hash.

When authentication using a DSA keypair or using SSL version 2.0 [Hic95], on the other hand, the client signs *only* a SHA-1 or an MD5 hash, respectively. In both these latter cases, however, the signed MAC is function of previous handshake values, including random values generated by the client and structured values such as complete handshake messages (in TLS or SSLv3) or the server's certificate (in SSLv2; note that the client will already have verified this certificate before sending its own authentication materials). As a result, it is not possible for the server to get the client to sign a value that is a valid SPKI/SDSI certificate structure. A malicious server would need to engineer its own handshake values in such a way that the entire sequence signed by the client has the same MAC value as the server's desired

[6] As this paper goes to press, we have successfully completed an initial test of VPN guest access using our existing client tools and RADIUS server; see Goffee [Gof04] for further details.

[7] An in-progress revision of our delegator tool requires the user to select the correct visual hash from among several choices.

[8] Also see Rescorla [Res01] for further discussion of both these protocols.

SPKI/SDSI certificate. This would require finding a collision in the hash function used; if a particular hash function is proven to be vulnerable to such attacks, we could begin accepting only those authorization certificates signed using stronger hash function.

Location-aware authorization and services. By definition, the RADIUS server making the access-control decision knows the supplicant's current access point. In some scenarios, we may want users to access the network only from certain access points; in some scenarios, users should be able to access some applications only from certain access points; potentially, the nature of the application content itself may change depending on access location.

In the second phase, we plan to extend the Greenpass infrastructure to enable authorization certs to specify the set of allowable access points. We will also enable the RADIUS back-end to sign short-term certificates testifying to the location of the supplicant (which requires an authorization cert for the server public key), and to enable applications to use these certificates for their own location-aware behavior. For example, we might put different classes of users (professors, students, guests, etc.) on different VLANs according to the resources we would like them to access. It might also be interesting to allow certain users to access the WLAN only from certain locations—e.g., conference rooms and lecture halls.

Who is being authorized? Campus environments are not monolithic. At Dartmouth, we already have multiple schools, departments, and categories of users within departments. Managing authorization of such internal users is a vexing problem. Centralized approaches are awkward and inflexible: a colleague at one university ended up developing over 100 different user profiles; a colleague at another noted she has to share her password to team-teach a security course, because the IT department has no other way to let her share access to the course materials.

In the second phase, we plan to extend the Greenpass infrastructure to support authorization delegation for "local users" as well as guests, and to permit local users to easily manage authorization for information resources they own or create.

Devices. Currently, laptops are probably the most common platform for access to wireless networks. Other platforms are emerging, however. At Dartmouth, students and staff already carry around an RFID tag embedded in their ID cards, a research team is developing experimental wireless PDAs for student use, and we are beta-testing Cisco's new VoIP handset device; we're also testing Vocera's device for Wi-Fi voice communication.

In the second phase, we plan to explore using these alternate devices in conjunction with Greenpass. For example, a department's administrative assistant might be able to create a SPKI/SDSI cert and enter it in a directory simply by pointing a "delegation stick" (RFID tag reader) at the student (detecting the student's ID card). In another example, when a physician at the Dartmouth-Hitchcock Medical Center collars a passing colleague for advice on a difficult case, he might be able to delegate permission to read that file simply by pointing his PDA at the colleague's PDA.

Distributed authorization. The PKI community has long debated the reason for PKI's failure to achieve its full potential. The technology exists and has clear benefits; adoption and deployment has been a challenge.

One compelling hypothesis is that the centralized, organizational-specific hierarchy inherent in traditional approaches to PKI, compounded by a dependence on usable, globally unique names and awkward certificate structure, does not match the way that authorization really flows in human activities. By permitting authorization to start at the end-users (rather than requiring that it start at centralized places), and by using a system (SPKI/SDSI) designed to address the namespace and structure issues, Greenpass may overcome these obstacles.

In the second phase, we plan to extend Greenpass to reproduce real-world policies more complex than just "Prof. Kotz said it was OK," to examine (with our colleagues in the Dept of Sociology) how readily this authorization system matches the norms of human activity, and to examine whether humans are able to manage the user interfaces our Greenpass tools provide.

We also plan to take a closer look at how other authorization schemes might fit in this setting, in comparison to SPKI/SDSI. Some candidates that might work in this setting include the X.509-based PERMIS attribute certificate system [COB03, PER] and X.509 proxy certificates [TWE+03, WFK+04], as well as KeyNote [BFIK99, Key]. Theses by Nazareth [Naz03] and Goffee [Gof04] give overviews of many such systems.

10 Conclusion

In this paper we described a method of securing a wireless network while providing meaningful guest access. We added a step to EAP-TLS authentication that performs an additional authorization check based on SPKI/SDSI certifi-

cates. By using SPKI/SDSI, we eliminate the need for a cumbersome central authority; by grafting it on top of the existing X.509-based PKI, we do not require our users to install any additional client software.

The two major components of the Greenpass project are the delegation tools and the modified RADIUS server. The delegation tools automate the process of creating temporary SPKI/SDSI certificates for a guest, allowing an authorized (but not necessarily computer-savvy) delegator to grant an invited guest permission to use the network. The modified RADIUS server takes into account that guests will want to access the network and checks for guest credentials before making a decision to accept or reject a supplicant's request for network access.

The goal of our project is to create a solution that implements delegation in a way that reflects real-world authorization flow that does not rely too heavily on a centralized authority; SPKI/SDSI allows us to accomplish this goal. Our future work will allow us to investigate how our solution fits with other existing ideas, hopefully resulting in a solution that is secure, completely decentralized, and capable of adapting to new technology and delegation policies.

Acknowledgments

We gratefully acknowledge Kwang-Hyun Baek, Bob Brentrup, Bill Cote, Peter Glenshaw, Dave Kotz, Brad Noblet, and our colleagues at Cisco for the advice, and Eileen Ye for her initial SPKI/SDSI explorations. We plan to make code for this project available under an open-source license.

References

[AD99] Christopher Allen and Tim Dierks. The TLS Protocol Version 1.0. IETF RFC 2246, January 1999.

[AS99] Bernard Aboba and Dan Simon. PPP EAP TLS Authentication Protocol. IETF RFC 2716, October 1999.

[BDS+03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret Handshakes from Pairing-Based Key Agreements. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 180–196, May 2003.

[BFIK99] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote Trust-Management System Version 2. IETF RFC 2704, September 1999.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1996.

[BGW01] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, 2001.

[BSSW02] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2002.

[BV98] Larry J. Blunk and John R. Vollbrecht. PPP Extensible Authentication Protocol (EAP). IETF RFC 2284, March 1998.

[CEE+01] D. Clark, J. Elien, C. Ellison, M. Fredette, A. Morcos, and R. Rivest. Certificate Chain Discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[CG02] Oscar Canovas and Antonio F. Gomez. A distributed credential management system for spki-based delegation scenarios. In *Proceedings of the 1st Annual PKI Research Workshop*, April 2002.

[Cla01] Dwaine E. Clarke. SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI. Master's thesis, Massachusetts Institute of Technology, September 2001.

[COB03] David W. Chadwick, Alexander Otenko, and Edward Ball. Role-Based Access Control with X.509 Attribute Certificates. *IEEE Internet Computing*, March-April 2003.

[CS01] IEEE Computer Society. IEEE Standard for Local and metropolitan area networks: Port-Based Network access control. IEEE Standard 802.1X-2001, October 2001.

[CWHWW03] Nancy Cam-Winget, Russ Housley, David Wagner, and Jesse Walker. Security Flaws in 802.11 Data Link Protocols. *Communications of the ACM*, 46(5):35–39, May 2003.

[DE02] Steve Dohrmann and Carl M. Ellison. Public-Key Support for Collaborative Groups. In *Proceedings of the 1st Annual PKI Research Workshop*, April 2002.

[EA03] Jon Edney and William A. Arbaugh. *Real 802.11 Security*. Addison-Wesley, 2003.

[EFL+99a] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. Simple public key certificate. IETF Internet-Draft, `http://theworld.com/~cme/examples.txt`, July 1999.

[EFL+99b] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylonen. SPKI Certificate Theory. IETF RFC 2693, September 1999.

[EN01] Pasi Eronen and Pekka Nikander. Decentralized Jini Security. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pages 161–172, February 2001.

[FH02] Stephen Farrell and Russell Housley. An Internet Attribute Certificate Profile for Authorization. IETF RFC 3281, April 2002.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL Protocol Version 3.0. IETF Internet-Draft, `http://wp.netscape.com/eng/ssl3/draft302.txt`, November 1996.

[Gof04] Nicholas C. Goffee. Greenpass Client Tools for Delegated Authorization in Wireless Networks. Master's thesis, Dartmouth College, June 2004. Technical Report TR2004-509.

[Gol] Ian Goldberg. Visual Fingerprints (Visprint software homepage). `http://www.cs.berkeley.edu/~iang/visprint.html`.

[Hic95] Kipp E. B. Hickman. SSL 2.0 Protocol Specification. Netscape draft specification, `http://wp.netscape.com/eng/security/SSL_2.html`, February 1995.

[HK00] Jon Howell and David Kotz. End-to-end authorization. In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI)*, pages 151–164, 2000.

[Joh] David Johnston. Visprint, the Visual Fingerprint Generator. `http://www.pinkandaint.com/oldhome/comp/visprint/`.

[Key] Keynote home page. `http://www.cis.upenn.edu/~keynote/`.

[Kim04] Sung Hoon Kim. Greenpass RADIUS Tools for Delegated Authorization in Wireless Networks. Master's thesis, Dartmouth College, June 2004. Technical Report TR2004-510.

[KNRP00] Juha Koponen, Pekka Nikander, Juhana Rasanen, and Juha Paajarvi. Internet Access through WLAN with XML-encoded SPKI Certificates. In *Proceedings of NORDSEC 2000*, 2000.

[May00] Andrew J. Maywah. An Iplementation of a Secure Web Client Using SPKI/SDSI Certificates. Master's thesis, Massachusetts Institute of Technology, May 2000.

[Mor98] Alexander Morcos. A Java Implementation of Simple Distributed Security Architecture. Master's thesis, Massachusetts Institute of Technology, May 1998.

[Naz03] Sidharth Nazareth. SPADE: SPKI/SDSI for Attribute Release Policies in a Distributed Environment. Master's thesis, Department of Computer Science, Dartmouth College, May 2003. `http://www.cs.dartmouth.edu/~pkilab/theses/sidharth.pdf`.

[PER] PERMIS home page. `http://www.permis.org/`.

[PGP] PGPfone: Pretty Good Privacy Phone Owner's Manual. `http://web.mit.edu/network/pgpfone/manual/`.

[Res01] Eric Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2001.

[Rig00] Carl Rigney. RADIUS Accounting. IETF RFC 2866, June 2000.

[Ros03] Ken Roser. HOWTO: EAP-TLS Setup for FreeRADIUS and Windows XP Supplicant. `http://3w.denobula.com:50000/EAPTLS.pdf`, February 2003.

[RWC00] Carl Rigney, Ward Willats, and Pat R. Calhoun. RADIUS Extensions. IETF RFC 2869, June 2000.

[RWRS00] Carl Rigney, Steve Willens, Allan C. Rubens, and William Allen Simpson. Remote Authentication Dial In User Service (RADIUS). IETF RFC 2865, June 2000.

[Sul02] Adam Sulmicki. HOWTO on EAP/TLS authentication between FreeRADIUS and XSupplicant. `http://www.missl.cs.umd.edu/wireless/eaptls/`, April 2002.

[TER] Trans-European Research and Education Networking Association (TERENA). `http://www.terena.nl/tech/task-forces/tf-mobility/`.

[TWE+03] Steven Tuecke, Von Welch, Doug Engert, Laura Pearlman, and Mary Thompson. Internet X.509 Public Key Infrastructure Proxy Certificate Profile. IETF Internet-Draft, `http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-10.txt`, December 2003.

[WFK+04] Von Welch, Ian Foster, Carl Kesselman, Olle Mulmo, Laura Pearlman, Steven Tuecke, Jarek Gawor, Sam Meder, and Frank Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. In *3rd Annual PKI Research and Development Workshop*, 2004.

X.509 Proxy Certificates for Dynamic Delegation

Von Welch[1], Ian Foster[2,3], Carl Kesselman[4], Olle Mulmo[5], Laura Pearlman[4], Steven Tuecke[2], Jarek Gawor[2], Sam Meder[3], Frank Siebenlist[2]

[1]National Center for Supercomputing Applications, University of Illinois
[2]Mathematics and Computer Science Division, Argonne National Laboratory
[3]Department of Computer Science, University of Chicago
[4]Information Sciences Institute, University of Southern California
[5]Royal Institute of Technology, Sweden

Contact author: Von Welch (vwelch@ncsa.uiuc.edu)

Abstract

Proxy credentials are commonly used in security systems when one entity wishes to grant to another entity some set of its privileges. We have defined and standardized X.509 Proxy Certificates for the purpose of providing restricted proxying and delegation within a PKI-based authentication system. We present here our motivations for this work coming from our efforts in Grid security, the Proxy Certificate itself, and our experiences in implementation and deployment.

1 Introduction

"Grids" [10] have emerged as a common approach to constructing dynamic, inter-domain, distributed computing and data collaborations. In order to support these environments, Grids require a light-weight method for dynamic delegation between entities across organizational boundaries. Examples of these delegation requirements include granting privileges to unattended processes which must run without user intervention, the short-term sharing of files for collaboration, and the use of brokering services which acquire resources (e.g., storage, computing cycles, bandwidth) on behalf of the user.

The Globus Toolkit® [11] has emerged as the dominant middleware for Grid deployments worldwide. The Grid Security Infrastructure (GSI) [39,2,9] is the portion of the Globus Toolkit that provides the fundamental security services needed to support Grids. GSI provides libraries and tools for authentication and message protection that use standard X.509 public key certificates [5,16], public key infrastructure (PKI), the SSL/TLS protocol [6], and X.509 Proxy Certificates, an extension defined for GSI to meet the delegation requirements of Grid communities.

Proxy Certificates allow an entity holding a standard X.509 public key certificate to delegate some or all of its privileges to another entity which may not hold X.509 credentials at the time of delegation. This delegation can be performed dynamically, without the assistance of a third party, and can be limited to arbitrary subsets of the delegating entity's privileges. Once acquired, a Proxy Certificate is used by its bearer to authenticate and establish secured connections with other parties in the same manner as a normal X.509 end-entity certificate.

Proxy Certificates were first prototyped in early implementations of GSI. Subsequently, they have been refined through standardization in the IETF PKIX working group [17]

and have achieved RFC status. (At the time of this writing, the Proxy Certificate internet draft [37] has passed IETF-wide public comment and is only awaiting assignment of an RFC number). GSI currently implements this standard.

GSI and Proxy Certificates have been used to build numerous middleware libraries and applications that have been widely deployed in large production and experimental Grids [2,3,4,19,35]. This experience has proven the viability of proxy delegation as a basis for authorization within Grids, and has further proven the viability of using X.509 Proxy Certificates.

We start with a discussion of the requirements that spurred our use of X.509 public key certificates and motivated our development of Proxy Certificates. We follow with a technical description of the format of Proxy Certificates in Section 3. Section 4 describes how Proxy Certificates can be used to achieve single sign-on and delegation, and Section 5 describes how Proxy Certificates can be integrated with different types of authorization systems. Section 6 discusses current implementations and applications of Proxy Certificates. Section 7 discusses performance issues with Proxy Certificates and security tradeoffs in regards to those issues. We conclude with a discussion of related work in Section 8 and a summary in Section 9.

2 Motivation

We discuss first our motivation for the use of X.509 certificates and PKI as the basis for our GSI implementation. Then we discuss the motivations that lead to the creation of Proxy Certificates as an enhancement to standard X.509 public key certificates.

2.1 Motivation for X.509 Certificates

GSI uses X.509 public key certificates and Secure Socket Layer (SSL) for authentication not only because these are well-known technologies with readily available, well-tested open source implementations, but because the trust model of X.509 certificates allows an entity to trust another organization's certification authority (CA) without requiring that the rest of its organization do so or requiring reciprocation by the trusted CA.

This flexibility of trust model for X.509 certificates was a deciding factor between X.509 certificates and other common authentication mechanisms. For example, Kerberos [29] requires that all cross-domain trust be established at the domain level, meaning that organizations have to agree to allow cross-domain authentication, which can often be a heavy-weight administrative process. In many common Grid deployments, only a few users and resources at a particular organization may participate in the Grid deployment, making the process of acquiring buy-in from the organization as a whole to establish the authentication fabric prohibitive.

2.2 Motivation for Proxy Certificates

The establishment of X.509 public key certificates and their issuing certification authorities provides a sufficient authentication infrastructure for persistent entities in Grids. However, several use cases exist that are not well covered by X.509 public key certificates alone.

- *Dynamic delegation*: It is often the case that a Grid user needs to delegate some subset of their privileges to another entity on relatively short notice and only for a brief amount of time. For example, a user needing to move a dataset in order to use it in a computation may want to grant to a reliable file transfer service the necessary rights to access the dataset and storage so that it may perform a set of file transfers on the user's behalf. Since these actions may be difficult to predict, having to arrange delegation ahead of time through some administrator is prohibitive.
- *Dynamic entities*: In addition to delegation to persistent services and entities, the requirement exists to support delegation of privileges to services that are created dynamically, often by the user them self, that do not hold any form of identity credential. A common scenario is that a user submits a job to a computational resource and wants to delegate privileges to the job to allow it to access other resources on the user's behalf, for example, to access data belonging to the user on other resources or start sub-jobs on other resources. An important point here is that the user wants to delegate privileges specifically to the job and not to the resource as a whole (i.e., other jobs being run by other users on the resource should not share the rights).
- *Repeated Authentication*: It is common practice to protect the private keys associated with X.509 public key certificates either by encrypting them with a pass phrase (if stored on disk) or by requiring a PIN for access (if on a smart card). This technique poses a burden on users who need to authenticate repeatedly in a short period of time, which occurs frequently in Grid scenarios when a user is coordinating a number of resources.

A number of existing mechanisms could satisfy the first use case. For example, user-issued X.509 attribute certificates [8] could be used to delegate rights to other bearers of X.509 public key certificates. However, the heavy-weight process of vetting associated with the issuing of public key certificates makes it prohibitive to use this method for the dynamically created entities described in the second use case: acquiring public key certificates for dynamically created, and often short-lived entities, would be too slow for practical use. It would have been possible to use other means for authenticating these dynamic entities, for example bare keys as described in Section 8.5, but this approach would have required protocol modifications (or a new protocol) to accommodate the new authentication mechanism.

The third scenario could be solved by caching the pass phrase or PIN required for access to the private key. However, this caching increases the risk of compromising the private key if the memory storing the pass phrase or PIN is somehow accessible to an attacker or is written out to disk (e.g., if it is swapped or in a core dump). In addition, reliably caching the PIN for a set of simultaneously running applications is a non-trivial software engineering exercise.

These requirements led us to develop an authentication solution that allows users to create identities for new entities dynamically in a light-weight manner, to delegate privileges to those entities (again in a dynamic, light-weight manner), to perform single sign-on, and that allows for the reuse of existing protocols and software with minimal

modifications. The result is the X.509 Proxy Certificate, which we describe in the following sections.

We note that while it may be possible to use Proxy Certificates for uses other than authentication, delegation and message protection, for example the signing or encryption of long-lived documents, these alternate uses were not motivating factors in the Proxy Certificate design and we have not investigated such use .

3 Description of Proxy Certificates

We now describe the contents of a Proxy Certificate and briefly discuss methods of revocation and path validation.

3.1 Proxy Certificate Contents

Proxy Certificates use the format prescribed for X.509 public key certificates [5,16] with the prescriptions described in this section on the contents. Proxy Certificates serve to bind a unique public key to a subject name, as a public key certificate does. The use of the same format as X.509 public key certificates allows Proxy Certificates to be used in protocols and libraries in many places as if they were normal X.509 public key certificates which significantly eases implementation.

However, unlike a public key certificate, the issuer (and signer) of a Proxy Certificate is identified by a public key certificate or another Proxy Certificate rather than a certification authority (CA) certificate. This approach allows Proxy Certificates to be created dynamically without requiring the normally heavy-weight vetting process associated with obtaining public key certificates from a CA.

The subject name of a Proxy Certificate is scoped by the subject name of its issuer to achieve uniqueness. This is accomplished by appending a CommonName relative distinguished name component (RDN) to the issuer's subject name. The value of this added CommonName RDN should be at least statistically unique to the scope of the issuer. The value of the serial number in the Proxy Certificate should also be statistically unique to the issuer. Uniqueness for both of these values in our implementations is achieved by using the hash of the public key as the value. Unique subject names and serial numbers allow Proxy Certificates to be used in conjunction with attribute assertion approaches such as attribute certificates [8] and have their own rights independent of their issuer.

The public key in a Proxy Certificate is distinct from the public key of its issuer and may have different properties (e.g., its size may be different). As we describe in more detail in Section 4, except when using Proxy Certificates for single sign-on, the issuer does not generate the public key-pair and has no access to the private key.

All Proxy Certificates must bear a newly-defined critical X.509 extension, the Proxy Certificate Information (PCI) extension. In addition to identifying Proxy Certificates as such, the PCI extension serves to allow the issuer to express their desire to delegate rights to the Proxy Certificate bearer and to limit further Proxy Certificates that can be issued by that Proxy Certificate holder.

The issuer's desires towards delegation to the Proxy Certificates bearer are expressed in the PCI extension using a framework for carrying policy statements that allow for this delegation to be limited (perhaps completely disallowed). There exist today a number of policy languages for expressing delegation policies (e.g., Keynote, XACML, XrML), instead of defining a new mechanism or selecting a single existing policy language for expressing delegation policy (which probably would have bogged the process of standardizing Proxy Certificates down considerably), Proxy Certificates instead allow the issuer to use any delegation policy expression it chooses. The only restriction being that the issuer needs to know (through some out-of-band method) that the relying party understands its method of expression. This allows different deployments to select (or create) a method of delegation policy expression best suited for their purposes.

This use of arbitrary policy expressions is achieved through two fields in the PCI extension: a policy method identifier and a policy field. The policy method identifier is an object identifier (OID) that identifies the delegation policy method used in the policy field. The policy field then contains an expression of the delegation policy that has a format specific to the particular method (and may be empty for methods that do not require additional policy). For example, the identifier could contain an OID identifying the method as XACML and then the policy would contain an XACML policy statement.

The Proxy Certificate RFC defines two policy methods that must be understood by all implementations of Proxy Certificates (in addition to any more sophisticated methods they may implement):

- *Proxying*: This policy type indicates that the issuer of the Proxy Certificate intended to delegate all of their privileges to the Proxy Certificate bearer.
- *Independent*: This policy type indicates that the issuer of the Proxy Certificate intended the Proxy Certificate by itself to convey none of the issuer's privileges to the bearer. In this case the Proxy Certificate only serves to provide the bearer with a unique identifier, which may be used in conjunction with other approaches, such as attribute certificates, to grant its bearer privileges.

For both of these methods, the policy field is empty since the intended delegation policy is explicit in the type.

Certificate attribute	X.509 Public key certificate	X.509 Proxy Certificates
Issuer/Signer	A certification authority	A public key certificate or another Proxy Certificate
Name	Any as allowed by issuer's policy	Scoped to namespace defined by issuer's name
Delegation from Issuer	None	Allows for arbitrary policies expressing issuer's intent to delegate rights to Proxy Certificate bearer.
Key pairs	Uses unique key pair	Uses unique key pair

Table 1: Comparison of X.509 public key certificates and X.509 Proxy Certificates.

The PCI extension also contains a field expressing the maximum path lengths of Proxy Certificates that can be issued by the Proxy Certificate in question. A value of zero for this field prevents the Proxy Certificate from issuing another Proxy Certificate. If this field is not present, then the length of the path of Proxy Certificates, which can be issued by the Proxy Certificate, is unlimited.

3.2 Proxy Certificate Path Validation

Validation of a certificate chain has two distinct phases. First validation of the certificate chain up to the public key certificate occurs, as described by RFC 3280 [16]. Validation of the Proxy Certificate portion of the chain is then performed as described in the Proxy Certificate RFC [37]. In summary these rules are:

- Ensuring each Proxy Certificate has a valid Proxy Certificate Information extension as described in the previous section;
- Each Proxy Certificate must have a subject name derived from the subject name of its issuer;
- Verifying the number of Proxy Certificates in the chain does not exceed the maximum length specified in any of the Proxy Certificate Information extensions in the chain; and
- Storing the delegation policies of each Proxy Certificate so that the relying party can determine the set of rights delegated to the bearer of the end Proxy Certificate used to authenticate.

3.3 Revocation of Proxy Certificates

There currently exists no implemented method for revocation of Proxy Certificates. The intent is that Proxy Certificates are created with short life spans, typically on the order of hours (with eight hours being the default of our implementation). Therefore, revocation has not been a pressing issue since this short lifetime limits the length of misuse if a Proxy Certificate were to be compromised. However, Proxy Certificates can be uniquely identified in the same manner as normal end-entity certificates, through the issuer and serial number, so the potential exists to revoke them using the same mechanisms (e.g., CRLs [16] or OCSP [28]).

4 Use for Single Sign-on and Delegation

In this section we describe how Proxy Certificates can be used to perform single sign-on and delegation.

4.1 Enabling Single Sign-on

Normally the private key associated with a set of long-term X.509 credentials is protected in some manner that requires manual authentication on the behalf of its owner. While this process serves to provide a high level of protection of the private key, it can be prohibitively burdensome if the user needs to access the key frequently for authentication to other parties.

Proxy Certificates solve this problem by enabling single sign-on: that is, allowing the user to manually authenticate once in order to create a Proxy Certificate which can be used repeatedly to authenticate for some period of time without compromising the protection on the user's long-term private key. This is accomplished by creating a new key pair (composed of a public and private key), and by subsequently using the user's long-term private key to create a short-lived Proxy Certificate. The Proxy Certificate binds the new public key to a new name and delegates some or all of the user's privileges to the new name. The Proxy Certificate and the new private key are then used by the bearer to authenticate to other parties. Since the Proxy Certificate has a short lifetime, it is typically permissible to protect it in a less secure manner than the long-term private key. In practice this means the Proxy Certificate private key is stored on a local file system and is protected by only local file system permissions, which allows the user's applications to access it without any manual intervention by the user.

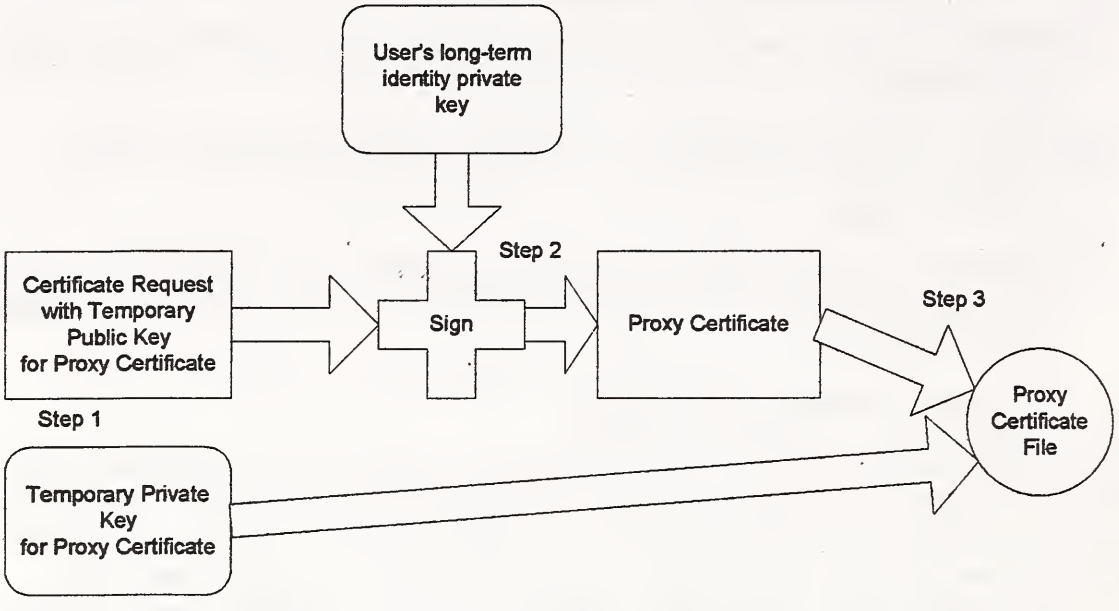


Figure 1: Creation of a Proxy Certificate for single sign-on. Steps are described in the text.

The process of creating a Proxy Certificate for single sign-on is shown in Figure 1. The steps, which are normally all done by a single application run by the user, are:

1. A new key pair, consisting of a public and private key, is generated for use in the Proxy Certificate. The public key is encoded in a certificate request [20] for further processing.
2. The user's private key associated with their long-term public key certificate is accessed (possibly requiring the manual entering of a pass phrase or PIN by the user) to sign the certificate request containing the public key of the newly generated key pair hence generating a Proxy Certificate. After signing the Proxy Certificate, the user's long–term private key can remain secured (or the associated smart card can be removed) until the Proxy Certificate expires.
3. The Proxy Certificate and its associated private key are then placed in a file. This file is protected only by local file system permissions to allow for easy access by the user.

When the Proxy Certificate expires, this process is repeated by the user to generate a new key pair and Proxy Certificate. The result from the perspective of the user is that manual authentication is required only infrequently to enable applications to authenticate on their behalf.

4.2 Delegation over a Network

Proxy Certificates can also be created so as to delegate privileges from an issuer to another party over a network connection without the exchange of private keys. This delegation process requires that the network connection be integrity-protected to prevent malicious parties from tampering with messages, but does not require encryption as no sensitive information is exchanged.

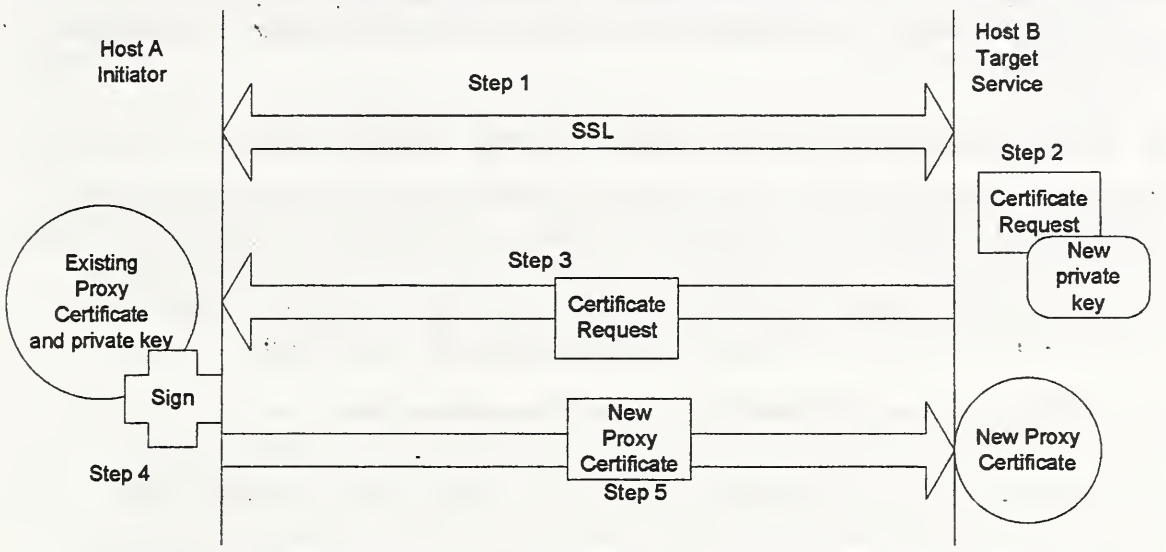


Figure 2: Delegation of a Proxy Certificate over a secured network connection. Steps are described in the text.

Figure 2 shows the steps involved in the delegation of privileges by creation of a Proxy Certificate over a network connection:

1. The initiator, on host A at left, connects to the target service on host B at right. The initiator and target service perform mutual authentication, the initiator using its existing Proxy Certificate and the target service uses the public key certificate of its own (not shown). After authentication, an integrity protected channel is established. These two steps can be accomplished by using the SSL protocol.
2. After the initiator expresses its desire to delegate by some application-specific means, the target service generates a new public and private key pair.
3. With the new public key, a signed certificate request is created and sent back over the secured channel to the initiator.
4. The initiator uses the private key associated with its own Proxy Certificate to sign the certificate request, generating a new Proxy Certificate containing the newly generated public key from the target service. The initiator fills in appropriate values for the fields in the Proxy Certificate as described in Section 3.1 as well as a policy describing the rights it wishes to delegate.

5. The new Proxy Certificate is sent back over the secured channel to the target service, which places it into a file with the newly generated private key. This new Proxy Certificate is then available for use on the target service for applications running on the user's behalf.

While the host receiving the delegated Proxy Certificate may have a long-term key pair of its own (bound to an X.509 public key certificate that it used for authentication), this key pair is typically not reused for the delegated Proxy Certificate. The reason is that a given host may have multiple users delegating privileges to it that are intended to be bound to specific processes and not shared across processes. The generation of a new key pair for each process greatly simplifies the task of keeping privileges compartmentalized. This approach also allows a user to "revoke" the delegation by deleting the Proxy Certificate private key as described in [13]. We discuss the performance ramifications of this approach in Section 7.

5 Authorization Models for Proxy Certificates

Proxy Certificates have three obvious modes of integration with authorization systems: full delegation of rights from the issuer—in effect, impersonation; no delegation of rights from the issuer, solely using attribute assertions to grant privileges; and a restricted delegation of some subset of the issuer's rights to the Proxy Certificate bearer. In this section we describe our experiences with each of these three methods.

5.1 Identity-based Authorization with Impersonation

In our initial implementations, we used Proxy Certificates almost exclusively as impersonation credentials that granted the bearer the full rights of the issuer. This approach has the advantage of integrating easily with identity-based authorization systems since these systems can simply treat the bearer of such a Proxy Certificate as they would its issuer. However, such usage is not ideal from the point of view of trying to achieve least privilege delegation since it only supports the full delegation of the issuer's rights. Thus, we explored other methods.

5.2 Proxy Certificates with Restricted Delegation

Proxy Certificates can be created with policies that delegate only a subset of the issuer's rights to the Proxy Certificate bearer. While this form of usage is more in line with the goal of enabling least privilege delegation, the implementation becomes more complex. As we described in Section 3.1, Proxy Certificates do not mandate any particular delegation language for the issuer to express their delegation policy, but instead provide a framework for containing policy statements using a method of the issuer's choosing.

The primary complication is that the relying party accepting the restricted Proxy Certificate must both understand the semantics of the delegation policy used and be able to enforce the restrictions that it imposes. Since these policies often contain application-specific restrictions, it is difficult for a security library handling the authentication of the Proxy Certificate to know what restrictions the application understands and is capable of enforcing. Without assurance that the application (or some other part of the software stack) will handle the enforcement of the restrictions, the authentication library cannot safely accept a restricted Proxy Certificate.

We have attempted to solve this problem by extending the API between the application and the security libraries to allow the application to express to the security libraries its knowledge and ability to handle given restriction delegation policies. However, this approach is difficult in practice since it must be done on a per-application basis. For this reason, we have not used this form of Proxy Certificate authorization to a large degree.

5.3 Identity Creation with Additional Assertions

The third method of using Proxy Certificates in authorization systems is to have Proxy Certificates convey no rights to the bearer (i.e., a policy type of "independent" as described in Section 3.1) and then use attribute assertions to assign rights to the bearer. This method has the advantage that attributes may be granted to the bearer from a number of different sources and may be done so at times other than the creation of the Proxy Certificate.

However, there are two difficulties in implementation of this method that have slowed our adoption:

- *Lack of protocol support:* The TLS protocol [6] and implementation of OpenSSL [32] (before the latest, version 0.9.7) lack support for X.509 attribute certificates. Thus, every application protocol must be modified to include a means of transporting attribute certificates. (We do note that our recent move to a web service based protocol [39] may ease this burden.)
- *Lack of granularity in enforcement systems:* Many enforcement systems do not have the ability to enforce any policies with finer granularity than simple groups. Although there has been some work in finer-grained enforcement [25,33,12,21,22] these results are not yet portable across all applications and operating systems.

6 Proxy Certificate Implementations and Applications

Here we briefly describe our implementation of Proxy Certificates and some applications that use Proxy Certificates.

6.1 Implementation in Globus Toolkit's Grid Security Infrastructure

The Grid Security Infrastructure (GSI) implements Proxy Certificates to provide authentication and delegation capabilities for the Globus Toolkit. It allows application users to employ proxy certificates to authenticate to GSI-based services and to delegate Proxy Certificates to those services so that they may act on the user's behalf.

GSI is primarily intended to work with identity-based authorization systems and as such returns to the calling application an identity for the remote client. It is further intended to be used primarily with Proxy Certificates that have policies delegating the full set of their issuer's rights to their bearer. In this case it returns the subject name from the X.509 public key certificate that issued the original Proxy Certificate in the chain. As we describe in Section 6.4, GSI has also been used successfully with a combination of Proxy Certificates and attribute assertions. The use of GSI with restricted Proxy Certificates has been hampered by the issues described in Section 5.2.

GSI includes a GSS-API [24] library, which handles authentication and delegation using Proxy Certificates. This library is based heavily on the OpenSSL [32] library, an open source implementation of the SSL protocol. The library uses OpenSSL to provide protocol support, including message protection and basic X.509 path validation. It adds to OpenSSL custom code for handling Proxy Certificates in addition to normal X.509 public key certificates and performing delegation.

6.2 MyProxy: An Proxy Certificate Repository

MyProxy [31,26] is a credential repository service that enables credential mobility and also alleviates the burden on users of managing and protecting files containing long-term secrets (i.e., private keys). We describe MyProxy briefly here, directing readers interested in more information to the references.

MyProxy is similar in function to a traditional credential repository as defined in the IETF SACRED working group [18]. However, by using Proxy Certificates it can operate without long-term private keys ever leaving the MyProxy service. MyProxy allows a user to establish a protected channel to the MyProxy service using SSL (without a client-side certificate), to authenticate over that channel from a remote system using, for example, a username and pass phrase, and then obtain a Proxy Certificate bearing their privileges without having to carry their long-term public key certificate and private key around with them (a potentially error-prone and insecure process).

6.3 Use in other Applications

The GSI libraries have also found uses in common applications. For example, Proxy Certificates can be used as an alternative authentication mechanism in secure shell (SSH) [15], CVS [14], and FTP [1]. These and other applications use the GSS-API library from GSI to allow a user to authenticate to an appropriate GSI-enabled daemon using their Proxy Certificate. The GSI-enabled SSH application also allows the user to delegate a new Proxy Certificate so that other GSI-enabled applications can be used on the remote system.

6.4 Proxy Certificates as Attribute Assertion Carriers

Combining public-key certificates with attribute assertions allow for the reuse of a single PKI across multiple application domains. In such a scenario, the PKI is used as a identity provider and all applications or domain specific privilege information (e.g., group memberships, clearance level, citizenship) is conveyed by separate attribute authorities.

However, as we mention in Section 5.3, many security protocols do not offer support for conveying attribute assertions. For example, the TLS protocol does not allow for attribute certificates in the set of provided client credentials. Thus, each application protocol must be modified to accommodate attribute assertions.

One way to circumvent this problem is by way of Proxy Certificates: when creating a Proxy Certificate, the proxy certificate issuer has the opportunity to add additional information to the proxy certificate by way of certificate extensions (in addition to the PCI extension described in Section 3.1). Several Grid projects use this technique to bundle application-specific attributes dynamically in the Proxy Certificate. The Community Authorization Service (CAS) [33,12] makes use of SAML authorization

decisions [34] to assert that the identity may perform (a group of) actions on (a group of) objects. The VO Membership Service (VOMS) [38] is a role-based authorization system that uses X.509 attribute certificates to assert a user's group membership(s), role(s), and capabilities. PRIMA [25] is a similar system that uses X.509 attribute certificates containing XACML [7] statements to assert a user's capabilities.

7 Performance and Security Issues

The expensive part of a Proxy Certificate creation is generating the new key pair. In this section, we only consider RSA key pairs due to lack of support in commonly used open source software stacks for alternatives, such as elliptic curve cryptography (ECC) [27].

Generating an RSA public key pair involves finding a pair of suitable prime numbers, which is a non-trivial amount of work that furthermore scales exponentially with the key length. Table 2 shows timings for key pair generation on a 2.8GHz Pentium 4 processor using the OpenSSL 0.9.7 library. We measure system CPU time and give averages over 100 keys.

Size (bits)	Time (seconds)
512	0.040
768	0.094
1024	0.176
1536	0.415
2048	1.348

Table 2: Key generation times for RSA key pairs

Unfortunately, use of specialized hardware such as cryptographic accelerators does not help these timings much, as such hardware is built with the assumption that RSA key generation occurs seldom and thus is not a performance sensitive operation.

Consequently, key generation of normal key sizes may consume a substantial amount of CPU for hosts receiving delegated Proxy Certificates from multiple clients. It is tempting to use smaller key sizes since the lifetime of a Proxy Certificate key pair is comparably short. (Indeed, the 3.0 release of the Globus Toolkit does just this.) While a smaller key size may yet meet the targets for complexity necessary to make brute force attacks infeasible within the short lifetime of the key pair, one must remember the cascading effects on the context in which such a key is used. For example, private data transferred during an ftp connection will typically remain sensitive long after the transfer is completed, and if an eavesdropper records the whole ftp transfer they have a longer period of time than the life of the key pair during which they may attack the protection it provided.

Thus, we note that Proxy Certificate generation comes with a non-negligible penalty in server-side key generation. Currently this means that services must take appropriate precautions when accepting Proxy Certificate delegations to prevent denial of service

attacks. At the time of writing, the development of solutions that mitigate this problem is left as future work.

8 Related work

A number of schemes offer delegation in a similar manner to Proxy Certificates. We discuss a few of these schemes here and compare them to our Proxy Certificate work.

8.1 Kerberos V5

The Kerberos Network Authentication Protocol [23,29] is a widely used authentication system based on conventional (shared secret key) cryptography. It provides support for single sign-on via creation of "Ticket Granting Tickets" (TGTs), and support for delegation of rights via "forwardable" and "proxyable" tickets. The initial use of proxy credentials in Kerberos was described by Neuman [30], who also described restricted proxy credentials and proposed several uses for them, including cascaded delegation (using a proxy credential that contains restrictions to generate a new proxy with greater restrictions), authorization servers (servers that grant restricted proxy credentials based on a database of authorization information), and group servers (servers that grant restricted proxy credentials that convey rights to assert membership in groups).

From the perspective of a user, applications using Kerberos 5 are similar to applications using X.509 Proxy Certificates. The features of Kerberos 5 tickets formed the basis of many of the ideas surrounding X.509 Proxy Certificates. For example, the local creation of a short-lived Proxy Certificate can be used to provide single sign-on in an X.509 PKI based system, just as creation of short-lived TGT allows for single sign-on in a Kerberos based system. And a Proxy Certificate can be delegated just as a forwardable ticket can be forwarded. Proxy Certificate and Kerberos also share the common method of protecting a TGT and protecting the private key of a Proxy Certificate by using local filesystem permissions.

The major difference between Kerberos TGTs and X.509 Proxy Certificates is that creation and delegation of a TGT requires the involvement of a third party (the Kerberos Domain Controller), while Proxy Certificates can be unilaterally created by their issuers without the active involvement of a third party.

8.2 X.509 Attribute Certificates

An X.509 attribute certificate (AC) [8] can be used to grant to a particular identity some attribute such as a role, clearance level, or alternative identity such as "charging identity" or "audit identity." Authorization decisions can then be made by combining information from the identity itself with signed attribute certificates providing binding of that identity to attributes. Attribute certificates can either be issued by a trusted entity specific to the issuance of attributes, known as an attribute authority, or by end entities delegating their own privileges.

In the case of an attribute authority, this method works equally well with attributes certificates bound to public key certificates or Proxy Certificates. For example, Proxy Certificates can be used to delegate the issuer's identity to various other parties who can claim attributes of the issuer. An AC could also be bound directly to a particular Proxy Certificate using the unique subject name from the Proxy Certificate.

The uses of ACs that are granted directly by end entities overlap considerably with the uses of Proxy Certificates. However, this AC based solution to delegation has some disadvantages as compared to the Proxy Certificate based solution:

- A similar modification to the validation framework, as in the Proxy Certificate RFC and described in Section 3.2, is needed in order to allow ACs to be signed by end entities.
- Identifying short-lived, dynamically created identities as described in Section 2.2, remains a non-resolved problem.
- All protocols, authentication code, and identity based authorization services must be modified to understand ACs.
- ACs must be created and signed by the long-term identity credentials of the end entity. This implies that the entity must know in advance which other identities may be involved in a particular task in order to generate the appropriate ACs. On the other hand, Proxy Certificates bearers can delegate privileges through the creation of new Proxy Certificates without interaction of the entity holding the long-term identity credentials.

We believe there are many unexplored tradeoffs between ACs and Proxy Certificates. Reasonable arguments can be made in favor of either an AC-based solution to delegation or a Proxy Certificate based solution to delegation. The approach to be taken in a given instance may depend on factors such as the software that it needs to be integrated into, the type of delegation required, and religion.

8.3 SPX

SPX [36] uses a structure entitled a "ticket" for delegation and single sign-on which is similar in purpose to Proxy Certificates. The two mechanisms share many common features: the SPX ticket is combined with a private key to provide a set of credentials to provide the means for authentication; the ticket and its private key are short-lived and normally stored in a file protected by file permissions; and the implementation uses the GSS-API as the application interface.

The main difference is that SPX defines its own format for the ticket and its own protocols for authentication. Proxy Certificates, being based on X.509 public key certificates, allow for a significant reuse of the existing protocols and software designed for those certificates.

Proxy Certificates also include the concept of a delegation policy (Section 3.1), which allows for arbitrary delegation of subsets of the issuers rights to the Proxy Certificate bearer. In contrast, SPX tickets only offer an impersonation mode.

8.4 Delegation in Digital's DSSA

Gasser and McDermott [13] describe a delegation scheme used in Digital's Distributed System Security Architecture (DSSA). This restricted public-key based delegation is similar to Proxy Certificates in that it allows for cascading delegation, has delegations bound to unique keys, and has similar motivations. The primarily difference between Proxy Certificates and the DSSA work is our starting from X.509 public key certificates

in order to allow for maximum protocol and software reuse. It is also unclear to what extent the DSSA work was implemented.

8.5 Future XML Alternatives

Proxy Certificates offer a pragmatic approach to delegation of rights in a SSL- and X.509-dominated world. By basing Proxy Certificates on the well established X.509 certificates, the Proxy Certificates chains are easily exchanged in the SSL authentication protocol. Furthermore, by embedding the delegation policy statements inside of the Proxy Certificate, these delegation directives are exchanged as part of the SSL authentication process

At this time, we appear to be moving towards a web services dominated world. We envision that pure XML-based alternatives to SSL/TLS will be invented for authentication and key exchange based on new and emerging specifications and standards, such as XML-Signature, XML-Encryption, WS-Trust, WS-SecureConversation, etc. We expect these new standards to be more authentication mechanism agnostic and supporting alternatives to X.509, such as PGP, SPKI, bare keys, etc. Furthermore, these protocols are also expected to be able to communicate attribute and authorization assertions transparently without requiring modification of the application protocol. Some of our initial work in this area is described in [39].

We are currently investigating these XML-based technologies as alternatives or enhancements to Proxy Certificates. For example, the equivalent functionality of a Proxy Certificate could be achieved through a fine-grained SAML [34] authorization assertion expressed or an XACML policy statement that empowers a bare key. The generation of this key and the issuing of this authorization assertion could follow the same procedure and pattern as we use for Proxy Certificates.

9 Summary

Standard X.509 identity and attribute certificates allow for the static assignment of identities and rights. However, some environments require that end entities be able to delegate and create identities quickly. We have described Proxy Certificates, a standard mechanism for dynamic delegation and identity creation in public key infrastructures. Proxy certificates are based on X.509 public key certificates in order to allow for significant reuse of protocols and open source software. Our Grid Security Infrastructure (GSI) implementation of Proxy Certificates exploits these opportunities for reuse to provide a widely used implementation of Proxy Certificate mechanisms. A number of applications and widespread deployment demonstrate the viability of Proxy Certificate mechanisms.

10 Acknowledgements

We are pleased to acknowledge significant contributions to the Proxy Certificate RFC by David Chadwick, Doug Engert, Jim Schaad, and Mary Thompson. We are also grateful to numerous colleagues for discussions regarding Proxy Certificates, in particular: Carlisle Adams, Joe Bester, Randy Butler, Keith Jackson, Steve Hanna, Russ Housley, Stephen Kent, Bill Johnston, Marty Humphrey, Sam Lang, Ellen McDermott, Clifford Neuman, and Gene Tsudik. Doug Engert coded the initial prototype implementation of Proxy

Certificates in GSI. Sam Meder, Jarek Gawor and Sam Lang coded the current implementations. We also thank Jim Basney and the anonymous members of the program committee for reviewing and commenting on early versions of this paper.

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under contracts W-31-109-Eng-38, DE-AC03-76SF0098, DE-FC03-99ER25397 and No. 53-4540-0080.

11 References

1. Allcock, B., et. al., Data Management and Transfer in High Performance Computational Grid Environments. *Parallel Computing Journal*, Vol. 28 (5), May 2002, pp. 749-771.
2. Butler, R., Engert, D. Foster, I. , Kesselman, C. , Tuecke, S. , Volmer, J. , and Welch, V. A National-Scale Authentication Infrastructure. *IEEE Computer*, 33(12):60-66, 2000.
3. Beiriger, J., Johnson, W., Bivens, H., Humphreys, S. and Rhea, R., Constructing the ASCI Grid. *In Proc. 9th IEEE Symposium on High Performance Distributed Computing*, 2000, IEEE Press.
4. Brunett, S., Czajkowski, K., Fitzgerald, S., Foster, I., Johnson, A., Kesselman, C., Leigh, J. and Tuecke, S., Application Experiences with the Globus Toolkit. *In Proc. 7th IEEE Symp. on High Performance Distributed Computing*, 1998, IEEE Press, 81-89.
5. CCITT Recommendation, X.509: The Directory – Authentication Framework. 1988.
6. Dierks, T. and Allen, C., The TLS Protocol Version 1.0, *RFC 2246*, IETF, 1999.
7. eXtensible Access Control Markup Language (XACML) 1.0 Specification, OASIS, February 2003.
8. Farrell, S. and Housley,R., An Internet Attribute Certificate Profile for Authorization, *RFC 3281*, IETF, April 2002.
9. Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A Security Architecture for Computational Grids. *ACM Conference on Computers and Security*, 1998, 83-91
10. Foster, I. and Kesselman, C. Computational Grids. Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 2-48.
11. Foster, I. and Kesselman, C. Globus: A Toolkit-Based Grid Architecture. Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 259-278.
12. Foster, I., Kesselman, C. , Pearlman, L., Tuecke, S., Welch, V. , The Community Authorization Service: Status and future. *Proceedings of the International Conference on Computing in High Energy Physics - CHEP 2003*.
13. Gasser, M. and McDermott, E., An Architecture for Practical Delegation in a Distributed System. *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, 1990, IEEE Press, 20-30.
14. gridCVS, *http://www.globus.org/gridcvs/*, 2002.
15. GSI-Enabled OpenSSH, *http://grid.ncsa.uiuc.edu/ssh/*, 2004.
16. Housley, R., Polk, W., Ford, W., and Solo, D., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *RFC 3280*, IETF, April 2002.
17. IETF Public-Key Infrastructure (X.509) (pkix) working group. *http://www.ietf.org/html.charters/pkix-charter.html*, January 2004.
18. IETF Securely Available Credentials (sacred) working group. *http://www.ietf.org/html.charters/sacred-charter.html*, 2003.
19. Johnston, W.E., Gannon, D. and Nitzberg, B., Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. *In Proc. 8th IEEE Symposium on High Performance Distributed Computing*, 1999, IEEE Press.

20. Kaliski, B., PKCS #10: Certification Request Syntax v1.5, *RFC 2314*, October 1997.
21. Keahey, K., Welch, V., Lang, S., Liu, B., Meder. S., Fine-Grain Authorization Policies in the GRID: Design and Implementation. *1st International Workshop on Middleware for Grid Computing*, 2003.
22. Keahey, K., and Welch, V. Fine-Grain Authorization for Resource Management in the Grid Environment. *Proceedings of Grid2002 Workshop*, 2002.
23. Kohl, J., and Neuman, C., The Kerberos Network Authentication Service (V5), *RFC 1510*, IETF, 1993.
24. Linn, J. Generic Security Service Application Program Interface, Version 2. *RFC 2078*, 1997.
25. Lorch, M., Adams, D., Kafura, D., Koneni, M., Rathi, A., and Shah, S., The PRIMA System for Privilege Management, Authorization and Enforcement in Grid Environments, *4th Int. Workshop on Grid Computing - Grid 2003*, 17 November 2003 in Phoenix, AR, USA.
26. Lorch, M., Basney, J., and Kafura, D., A Hardware-secured Credential Repository for Grid PKIs, *4th IEEE/ACM International Symposium on Cluster Computing and the Grid*, Chicago, Illilnois, April 19-22, 2004 (to appear).
27. Menezs, A., Elliptic Curve Public Key Cryptosystems, *Kluwer Academic Publishers, 1993*
28. Myers, M., et. al., X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, *RFC 2560*, IETF, June 1999.
29. Neuman, B. C. and Ts'o, T. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine*, *32* (9). 33-88. 1994.
30. Neuman, B.C. Proxy-Based Authorization and Accounting for Distributed Systems. *In Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283-291, May, 1993.
31. Novotny, J., Tuecke, S., and Welch, V., An Online Credential Repository for the Grid: MyProxy. *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.
32. OpenSSL, *http://www.openssl.org*, 2002.
33. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
34. Security Assertion Markup Language (SAML) 1.1 Specification, OASIS, November 2003.
35. Stevens, R., Woodward, P., DeFanti, T. and Catlett, C. From the I-WAY to the National Technology Grid. *Communications of the ACM*, 40(11):50-61. 1997.
36. Tardo, J.J. and Alagappan, K., SPX: global authentication using public key certificates, *Proceedings., 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, Vol., Iss., 20-22 May 1991. Pages:232-244
37. Tuecke, S., Welch, V. Engert, D., Thompson, M., and Pearlman, L., Internet X.509 Public Key Infrastructure Proxy Certificate Profile, *draft-ietf-pkix-proxy-10 (work in progress)*, IETF, 2003.
38. VOMS Architecture v1.1, *http://grid-auth.infn.it/docs/VOMS-v1_1.pdf*, May 2002
39. Welch, V., et. al. Security for Grid Services, *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*, IEEE Press, to appear June 2003.

Experiences of establishing trust in a distributed system operated by mutually distrusting parties

Scott Crawford, Enterprise Management Associates, Boulder, CO, USA
Email: scrawford@enterprisemanagement.com
David Chadwick, University of Salford, Salford, England, M5 4WT
Email: d.w.chadwick@salford.ac.uk

Introduction

The organization that is the subject of this case study is engaged in the worldwide monitoring of environmental information. This information provides evidence about the production of contaminants in one country that can be harmful to its neighbors. The project, which started in early 1999, was to develop an IT system that could authenticate data collected from widely distributed sources, in a manner that could be trusted by the participating countries, even though those countries might not trust each other.

Because of the potential political implications of this monitoring activity and the data collected, the subject organization represents the interests of the governments of the participating countries. Therefore, conclusions drawn from the collected data must be based on a reasonable degree of trust in the integrity and authenticity of the data and the data collection, archiving and distribution systems involved.

Because the interests of multiple sovereign and independent nations are involved, none of the participating nations is willing to subordinate its national interest to the subject organization by allowing the organization to speak on its behalf regarding the veracity of the data or any evidentiary conclusions that may be drawn or implied. For example, while a chemical or nuclear accident such as the 1984 Bhopal, India or 1986 Chernobyl disasters could conceivably produce contaminants indicative of hostile military activity, to draw such a conclusion from the data collected from a similar accident would be in error, but not outside the realms of possibility for one nation seeking to thwart the national interest of another nation in which such an accident had taken place. Therefore, one of the principal goals of the data authentication system was to assure that the trust placed in it – and, by extension, in the data itself – be a matter shared amongst, if not all, at least a significant enough number and distribution of participant nations to give a reasonable assurance to the organization as a whole that the integrity and veracity of the data is trustworthy. Each participant nation or any other observer would then be free to draw their own conclusions as they see fit.

To support this goal, the monitoring regime involved in collecting the data was developed along the following lines:

- Its human structure paralleled that of the organization itself. Its policy-making bodies were designed to be democratic and deliberative, and its operational staff developed along lines of proportional representation of participating nations, with oversight by the representative policy-making organs.

- Technical systems were developed to reflect the collective and representative nature of the organization. The data collection system was designed around the placement and distribution of monitoring sites worldwide, with locations distributed among as many participating countries as possible to monitor the global environment as a whole. A number of scientific disciplines were involved, for purposes of confirmation and cross-referencing of data indications.

Monitoring is continuous wherever possible. Monitoring sites have been networked with data management centers to enable timely collection and analysis. The data itself, as well as the data collection, archiving and distribution systems, are open to the scrutiny of all participating nations. The influence of any one or minority of participants on the data – particularly nations hosting monitoring and networking sites – should be minimized as far as possible. This was to be achieved by the participation of representatives of the subject organization in the construction, operation and maintenance of the data collection sites.

System Trust Requirements

These principal considerations influenced the nature of the IT system that was developed to authenticate the veracity and integrity of the collected data. Organizational policy-makers required the authentication system to implement an architecture that distributed the trust among the participants. Policy-makers further required the authentication architecture to parallel the construction of the data collection system and to be open to the highest possible scrutiny and periodic evaluation by representative groups. This requirement, however, had to be balanced against the need to protect the system and its individual sites and components from exploitation. For example, a malicious party seeking to blind the organization to polluting or contaminating activity in a specific location might seek to interfere with the monitoring ability of a site through interfering with its network connectivity or system operation. It also had to be balanced against the risks posed by a pragmatic need to delegate contractual, implementation and operational responsibilities to those having the necessary expertise. Such delegation was, however, subjected wherever possible to oversight by representative groups reflective of the collective nature of the subject organization as a whole. An overriding principle was that no part of the system installation or operation that formed part of the trust infrastructure, should be entrusted to a single individual.

Proposed Solution

Because the monitoring data could be represented as either a networked bitstream or a discrete message, it was determined that digital signatures could be applied as a means of assuring data authenticity. Pioneered by [DH 76] and elaborated in [RSA 78] and subsequent innovations and standardizations (e.g. the PKCS#1 standard for the RSA algorithm [PKCS 1]), public key cryptography (PKC) implements digital signatures through the combination of public/private keypairs and hash algorithms. Encryption of the data with a private key and successful decryption with the corresponding public key assures that only a specific private key could have performed the encryption. If the encrypted data is a "one-way" hash of the actual subject data, such as provided by the Secure Hash Algorithm (SHA-1, [FIPS 180-1]), it provides a tamper-evident assurance that the data has not been altered since encryption, when the decrypted hash matches one generated over the received data. The authentication system in this application was therefore centered on digital signing of the data at the monitoring site at the time of observation, and as close to the data source as possible so as to limit opportunities for data alteration. Streaming networked data could be divided into discrete transmission frames to which individual digital signatures could be applied.

But while digital signatures may provide a mechanism for authenticating data, of themselves, they do not address the issue of distributing trust amongst the participating nations. For example, a recipient needs to know which private keys have been installed at which data monitoring sites, and that it has the correct corresponding public keys in its possession.

Review of Previous Work

Prior to implementation, other relevant implementations of PKI technology were studied so as to provide insight into how trust can be distributed amongst competing, and possibly mutually distrusting, member organizations. Candidate organizations were multinational organizations involved in international finance, banking and exchange systems. The stakes of individual participants in these multinational organizations, concerning the authenticity of information and data exchange, which represents large sums of money, are at least as significant as the risks borne by the participants in the subject organization.

One of the most significant parallels was found in the establishment of Identrus LLC, which took place at approximately the same time as the early stages of the subject implementation. Founded in April 1999 by eight leading US and European banking and financial institutions, Identrus was created for the purpose of establishing an architecture of trust in electronic transactions between participating banks and institutions, and between their customer businesses as well ([Identrus 98], [Identrus 02a]). One of the original participants in Identrus was the US company CertCo ([Identrus 98]). CertCo was differentiated from its competitors at the time by its implementation (with IBM cryptographers) of threshold public key cryptography ([Ankney 00]). [Desmedt 92] describes the goal of threshold PKC as a scheme "in which the power to perform a certain operation is shared." In a threshold cryptosystem, the factors of a key are *distributed* among a group such that, when the group members contribute their factor components for combination enabling an encryption operation, they do so without divulging their individual components to each other. More to the point, a threshold cryptosystem requires a minimum *threshold* number m out of the total number n (described as "t-out-of-l" in [Desmedt 92]) of all possible participants to contribute their components in order to enable the encryption operation.

Threshold cryptography was therefore studied as a possible enabling technology for the distribution of trust between the cooperative yet mutually-distrusting participants in the subject organization. However, a threshold technique posed significant operational complications when considered for application of digital signatures at the data source of an environmental monitoring station. Instead, attention turned to a threshold implementation in the management of the data-signing keys. Because a system of digital signatures relies on the integrity of the private keys used to generate the signatures, a system of management of the corresponding public keys predicated on the then-current X.509v3 standard of digital certificates [X.509] was decided upon. The use of threshold cryptography in generating the digital signatures on the certificates of the issuing certificate authority (CA) was considered.

Threshold cryptography was not, however, a panacea without its own flaws. [Langford 96] illustrated certain vulnerabilities in systems then current: A colluding subgroup of the minimum required number of participants was able to manipulate a forgery of a threshold signature without the knowledge of the other participants (effectively reducing m to 2-out-of-n, regardless of the intended size of m). A malicious participant was able to influence public key generation such that they were enabled to discover the complete private key which is supposed to be unable to be discovered by any participant or used without the threshold number of participants. The conclusion drawn by [Langford 96] was that systems "without a trusted key generation center... are more complicated than those that do allow a single trusted center and are therefore more vulnerable to manipulation." [Desmedt 97] pointed out that not all threshold algorithms had progressed to an equal state of security in their development. In particular, [Desmedt 97] noted that, at the time, "no practical threshold [implementation of] DSS [the

Digital Signature Standard implementation of the Digital Signature Algorithm (DSA), now [FIPS 186-2]]...has been presented so far."

The lack of threshold DSS conflicted directly with the preference of a number of the participant nations for the use of DSS in data signatures. At the time, many national governments had concerns regarding the export of encryption technology, and did not want an organization representing their national interests to be accountable for potentially enabling undesired access, potentially worldwide, to an encryption technology such as RSA, in which either public or private keys could be used to encrypt digital information. DSS was therefore preferred, as DSS private keys could be used (in principle) *only* for signature generation, and the corresponding public keys *only* for signature verification. Thus, non-DSA-based algorithms – including threshold cryptosystems then available – were ruled out.

The example of distributed trust as manifested in *m*-out-of-*n* threshold key management and certificate authority implementations was, however, retained in a requirement to implement a distributed key management system. A "mixed" system of threshold-based certificates of DSS signing keys was briefly considered, but abandoned due to the above-related issues with threshold cryptography and algorithm preference, as well as the problems foreseen for a system of mixed algorithms. Instead, an administrative, rather than technical, implementation of *m*-out-of-*n* signature generation in the issuance of DSA-signed certificates was undertaken. After an evaluation of solution providers worldwide, the UK company Baltimore Technologies was selected to provide tools and systems for implementing an *m*-out-of-*n* key management system predicated on DSA. A number of other vendors from several different nations participated in the implementation of DSA signature generation software at the data sources.

Initial implementation

The Baltimore Technologies implementation was selected, in part, because of its flexibility in "customizing" a CA architecture to the needs of an organization, including its ability to use multiple Registration Authorities (RAs) and Registration Authority Operators (RAOs) to meet the administrative *m*-out-of-*n* requirement in the issuance of digital certificates. In Baltimore's PKI, RAs are client systems that submit requests for an X.509v3 digital certificate to an issuing CA server. RAOs are parties (usually humans) that interact with the RA to enable the approval of a certificate request for forwarding by the RA client to the CA. The certificate request comprises the public key to be certified and other relevant information about the key and its holder, formatted according to the PKCS#10 standard [PKCS 10]. Baltimore's PKI supports both single and multiple RAs interacting with a CA to request a certificate, as well as multiple RAOs interacting with an RA before a request can be sent to the CA. By mandating that multiple RAOs must request the same certificate to be issued for a data monitoring station, effectively distributes the trust placed in the operation of the CA to the number of RAOs that are involved in issuing the certificate requests.

The implementation was staged over periods of preliminary design, pilot testing, final design prior to initial implementation, and the initial implementation itself. Laboratory implementations of the data signing architecture were developed to test the processes of: keypair generation, certificate request and issuance involving *m*-out-of-*n* RAOs, signature of actual data, transmission of data and signatures via networks, management and retrieval of digital certificates, and the use of certificates in signature verification of data. Parameters and issues of general system operation and maintenance were also evaluated.

As preliminary system design took shape, distribution of trust in the system became manifested in a variety of ways beyond key management *per se*. As noted earlier, a number of scientific disciplines were involved in the monitoring regime, to give corroboration and cross-referencing of data supporting indications of specific contaminants and contaminating actions. Thus trust was distributed across a number of monitoring techniques, from measurement of atmospheric compounds to highly sensitive detection of vibrational information transmitted through the earth's oceans and the earth itself. Such multiplicity of data sources and types contribute to the weight of evidence in any given case, even in cases where signature-based authentication at any one monitoring site or minority of sites might be compromised.

Multiple parties were also involved in the construction and deployment of specific monitoring sites as well as the central data collection and management points supporting the system as a whole. In each case, representatives of the entire range of participating nations were involved, reducing the possibility of subversion of critical system components at virtually every key point.

Distributing Responsibility

Such a distribution of responsibility was not, however, without its cost. In the development of the authentication system, at least six, and sometimes more, different contractors spread throughout the world were involved in the detailed technical specification of the various components of authentication. In some cases, different contractors were delegated responsibility for the elaboration of the signature-implementation systems for different monitoring disciplines. Differences in standards were also required for different data transmission techniques (i.e. networked bitstreams versus discrete or "segmented" messages). The organization defined its own standard technique for signing streamed data by allowing a 40-byte space for a DSS signature in each transmission frame. Segmented message-format data had to be signed according to a standard that could be interpreted by both the implementing contractors and the subject organization. The standard chosen was that in most common use at the time, S/MIMEv2 (Secure Multipurpose Internet Mail Extensions) [RFC 2311]. S/MIME defines how to create a MIME body part that has been cryptographically protected according to [PKCS 7]. However, neither S/MIME nor PKCS#7 define the object identifier to be used with the DSA/DSS signing algorithm (they only specify ones for use with RSA). Therefore, accommodation was required among the contractors to enable the DSS signing algorithm. Laboratory implementations were ultimately successful using a variety of tools, including adaptation of open source reference implementations such as OpenSSL (then at version 0.9.5).

One of the implications of the unique nature of the monitoring regime was the necessity for custom developments in certain monitoring installations. For example, certain subterranean monitoring installations at deep levels below the earth's surface posed special problems for system endurance and form factor, as did underwater detectors placed beneath the ocean's surface. In certain cases, placing signature-generation devices at the *exact* point of data collection were impractical. In many cases significant barriers and challenges had to be overcome. For example, in some installations, the technologies necessary to compose standard certificate requests strictly formatted to PKCS#10 were beyond the physical and technological constraints of the systems at their then-current state of development. The solution to this involved on-site personnel obtaining "raw" public key information from such devices. The absence of a formal PKCS#10 request and an associated signature generated by the corresponding private key (which, when verified by the public key contained within the

PKCS#10 request, is a crucial step in demonstrating certificate request integrity and private key ownership) would be compensated for by the presence of on-site observers who received and verified the integrity of the generated public key from the remote constrained detector. The public key material thus obtained would then be sent in one or more PKCS#7-compliant messages to a key management center by the on-site observers. At the key management center, an adaptation of certificate issuance systems was developed to permit direct submission of such public key material to the CA when verified by the RAOs. Laboratory tests of this combination of techniques were successful in obtaining a certificate for a keypair generated in this manner. Demonstration of the integrity of the process was verified by the auditable recording of participant actions in order to preserve the "chain of trust".

This technique illustrates one example of how the presence and participation of multiple persons at virtually every crucial step of the authentication system became essential to establishing and maintaining the concept of distribution of trust, necessary for the system as a whole. Implicit in such a system, however, is the necessity of informed human participation; but this, after all, is to be expected in a system predicated on trust, which is essentially a human phenomenon. A certificate-authority-based key management architecture is, by definition, based on an assumption of trust in the authority itself. Trust, however, may be interpreted and manifested in a multiplicity of ways ([Mayer et al 95], [AJ 98], [Kramer 99]). Multiple parties may not – perhaps *will* not – all agree on their individual perceptions of what is trustworthy and what is not. However, the assumptions made in the design of this system considered that when a significant number of participants were agreed that they could place their trust in a system consisting of a number of verifiable measures and components, the requirement for trust distribution would be satisfied.

This also, however, implied that a certain number of duplications in implementation would be necessary to assure the necessary participation of multiple parties at significant points in the architecture. No one person could be allowed to operate alone in the presence of crucial system components, when those components might be susceptible to exploitation by an individual. The system would have to enforce multiple authorizations for access, manipulation and control beyond the requirement of *m*-out-of-*n* necessary for certificate issuance. The possibility existed that system operators might be required to be responsible for several components such as smartcards and other tokens necessary to enable operation of certain system elements. Backups of key material would have to be distributed among a number of points, all in an auditable fashion.

To meet these exigencies, a minimum set of qualities were sought as design goals. The threshold number of persons or components among which crucial elements of the system would be distributed would be kept to as practical a minimum as possible without subjecting the system to the susceptibility of individual operators. This did not rule out the actions of a malicious minority in all cases, but the sheer preponderance of numbers of persons and steps toward authentication involved throughout the architecture mitigated the possibility of such isolated actions subverting the system as a whole. Standards of procedure and operation would also be developed, with the intent that persons interacting with the system would be informed and knowledgeable. Operators would be instructed regarding what they would be doing and the reasons why trust in the system would be enabled by their actions, while those depending on the system for trusted demonstrations of authenticity would be aware of how and why the system should be trusted. System operations as well as the signed data itself would be auditable and open to scrutiny by appropriate parties, thus fostering the openness necessary to the development

of trust described in [Mayer et al 95], [AJ 98], [Kramer 99], and others. The use of OpenSSL in bespoke development of authentication components, for example, enabled clear examination of source code used in implementing authentication. Wherever possible, similar cooperation was obtained from contractors, sometimes in the form of "source code escrow," preserving the contractor's proprietary rights in maintaining source code confidentiality while enabling the organization to have the option of source code review should it be desired.

It would be inevitable that, beyond outright exploit, human as well as technical errors would eventually begin to be manifested in such a system, perhaps posing a more significant threat than malicious exploit. Again, however, the preponderance of the number of monitoring sites, the numbers of points throughout the architecture in which multiple parties would be involved, and the numbers of persons involved in critical operations, mitigated the potential consequences of any one error or a small number of errors. Added to these factors are data management systems at the data centers receiving the signed data that are able to alert operators when signature verification failures occur. The data centers hosted by individual participating nations help to verify the validity of such incidents and may themselves track such occurrences independently, thus helping to keep them from being hidden in a possible exploit scenario. Thus, a general development of "trust by consensus" in which the number of individual actions and steps in data authentication accumulate towards a body of data supporting trust, began to emerge as the system design progressed.

In summary, a description of the initial implementation proposed for the distributed management of trust is as follows. At the time a monitoring site is to be enabled with a digital signature capability, an on-site team of operators generates the keypair. If satisfied with the key generation process and the integrity of the resulting keypair, the on-site team then forwards the resulting public key to the key management center in one or more PKCS#7-compliant messages bearing the digital signatures of the on-site observers. At the key management center, the signatures of the received messages are verified against the signer's certificate(s) by a group of authorized RAOs. If a minimum *m* out of a total number of *n* RAOs agree that the signed message(s) containing the submitted public key are trustworthy based on signature verification and other verifications of the on-site observers' presence at the site, the RAOs approve the certificate request, and the certificate is duly issued by the CA. Signed data thereafter received from the site is verified on receipt at the data management center by signature verification using the issued certificate accessed from a local directory of certificates and certificate revocation lists (CRLs are as defined in edition 3 of X.509 [X.509]). This directory is accessed according to the Lightweight Directory Access Protocol (LDAP) described in [RFC 1777] and [RFC 2251], with an organizational namespace rooted on the name of the organization itself (being, as it is, an international entity).

System Maintenance

Yet to be fully elaborated are issues of key rollover and replacement of valid keys. The assumption to date has been that as the current key lifetimes reach their pre-determined limit (set to a minimum of 5 years) PKC technology will have matured to the point where a more evolved implementation may be indicated. Regardless, a preliminary protocol has been worked out, in which a currently trusted signing key is used to "countersign" the certificate request generated for a new keypair. Thus, a data generating system needs to be able to "cache" a currently valid keypair while awaiting issuance of the replacement keypair's certificate and authorization to use the new signing key. In cases where such caching is not possible, data would need to be signed

immediately with the new signing key. Authentication by the recipient is then contingent upon the issuance of the replacement certificate for the new keypair, during which time the validity of the site's data would be in a state of suspension. In any event, under such a scheme a new keypair would not be generated except on the site's receipt of an authenticated (digitally-signed) command issued by a minimum number of authorized parties, identified by certificates available to the site itself. Unauthorized keypair generation messages would be detected when data signed by an unauthorized key is received at the data management center. No authorized certificate would be available for data verification, and authentication would fail. In this case data from such a site would be "suspended" from authentication until on-site remediation was undertaken to restore the site to a trusted state.

In such a scenario, monitoring sites would need to have certificates of authorized command-issuers available to them, in order to enable command authentication. For any site installation, a certain number of individuals and groups must be delegated the responsibility for trusted operations on the site. Again, the limits of trust related to the number of individuals authorized to operate at a site is mitigated by the numbers of sites, the numbers of persons involved, and the oversight of such operations made possible by open scrutiny and the auditability of actions. In its role as the facilitating entity for the regime as a whole, the subject organization is able to call on on-site representatives and other means to monitor and corroborate site changes. It is also able to track any site changes that are authorized, thereby further mitigating the risks arising from trust placed in the site.

At the site, authorized signed commands are distinguished by the positive identification of the command-issuer through verification of the issuer's digital certificate. Without such verification, commands are ignored. To maintain the availability of the most up-to-date certificates, as well as information regarding suspended or revoked certificates, two methodologies were proposed. One was network-based access to directories containing certificates and CRLs; the other was on-site storage and maintenance of the necessary certificates and CRLs. The ultimate goal in the future is network enablement of certificate status checking for the most timely validation by the monitoring sites, using a protocol such as OCSP ([RFC 2560]). However, not all sites are currently capable of supporting such technological demands. On-site storage of current necessary certificates and CRLs will supplement such sites. In such cases, it is possible that, for example, an authorized individual whose authorization has been revoked may command a site as yet unaware of the revocation. Such cases are mitigated by limitations on access to the command-and-control functionality itself, and by requiring more than one individual to issue the same command (but this latter functionality has not been implemented yet). Also, the number of sites and persons involved spreads the individual risks. In no case would any one individual be authorized to command more than a significant minority of sites.

Local vulnerabilities in the monitoring sites are mitigated through a number of measures intended to develop tamper-evident installations that generate alerts whenever a site exploit is attempted. This is enabled through the triggering of functionality that includes site ill-state-of-health information in the data flow indicating that the site has been accessed. Cutting off the site's network connectivity in order to "blind" either the subject organization to an exploit or the site to the existence of, e.g., revoked command-issuer certificates, is detected by the absence of expected data from the site. This data cannot be mimicked to obscure the exploit without the attacker having access to the signing keys. Replay of valid data is not an option either, since the data contains replay detection information. Even scheduled maintenance may produce alerts of site intrusion, but such alerts are verified as scheduled maintenance from published operational

plans. There are, of course, limitations to the amount of trust that can be placed in such measures, but the "trust horizon" relative to the number of persons involved and their motivation to exploit is limited to the extent practical to the maintenance and purpose of the organization itself.

Initial results of implementation

At the time of writing, approximately one-fourth to one-third of monitoring sites have been equipped with the initial implementation of digital signatures and are sending authenticated data to the data management centers. While authentication of data in these cases has been successful, some of the most significant results are as follows.

The human interaction necessary to enable the system operation described above has been considerable. In particular, one of the author's personal experience in orienting operational staff and users to the system indicates that the learning curve alone is significant. Simply orienting users to the nature and operation of public key cryptography has been an abstraction difficult to communicate in many cases. Compensating for such challenges has been the high motivation and dedication of the subject organization's participants to fully understand the system. Thus, it is our subjective opinion that motivation is a significant factor in the success of such a trust-based system. In addition the aptitude of users to understand the operation of public key cryptography, as well as the ability of system developers to communicate the information essential to understanding, are essential to success.

The burden authentication technology places on data management systems themselves should not be overlooked. Performance measures of the time and resources necessary to authenticate a large number of DSS signatures received from stations continuously transmitting proprietary data protocol frames is not insignificant. System capacity planning and development is still taking shape to accommodate such demands. Computational and network resources are not the only demands placed on an organization seeking to implement a system such as this. Measures necessary to assure the security of all aspects of the implementation also take a toll on both human and material resources. A higher degree of vigilance and standardization of operational policies and procedures is necessary to assure the integrity of the system itself.

Nevertheless, the general consensus among users at this point appears to be divided between those who feel they understand the authentication system and those who do not. Surveys of users are currently underway to establish quantitative measures of success or failure of the implementation. Until they are received and analyzed, interviews with current system participants indicate that those who manifest an understanding of the system are satisfied that the system is functioning successfully. Nevertheless, they are not happy with the burden imposed by both the procedural and computational requirements of this distributed-trust implementation of digital-signature-based authentication. They are also less than satisfied with the "usability" of the human-interactive components of the system, which can be cumbersome and require the necessary understandings which can be challenging to communicate effectively to the involved personnel, as described above. Among those who do not express a high understanding of the system, the above-described burdens appear to be regarded as excessive relative to the benefits that are derived. It is not yet clear whether further education and dissemination of information regarding the system and its necessity to the requirements of the organization would help alleviate such concerns. However, future developments in the system will almost certainly take such measures into account.

Summary and conclusions

The most obvious conclusion drawn from this experience is that "distributed trust" means, first and foremost, distribution among people. While that statement may seem so apparent as not to even require being made, consider that first- and second-generation public key infrastructures (PKIs) such as this almost universally began as technological exercises. Focus on the algorithms and the technology of encryption and digital signature led to a number of implementations which did not sufficiently consider the human factors of trust, as well as other human factors such as how people perceive technology, how such perceptions affect their use of technology, and the relationship of such perceptions to assumptions – correct or not – made by system planners and developers, particularly as affects the success or failure of security technologies. This, in turn, led to the development of a framework for certification practice and certificate policies such as that described in [RFC 2527], but policies alone are not enough to compensate for the involvement of human beings in the technology of trust. Studies of human factors in security implementations such as [WT 98] and [WT 99] demonstrate that what often begins as a technology exercise often ends, successfully or not, as an exercise in implementing an appropriate understanding of the human factors involved. It is our belief that the technological limits of security implementation are often subject to the fact that both security and trust are fundamentally human concepts.

In the case of this implementation, the key goal was to implement a system in which all participants could trust the distributed data as far as it was possible and practical, notwithstanding the political nature of the organization and the lack of trust between the participants. It was essential to prevent a minority of persons with malicious intent from being able to subvert or exploit the data authentication system. Initially, this effort focused on the technology of distributing trust as manifested in the threshold cryptography system of digital signatures. Ultimately, the system has become one where trust is dependent upon the sheer numbers of people and systems that are involved, viz.: the number of points of data collection and their worldwide distribution necessary to obtain a reasonable number of overlapping systems and techniques of measurement; the numbers of participating nations and their representatives; the numbers of individuals involved in critical steps in the authentication process; and the volume of data itself. It must be noted that each of these factors was in existence in this organization before the authentication system itself was undertaken. Therefore, it would seem that authentication is dependent on the existence of the underlying factors that enable the necessary distribution of trust; the authentication system itself does not enable or distribute trust independent of these pre-existing factors.

Nevertheless, the system may at this point be judged a qualified success, in so far as it has succeeded in enabling a tangible measure of trust in the authenticity of the signed data, through the participation of a number of capable systems and motivated, knowledgeable individuals. However, the organization is distinguished as one which attracts individuals from throughout the world who are motivated to see it succeed. While this may in many respects be true of most professional organizations, the subject organization is able to call on the resources of national governments owing to the political nature of its existence. While environmental monitoring may not be a high priority with many participating governments, it nevertheless distinguishes the organization from, for example, those in the private sector with more limited resources. Only those organizations not just capable of, but also having a mandate for, fielding the necessary resources in terms of motivated, knowledgeable staff and technological capacity and development would likely be interested in such an undertaking. It is therefore not surprising

that architectures for building trust into distributed systems run by mutually distrusting or wary parties have to date only been undertaken primarily by banks, international financial institutions and other entities operating in the arena of marketing top-level trust assurance. More recently, international military coalitions [FRD 02] have also been shown to have the resources and mission to do so. This may be at least partly attributable to the potentially cumbersome nature of X.509-based hierarchical PKIs and their related technologies. Developments such as SPKI [IETF 01], authorization-based certificates and "federated" trust architectures such as the Internet2 Shibboleth project [I2 03] may succeed in helping to shape trust technology more closely to the realities of human use and interaction, particularly in more common, less well-endowed environments, but as yet it is too early to say so conclusively.

As a final note, the authors wish to state that the content of this paper represents the authors' own views. The authors do not represent or speak for or on behalf of the subject organization, nor should any statement in this paper be so construed.

References

[AJ 98]: P. J. Ambrose, G. J. Johnson. A Trust Based Model of Buying Behavior in Electronic Retailing. Association for Information Systems 1998 Americas Conference Proceedings, pp. 263-265. In http://www.isworld.org/ais.ac.98/proceedings/track06/ambrose.pdf.

[Ankney 00]: R. C. Ankney. Certificate management standards. CertCo, Inc., 2000, in http://www.certco.com/pdf/cms.pdf.

[Desmedt 92]: Y. Desmedt. Threshold cryptosystems. In J. Seberry, Y. Zheng, eds. AUSCRYPT '92. Lecture Notes in Computer Science 718. Springer-Verlag, Berlin/Heidelberg/New York, 1993, pp. 3-14.

[DH 76]: W. Diffie, M. E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, IT-22:644–654, 1976.

[FIPS 180-1]: National Institute of Standards and Technology, US Department of Commerce. Secure Hash Standard. 17 April 1995, in http://csrc.nist.gov/publications/fips/fips180-1/fips180-1.pdf.

[FIPS 186-2]: National Institute of Standards and Technology, US Department of Commerce. Digital Signature Standard. 27 January 2000, in http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf.

[FRD 02]: G. Fink, S. Raiszadeh, T. Dean. Experiences establishing an international coalition public key infrastructure. 1st International PKI Research Workshop – Proceedings. April 2002, pp. 193-206, in http://www.cs.dartmouth.edu/~pki02/Fink/paper.pdf.

[I2 03]: Internet2 Consortium. Shibboleth project. July 2003, in http://shibboleth.internet2.edu.

[Identrus 98]: Identrus LLC press release. Major financial institutions announce new company to provide businesses globally with a single electronic identity. 21 October 1998, in http://204.141.53.14/knowledge/releases/us/release_102198.xml.

[Identrus 02a]: Identrus LLC. About Identrus. 2002, in http://www.identrus.com/community/index.xml.

[IETF 01]: Internet Engineering Task Force. Simple Public Key Infrastructure (spki). IETF Charter, 16 January 2001, in http://www.ietf.org/html.charters/spki-charter.html.

[Kramer 99]: R. M. Kramer. Trust and distrust in organizations: Emerging perspectives, enduring questions. Annual Review of Psychology, vol. 50, pp. 569-598. Annual Reviews, Palo Alto, CA, 1999.

[Langford 96]: S. K. Langford. Weaknesses in some threshold cryptosystems. In N. Koblitz, ed. CRYPTO '96. Lecture Notes in Computer Science 1109. Springer-Verlag, Berlin/Heidelberg/New York, 1996, pp. 74-82.

[Mayer et al 95]: R. C. Mayer, J. H. Davis, F. D. Schoorman. An integrative model of organizational trust. Academy of Management Review. Vol. 20 no. 3, 1995, pp. 709-734.

[PKCS 1]: RSA Laboratories, Inc. PKCS #1 v2.1: RSA cryptography standard. 14 June 2002, in ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.doc.

[PKCS 7]: B. Kaliski. PKCS #7: Cryptographic message syntax version 1.5. Request for Comments 2315. IETF Network Working Group, March 1998, in http://www.ietf.org/rfc/rfc2315.txt.

[PKCS 10]: M. Nystrom, B. Kaliski. PKCS #10: Certification request syntax specification version 1.7. Request for Comments 2986. IETF Network Working Group, November 2000, in http://www.ietf.org/rfc/rfc2986.txt.
[RFC 2311]: S. Dusse et al. S/MIME Version 2 Message Specification. Request for Comments 2311. IETF Network Working Group, March 1998, in http://www.ietf.org/rfc/rfc2311.txt.
[RFC 2527]: S. Chokhani, W. Ford. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. Request for Comments 2527. IETF Network Working Group, March 1999, in http://www.ietf.org/rfc/rfc2527.txt.
[RFC 2560] 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. IETF Network Working Group, June 1999, in http://www.ietf.org/rfc/rfc2527.txt.
[RSA 78]: R. Rivest, A. Shamir, L. Adelman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, 21 (2), pp. 120-126, February 1978.
[WT 98]: A. Whitten, J. D. Tygar. Usability of Security: A Case Study. Carnegie Mellon University School of Computer Science Technical Report CMU-CS-98-155, December 1998, in http://reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-155.pdf.
[WT 99]: A. Whitten, J. D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. Proceedings of the 8th USENIX Security Symposium, August 1999, in http://www.usenix.org/publications/library/proceedings/sec99/whitten.html.
[X.500]: ITU-T (formerly CCITT) Recommendation X.500 / ISO/IEC/ITU Standard 9594 (series). Information Technology – Open Systems Interconnection – The Directory. 1988.
[X.509]: ITU-T (formerly CCITT) Recommendation X.509 (1997 E) / ISO/IEC/ITU Standard 9594-8. Information Technology – Open Systems Interconnection – The Directory, Part 8: Authentication Framework. June 1997.

Trusted Archiving

Santosh Chokhani & Carl Wallace
Orion Security Solutions

Abstract

Digital signatures are a powerful tool for demonstrating data integrity and performing source authentication. Timestamps are a powerful tool for confirming data existence by a particular point in time. Today, the value of digital signatures (and timestamps containing digital signatures) is limited due to a lack of tools and techniques that address the problems associated with digital signatures that accrue over time, including: expiration, revocation, cryptanalytic advances and computational advances. In this paper, we describe a system concept and protocol to achieve secure storage of data for long periods with preservation of integrity. The approach uses periodically refreshed time stamps to address these problems. The techniques can be used for a wide variety of applications, including those requiring long-term non-repudiation of digital signatures. The concept and protocol are based on minimizing trust in individual system components in order to reduce the security requirements for those components and to enhance the trust in the overall system. A proof-of-concept implementation based on the ideas and protocol described in this paper has been developed and successfully tested.

1. Introduction

One of the challenges of using digital signatures is how to prove the validity of signatures well into the future when the signer's, or a related certification authority's, credentials are no longer valid or available. Trusted archiving is a process that involves the active storage of data where evidence is periodically obtained, or generated, and stored to create an unbroken history demonstrating the integrity of data from storage time to verification time. Trusted archives are a missing piece of the PKI puzzle that are required if digital signatures are to have a durability similar to paper and ink signatures.

We have designed and developed a client-server system that addresses this problem. This paper describes our work. Section 2 contains the system concept. Section 3 provides an overview of the client-server protocol for implementing the system. Section 4 describes some security considerations. Section 5 provides a summary of the implemented system*. Section 6 describes lessons learned. Section 7 describes future plans.

2. Trusted Archive System Concept

A trusted archive should meet the following requirements, at a minimum:

- Provide evidence to demonstrate the integrity and, optionally, the source of data after the expiration of the cryptanalysis period for related keys and algorithms.
- Provide evidence to demonstrate the integrity and, optionally, source of data if a related certification authority (CA) is no longer operational.
- Provide active controls to protect the integrity of archived information.†

The central component of the solution is a trusted archive authority (TAA). A TAA accepts data for long-term storage and is responsible for ensuring that an evidence trail is produced and stored to enable demonstration of data integrity at any point in the future. TAAs participate in client-server transactions

* The ideas and work described in this paper (including the proof-of-concept) were funded by the United States Marine Corps.

† Many cryptographic mechanisms (such as digital signature or HMAC) are detection mechanisms with regard to integrity and source authentication. From a practical viewpoint, a trusted archive service needs to ensure that the archived information is protected from tampering. The mechanisms described in this paper extend the detection mechanisms and are not a substitute for secure, redundant storage, tamper protection, etc.

with entities seeking to exercise the TAA's services. TAAs use current credentials to generate signed responses as part of these transactions. Clients verify TAA signatures using a trust anchor known to the client at the time of the transaction.

Upon submission and periodically thereafter, the TAA obtains or generates a new time stamp for archived data in order to account for cryptanalytic advances against hashing or signature algorithms and to account for expiration of TSA keys. This periodic acquisition of new time stamps is referred to as "time stamp refresh" throughout the remainder of this document. The amount of trust invested in a TAA can be minimized by using the services of a trusted time stamp authority (TSA) to obtain time stamps for archived data instead of generating timestamps directly.

The client-server protocol between the client and TAA is a simple set of request/response transactions that enable submission of data to a TAA and retrieval or deletion[‡] of data from a TAA. The transactions are defined in ASN.1 and, generally, are DER encoded. Cryptographic Message Syntax (CMS), defined in [CMS], is used for all digital signatures. CMS was chosen because it is an IETF standard, many products support it, it provides flexibility to apply the cryptographic services appropriate for the application, and it provides the flexibility to include time stamps, certificates, revocation information, etc. as needed. An ASN.1 based protocol was chosen due to the requirement for an ASN.1 encoder/decoder to process most PKI artifacts. An XML submission format could be defined to provide a broad entry to a TAA. Retrieval should be sufficiently rare and in need of special purpose software, e.g. for historic algorithms, to be sustainable by a single format.

The TAA is designed to securely archive information of any type and need not have knowledge of the format of archived data. Where archived data contains digital signatures that must be verifiable in the future, collection and packaging of the items required to support signature verification are the responsibility of the archive submitter. Given the likelihood that the submitter will have performed verification, this requirement is not particularly onerous and can be easily implemented by packaging the artifacts from that verification operation, e.g. trust anchors, certificates, Certificate Revocation Lists (CRLs), Online Certificate Status Protocol (OCSP) responses, Simple Certificate Validation Protocol (SCVP) responses, etc., with the data to archive prior to submission to the TAA. Alternatively, a TAA may provide server-side verification services to simplify and streamline the process of verifying and archiving data.

Trust anchors will come and go over the course of time but always must be obtained in a trusted manner to support certificate path validation. A TAA may archive a set of trust anchors that can be provided to retrieval clients. This capability allows the retriever to validate a digital signature without having to rely on the good intentions of the original submitter. This capability also permits a functional separation in order to enhance the trustworthiness of the archival service, i.e. one TAA can be store archived data and evidence and another TAA can store trust anchors.

In summary, the system concept consists of the following:

- TAAs use digital signatures to demonstrate the integrity and source of responses from the TAA. This is primarily a concern where responses contain trust anchors.
- TAAs periodically refresh time stamps in order to protect against advances in technology that can break hash and signature algorithms and to maintain verifiability in cases of key (or certificate) expiration.
- Archive submission clients collect and submit all information (e.g., certificates, revocation information, SCVP responses, etc.) required for long-term non-repudiation of digital signatures that cover the data submitted to a TAA.
- Archive retrieval clients verify the signatures on the TAA response and on the associated archive record to confirm the integrity of the data. The retrieval client may use trust anchors from one or more of the following sources to verify signatures contained in the evidence record or in archived data itself, if the archived data was signed:
    - Trust anchors from the signed retrieval response.
    - Trust anchors obtained independently from the same or different TAA.
    - Trust anchors known to the retrieval client or obtained via other out-of-band means.

[‡] Where a TAA maintains archived data on write-only media, deletion may simply be cessation of refresh operations rather than actual deletion. Deletion is not addressed in detail in this document.

3. Trusted Archive Protocol (TAP)

3.1 Assumptions and Background

The Trusted Archive Protocol (TAP) was developed and submitted to the IETF as an Internet Draft (I-D) of the PKIX working group. The TAP I-D was a contributing factor in the formation of the IETF Long Term Archive and Notary (LTANS) working group (WG) and has served as input to the protocol being produced by that group. Activities of the LTANS WG and its relationship to this work are described in Further Research section.

TAP was designed using the following principles:

- The CMS will be used in all cases where digital signatures are applied.
- A TAA shall provide an archive submitter a response that includes a time stamp token, identifying information and a TAA-generated digital signature. Clients can verify the timestamp token to confirm the correct data was received and (presumably) archived by the TAA.
- The TAA shall verify the time stamp token received from the TSA in accordance with RFC 3161 [TSP].
- The TAA shall periodically refresh the time stamp token.
- The TAA shall provide all timestamps obtained for the archived data in the response.

The rest of this section provides a summary of the protocol defined in [TAP].

3.2 Definitions

During the development of TAP, the need arose for a common vocabulary describing the various processes and artifacts involved in archiving. The following terms were defined to meet this need:

Archived data: archived data is the data presented to the TAA by the submitter.

Archive token: an archive token is an object generated by the TAA when data is submitted and accepted for archiving. The archive token is returned to the submitter and may be used to request retrieval or deletion of the archived data and associated cryptographic information. For purposes of future retrieval or deletion, applications may treat the archive token as an opaque blob. The archive token includes: submitter DN, timestamp token, TAA date and time upon submission and, optionally, tracking information.

Archive record: an archive record contains the cryptographic refresh history compiled by the TAA. The initial archive record is the timestamp token obtained for the submitted data. The timestamp token format is defined in [TSP] and consists of a ContentInfo object containing a TSTInfo object. Upon each refresh, the most recent archive record becomes the prevArchRecord field of a new TimeStampedData object, a timestamp is obtained for the TimeStampedData object and is placed in the timestamp field of a new ArchiveRecordData and the entire ArchiveRecordData structure placed in a ContentInfo object. The ContentInfo object serves as the new archive record. When verifying an archive record, verification terminates when the original timestamp token is verified against the archived data.

Archive package: an archive package is an object containing, minimally, the archive token, archive record and archived data. The archive package may include additional cryptographic information. Archive packages are returned during retrieval.

Figure 1 illustrates the communication protocol among the TAA and the clients.

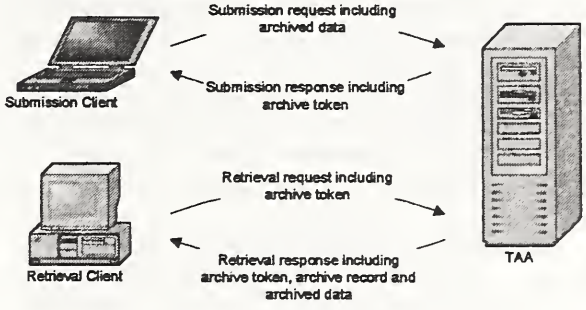


Figure 1 Client interactions with TAA

Figure 2 illustrates the archive record that is maintained by the TAA. The archive record contains nested timestamps with a timestamp covering the archived data at its innermost layer.

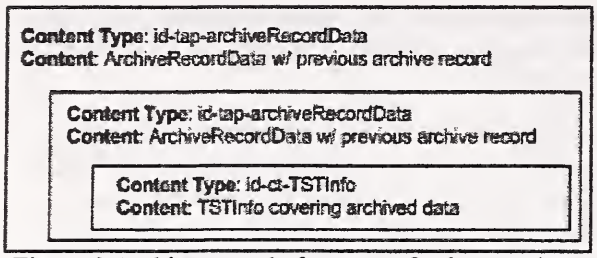


Figure 2 Archive record after two refresh operations

3.3 Protocol Summary

The [TAP] protocol defines 3 request types: submission, retrieval and deletion. The steps involved in a submission request are as follows:

1) A client prepares a data object for submission to a TAA. If the data object is signed, the client verifies the object and includes the necessary material to verify the object (except the trust anchor) in the object itself, e.g. in a certificate or CRL bag. Optionally, the client signs the request. The client sends the request to the TAA.

2) The TAA receives the request and verifies the signature on the request, if present. The TAA unpacks the data object and prepares a TSP request. Optionally, the TAA signs the TSP request. The TAA then sends the TSP request to a TSA.

3) The TSA receives the response and verifies the signature on the request, if present. The TSA then generates a signed timestamp token and returns it to the TAA.

4) The TAA stores the archived data and the timestamp token and starts the refresh clock for the archived data. The timestamp token is packaged in an archive token along with additional information. The archive token is included in a signed response and returned to the client.

5) The client verifies the signature on the response. The client verifies the archive token to ensure the correct data was archived by the TAA. The client may store the archive data along with the original data item, e.g. as an unsigned attribute.

The steps involved in a retrieval request are as follows:

1) A client prepares a retrieval request containing the archive token of the data item for which an archive record is required. The client signs the request and sends it to the TAA.

2) The TAA verifies the signature on the request and confirms the requestor has access to the requested data item. The TAA prepares an ArchivePackage containing the refresh history compiled for the requested data item, packages it in a signed response and returns it to the client.

3) The client verifies the signature on the response then verifies the archive package. The outermost layer in the archive record is verified using a current trust anchor. Interior layers are verified to a trust anchor provided by the TAA in the archive package.

3.4 Protocol Data Formats

The section describes some of the key data formats defined in [TAP].

Archive Submission

Archive submission requests are defined as follows:

```
ArchiveSubmissionReq ::= SEQUENCE
{
    version         TAPVersion DEFAULT v1,
    submitterName   GeneralName,
    policy          OBJECT IDENTIFIER
                        OPTIONAL,
    archiveControls [0] ArchiveControls
                        OPTIONAL,
    archivedData      ArchivedData
}
```

Archive Data

Archived data, i.e. data submitted to a TAA for preservation, has the following format.

```
ArchivedData ::= SEQUENCE
{
    type    ArchivedDataType  OPTIONAL,
    data    OCTET STRING
}
ArchivedDataType ::= CHOICE
{
    oid      OBJECT IDENTIFIER,
    mimeType UTF8String
}
```

Archive Submission Response

Archive submission responses are defined as follows:

```
ArchiveSubOrDelResp ::= SEQUENCE
{
    version          TAPVersion DEFAULT v1,
    status           ArchiveStatus,
    archiveToken     ArchiveToken
```

```
                        OPTIONAL,
    archiveControls   [0] ArchiveControls
                        OPTIONAL
}
```

Archive Token

Archive tokens have the following format.

```
ArchiveToken ::= ContentInfo
-- content type: id-tap-archiveToken
-- content: ArchiveTokenData
ArchiveTokenData ::= SEQUENCE
{
    submitterName     GeneralName,
    timestamp         TimeStampToken,
    curTime           GeneralizedTime,
    trackingInfo      TrackingInfos
                        OPTIONAL
}
```

The archiveControls field can be used to return information associated with a control included in the request, for example, the outcome of server-side validation or a nonce from the request. TAAs must not include controls in a response that are not associated with controls in a request. Submission clients should be able to process controls in accordance with the control definition.

Archive Record

The archive record contains a nested structure with the complete refresh history for the archived data. TAAs should store all cryptographic information necessary to verify each layer of the archive record in the certificates, CRLs and unsignedAttrs fields of the timestamp token, i.e. each timestamp token in the history should be self-contained for validation purposes under protection of the next layer in the archive record. A CryptoInfos unsignedAttrs field may be used to convey OCSP responses and/or trust anchor information. Archive record has the following format:

```
ArchiveRecord ::= ContentInfo
-- content type: id-tap-archiveRecordData
-- content: ArchiveRecordData


ArchiveRecordData ::= SEQUENCE
{
    timestampedData   TimeStampedData,
    timestamp         TimeStampToken
}
TimeStampedData ::= SEQUENCE
{
    prevArchRecord ContentInfo,
    messageImprint MessageImprint
}
```

The cryptoInfos field contains additional information that may be useful when verifying the archived data.

Archive Package

Archive packages are defined as follows:

```
ArchivePackage ::= SEQUENCE
{
    archiveToken    ArchiveToken,
    packageData     [0] ArchivePackageData
                        OPTIONAL,
    pollReference   [1] OCTET STRING
                        OPTIONAL
}
ArchivePackageData ::= SEQUENCE
{
    digestAlgs  DigestAlgorithmIdentifiers,
    policy      OBJECT IDENTIFIER
                    OPTIONAL,
    archRecord  ArchiveRecord,
    cryptoInfos [0] CryptoInfos
                    OPTIONAL,
    archivedData    ArchivedData
}
```

4. Security Considerations

This section provides an overview of a security analysis of the protocol.

Trust Anchors for Timestamp and Other Signature Verification on Archive Retrieval

TAAs can provide all or some of the trust anchors upon retrieval. These may include all the trust anchors required to verify the various timestamps in the archive record and/or all the trust anchors known to the TAA at the time of the archive submission (i.e., the timestamp on the archived data). The latter set of trust anchors may be useful in digital signature verification on the archived data, if the data was signed.

Trust anchors provided by the TAA upon archive retrieval are transmitted securely since they are included in the signed envelope of the retrieval response. The relying party (i.e., the retrieval client) must use a trust anchor it trusts independent of the trust anchors provided by the TAA to verify the TAA signature on the retrieval response.

The relying party (i.e., the retrieval client) can trust the TAA provided trust anchors or can ignore them. In the latter case, only the TSA (and not the TAA) needs to be trusted for the integrity of the archived data. In other words, the relying party will be able to detect the modifications made to the archived data by the TAA. Refreshing the timestamp on the archived data before the latest (i.e., most current or outermost) timestamp expires ensures this.

Algorithm and Technology Advances

In order to protect against algorithm (i.e., hashing and digital signature) compromise and/or computing technology advances, timestamps are periodically refreshed. For each timestamp token refresh, the

archived data is hashed using a current, secure hashing algorithm and a timestamp token generated using a current, secure digital signature algorithm.

Security of TSA

TAAs must be able to obtain a trusted timestamp (either by implementing timestamp functionality or by access to a timestamp service). Timestamp-related security considerations apply (see [TSP]).

ArchiveControls

ArchiveControls are optional request components that request server-side processing in addition to archiving, i.e. collection of certificates and CRLs. ArchiveControls that request alteration of the submitted data should define a response such that the timestamp contained in the archive token can be verified.

5. System Description

A trusted archive system using the requirements, concepts and protocol presented here has been developed to successfully demonstrate the concepts presented in this paper.

The components of the system are as follows

- A [TSP]-compliant Time Stamping Authority (TSA)
- A [TAP]-compliant TAA
- [TAP]-compliant TAA clients

The following diagram depicts the overall architecture of the implemented system.

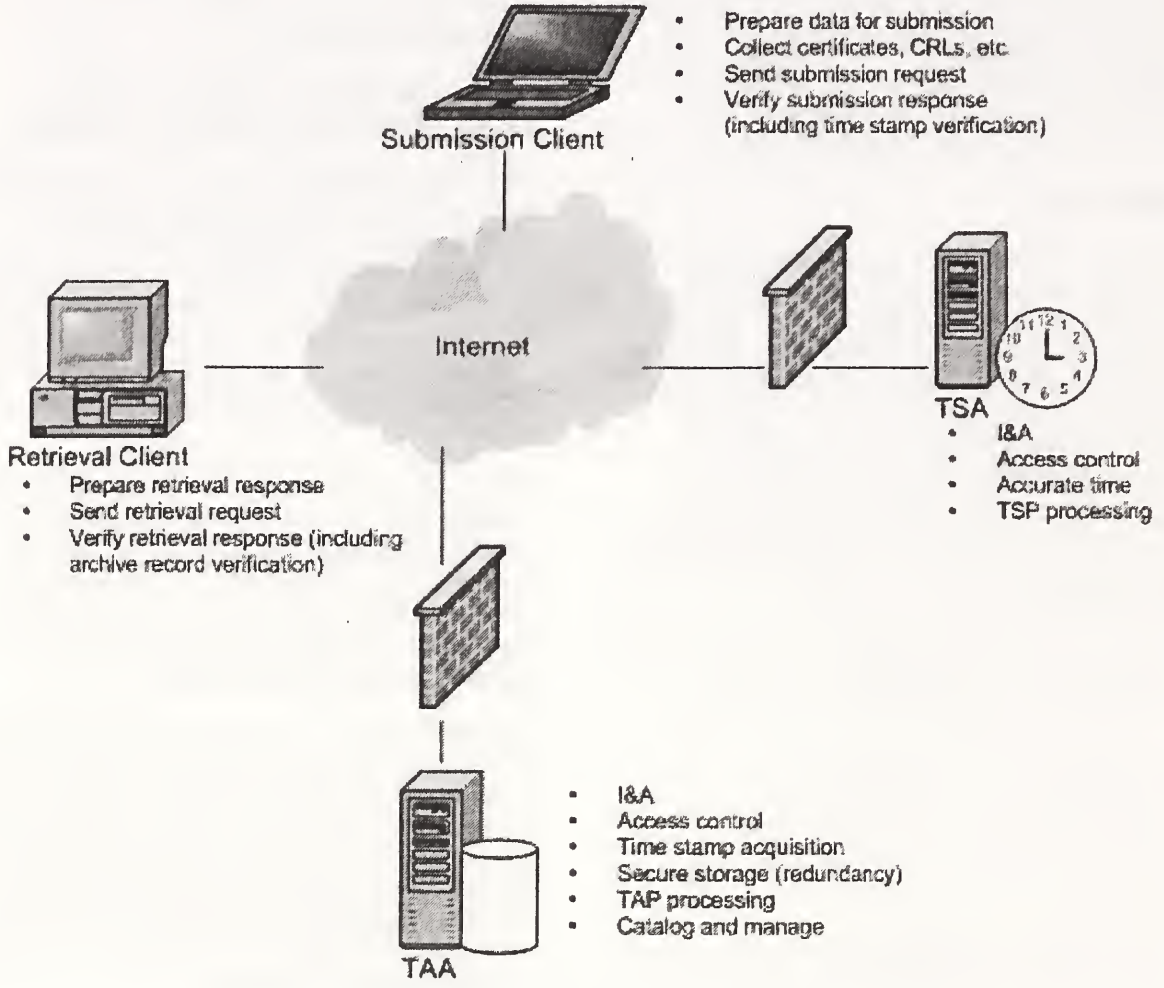


Figure 3 System Architecture

5.1 Time Stamp Authority (TSA)

The TSA is RFC 3161[TSP] compliant and is hosted on a PC running Windows 2000 Server. The TSA uses the National Institute of Standards and Technology (NIST) Internet based Network Time Protocol (NTP) service to set the Windows 2000 TSA Server system clock. The TSA interacts with TSA clients using HTTP. The TSA has a public key certificate issued by a PKI recognized by the TAA and, optionally, TAA clients.

5.2 Trusted Archive Authority (TAA)

The TAA is TAP I-D compliant and is hosted on a PC running Windows 2000 Server. The TAA has a public key certificate issued by the PKI recognized by the TAA clients. The TAA interacts with TAA clients using HTTP.

The TAA obtains the initial time stamp from the TSA upon submission of archived data. The TAA periodically refreshes the time stamps in accordance with the system concept and the TAP protocol.

The TAA catalogs archive data using the following attributes: submitter DN, time stamp token, submission date and time.

5.3 TAA Submission Client

The submission client operates on Windows workstations. The client validates the TAA signature in accordance with TAP and the time stamp token contained in the archive token in accordance with RFC 3161[TSP].

5.4 TAA Retrieval Client

The retrieval client operates on Windows workstations. The client provides an archive token to the TAA in order to retrieve the archived data. The client validates the TAA signature in accordance with TAP and the timestamp tokens contained in the archive record in accordance with TAP and RFC 3161[TSP].

A retrieval client unwinds the nested CMS package consisting of multiple nested time stamps. The retrieval client verifies the various time stamps as the CMS package is unwound using the time from the adjacent outer layer as the time of verification. The outermost layer is verified using the current time. The client determines the unwinding is complete when the innermost TSTInfo is reached. The innermost TSTInfo is used to verify the archived data.

The retrieval client may use trust anchors provided by the TAA during archive record verification or trust anchors available locally.

5.5 Operational Considerations

The system described in this section is a proof of concept implementation. If this were an operational system additional security measures are recommended akin to the operations of a CA. It is desirable that the servers and workstation use FIPS 140-2 validated hardware cryptographic modules and Common Criteria validated operating systems and application software. The operational systems and services should use Physical, Procedural, and Personnel (P^3) security controls commensurate with the security needs and perceived risks.

In addition to the computer security controls described for the TSA and TAA in System Description, appropriate boundary control product (e.g., validated to conform to [FWPP]) should be used to protect the TSA and TAA.

To enhance the security of the trusted archive service using the principle of separation of duties, consideration should be given where one TAA archives the data while another TAA trust anchors relevant to the verification of signatures on the archived data. Timestamps could be obtained from multiple TSAs to limit the damage resulting from TSA compromise. Redundant storage mechanisms should be employed to ensure that no archive data is lost due to device failure or catastrophe.

6. Lessons Learned

6.1 Metadata

One significant piece of information not included in the TAP protocol was filename and format of the archived data. This information is essential when working with material retrieved from an archive. Future versions of the protocol will include means of including a variety of metadata with an archive submission.

Metadata may also be useful in aggregating archived data over time. For example, to associate a refutation of a document with the original archived document or to associate data related by context, such as a various pieces of data in a criminal file.

6.2 Timestamp reliance

The archive record structure defined in TAP relies heavily on timestamps as defined in TSP. This has a number of potentially undesirable properties including:

- A new digital signature as part of the preservation of integrity of a digital signature
- A great degree of trust is invested in the TSA
- A one-to-one ratio of timestamps to documents to archived data objects makes development of high-performance applications difficult

Alternative timestamp structures have been defined that address these concerns by relying on the security of hash algorithms and the availability of published information, see [HOWTO] and [EFF].

6.3 Search features are important

TAP featured limited means for searching an archive. The retrieval interface was highly driven by hashes of archived data. While this works well if the data is stored with its archive token, such storage may not be the norm. Without the archive token, the effectiveness of a search is highly correlated with the submission volume of the original submitter. Search features should include means of searching based on content, metadata and/or keywords.

6.4 Auto-deletion

[TAP] defined no means for clients to define the period of time a TAA should preserve a data object. This leaves the burden on the submitter to stop the refresh process at some point in time. While this could be negotiated using the policy field, a better solution would be to provide a means for specifying the archivation period at submission time.

This leads to a need to manage the archivation period (or meta-data) post-submission. A better approach to the protocol may have been to specify a submission request, a management request and a single response type. The management request would be used to retrieve and delete archived data as well as to update meta-data, archivation period, etc. A single response format would simplify the handling of errors that are not request-specific ([TAP] defined two response types).

7. Future Directions

The LTANS WG has become quite active and is developing three standards in the area of trusted archive:

- Trusted Archive Requirements
- Evidence Record Syntax (based on [ATS])
- Trusted Archive Protocol (based on [TAP])

Another area of research is authentication and authorization for deletion and retrieval. The challenge of authentication and authorization validation in support of long-term non-repudiation can be summarized as follows:

- The identity of authorized parties may change over time. Generally, [TAP] was intended to support claims against data that occur within the memory of a person or institution where retrieval would be performed by the original submitter or by an authorized agent of an orgranization.
- The definition of authorization attributes may change over time and the naming of attributes may not prove to be unique in contexts that expand over time.
- The authorities such as CA or Attribute Authority (AA) may be no longer in existence.

Data formats, or data format migration, are another area of concern for long term archives. The formats of signed documents today may not be readily usable after a number of years.

Providing confirmation that a specific person generated a data item after a very long period of time is a very difficult problem. Over great periods of time it would be difficult to state with confidence that a particular signature was generated by a particular person. One approach may be to archive the information collected by a CA/TA to establish the binding between a person and a key. The need for this sort of demonstration may be very small. Notarization may be of assistance in this area. Rather than maintain evidence to demonstrate the binding of keys to members of the general population, it may be necessary to simply maintain evidence that binds keys to notaries, who generate attestations at a time when sufficient information is available to confirm the binding of a person to a key and a key to a signature. Biometric information provides another alternative for establishing a link between a specific individual and an archive record and/or an individual's key. This is an area that requires further consideration.

Other mechanisms for providing TAA functionality such as n of m splitting based on Shamir technique [SHA] that would provide a high degree of availability and integrity.

While these problems exist today in general, they are more likely to be encountered when one looks at 20 to 50 years and beyond. Solutions to these issues are a fertile area of research.

References

[ACTS] NIST Automated Computer Time Service (ACTS), http://tf.nist.gov/service/acts.htm

[ATS] Brandner, R., Gondrom, T., Pordesch, U. and M. Tieleman, "Archive Time-Stamps Syntax", draft-brandner-etal-ats-00.txt, July 2003.

[CMS] Housley, R., "Cryptographic Message Syntax", RFC 3369, August 2002.

[EFF] Bayer, D., Haber, S. and W.S. Stornetta, "Improving the Efficiency and Reliability of Digital Time-Stamping", *Sequences II: Methods in Communication, and Computer Science*, pp.329-334, March 1992.

[FWPP] U.S. Government Firewall Protection Profile for Medium Robustness Environments, Version 1.0, October 28, 2003.

[HOWTO] Haber, S. and W. S. Stornetta, "How to Time-Stamp a Digital Document", *Journal of Cryptology, Vol. 3, No. 2*, pp. 99-111, 1991.

[LTES] Pinkas, D., Ross, J., and N. Pope, "Electronic Signature Formats for long term electronic signatures", RFC 3126, September 2001.

[NTP] Network Time Protocol Specification, Implementation and Analysis, Internet RFC 1305, March 1992.

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

[SCVP] Malpani, A., Housley, R. and T. Freeman, "Simple Certificate Validation Protocol (SCVP)", draft-ietf-pkix-scvp-13.txt, October 2003.

[SHA] Shamir, A., "How to Share a Secret", Communications of the ACM 24 (1979), n. 11, (1979), 612-613.

[TAP] Chokhani, S. and C. Wallace, "Trusted Archive Protocol", draft-ietf-pkix-tap-00.txt, February 2003.

[TSP] Adams, C., Cain, P., Pinkas, D. and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.

PKI: Ten Years Later

Carlisle Adams
University of Ottawa
Ottawa, Canada
cadams@site.uottawa.ca

Mike Just
Treasury Board of Canada, Secretariat
Ottawa, Canada
just.mike@tbs-sct.gc.ca

Abstract

In this paper, we examine the history and evolution of so-called Public Key Infrastructure (PKI). We compare the original definition of PKI with a broader and more flexible definition that better reflects the variety of implementation philosophies available today. This current definition shows how the understanding of this technology has matured (although its essential characteristics have remained unchanged) and is derived, at least in part, from an evaluation and comparison of several quite different forms of PKI as well as a consideration of PKI criticisms over the years. The original definition of PKI may be dead or dying, but PKI technology continues to thrive as an extremely useful (and, in some cases, necessary) authentication solution.

1 Introduction

The technology known as "PKI" has been simultaneously maligned and praised. PKI praise can come in two flavours. The first results from a dislike for other security technologies. For example, a dislike for password-based authentication may result in a stronger preference for PKI solutions. Secondly, public-key technology offers some important benefits that are not similarly offered by other technologies, such as digital signatures. However, PKI is equally, if not more often, criticized. Difficulties around issues such as application integration, interoperability, and trust often lead critics to predict the end of PKI. While these shortcomings are very real, other issues have often been raised that either are orthogonal to PKI, or similarly impact non-public-key-based technology. In this paper, we try to identify such issues so that their true impact on PKI can be understood.

We attempt to provide some clarity to the status of PKI by discussing how it is understood today, compared with how it was initially defined. We examine the ways in which this updated definition encompasses ten years of PKI evolution, while remaining true to the original, essential characteristics of this technology. In order to do this, we compare several different PKI implementation models and see what lessons can be learned from some previous criticisms of PKI.

In Section 2, review and highlight several concepts related to public key technology and introduce six components that will contribute to our definition of a PKI. Section 3 reviews four PKI examples relative to these PKI components. In Section 4, we review and critique some well-known criticisms of PKI. Section 5 builds upon the previous PKI components, examples and critique to provide a modern definition for a PKI, while Section 6 recognizes those areas that still require development in order to support more successful PKI deployment.

2 Public Key Technology

In this section, we briefly examine some of the cryptographic properties of a PKI, and proceed to discuss how these properties may be used in practice.

2.1 Public Key Cryptography

Public key (a.k.a. "two key" or "asymmetric") cryptography was invented by Diffie and Hellman in 1976 [DH76]. Unlike secret key (a.k.a. "symmetric") cryptography, in which the same key K is shared between parties A and B, pairs of corresponding private and public keys for each user allow the unique realization of some operations. Specifically, let the respective private and public keys, $priv_A$ and pub_A, belong to party A. By operating on data with $priv_A$, A can *digitally sign* data that is verifiable by party B (or any other party) operating on the signed data with pub_A. Equivalently, party B (or any other party) can encrypt data using pub_A, where the encrypted data can only be decrypted with $priv_A$.

The true power of public key cryptography lies in the possession of a private key, *uniquely*, by each party. The "demonstration of knowledge" of the private key by operating on data with said key, provides a powerful tool that distinguishes asymmetric cryptography from its secret key counterpart.

2.2 Public Key Cryptography in Practice

Most secure transfers of data involve an exchange between identifiable parties. On its own, public key cryptography only supports asymmetric, mathematical operations on data; it does not by itself provide a connection to applications or environments such as e-commerce, e-mail, or the Web.

To provide such a connection, several additional pieces are necessary. These additional pieces form the definition of a PKI – an "infrastructure" that makes public key technology available to the applications and environments that wish to use it. In subsection 2.3 below, we identify several components that are integral to an infrastructure for supporting public key cryptography (these components are used to capture the evolving definition of a PKI in Section 5).

Identification is a property that is particularly critical to a PKI, and (at least historically) a strong differentiator between some different PKIs. Specifically, public key cryptography is made considerably more useful if the public key is *bound* to a so-called *identifier*. As distinguished below, this *identifier* may or may not provide direct information regarding an actual *identity*. This identifier may be an

- *Anonym*: "No name"; a single-use identifier providing no information as to the identity of the key owner.
- *Pseudonym*: "False name"; providing a "pretend" identity that can be used over a period of time to protect the real identity of the key owner.
- *Veronym*: "True name"; providing the identity of the key owner.

The identifier is typically meaningful only within a specific context or environment; it may or may not need to be globally unique, depending upon the applications for which it will be used.

Parties that use public key cryptography for encrypting data, or for verifying digitally signed data, will rely on the binding of an identifier to the public key (whether this binding is preserved in a certificate or database, for example) in order to associate that key with an entity with which they may have past or future interactions. This also supports repeated use of the same public key, whether or not the key is directly associated with an actual identity.

2.3 Public Key Infrastructure

Approximately ten years ago, the 1993 version of the ISO/IEC CCITT/ITU-T International Standard X.509 began to be disseminated, recognized, and implemented in small-scale environments. Late 1993 and early 1994 was effectively the beginning of PKI (although that

acronym had yet to be coined) because that version of the X.509 standard – more than the 1988 version – fleshed out some of the important details of certificates, certification authorities, and related concepts.[1]

In those early days, a "PKI" was defined fairly rigidly, although with hindsight we can identify six major components to the definition that are still critical today, three to do with the *validity* of bindings (authority[2] functions), and three to do with the *use* of bindings (client functions). With respect to the validity of the binding between a public key and an identifier, what is needed is

1. an *authority* whose responsibility it is to create and destroy these bindings, as required, or aid in related authoritative actions,
2. an *issuance process* for expressing these bindings in a way that can be understood by other parties (i.e., in an agreed syntax) and for making this information available to parties that wish to know it, and
3. a *termination process* for breaking bindings when necessary and making this information available to parties that need to know it.

With respect to the use of such bindings, what is needed is

4. an *anchor management process* for augmenting or diminishing the set of authority public keys that will serve as roots or trust anchors for the client[3],
5. a *private key management process* for ensuring that a client private key can be used for its desired purpose (this can include key pair generation and update, registering and binding an identifier to the corresponding public key, proper protection of the private key while it is valid, and backup & recovery of the private key in case of loss), and
6. a *binding validation process* for determining when the client should trust that a given public key (retrieved or acquired from some external entity) is authentically associated with a given identifier.

In Section 5, we develop a more detailed definition of a PKI, based on these components, that reflects the decade-long evolution of a PKI. The degree to which these components are implemented is commonly a risk management decision. The PKI examples in the next section can differ based upon such choices.

In the original Diffie and Hellman model [DH76], public keys would be retrieved from a secured repository. The security of this repository served to bind the public key to other attributes of the key owner. In support of offline binding production and distribution, Kohnfelder introduced the notion of a certificate [Kohn78], whereby a public key and an identifier (e.g., a name) were placed in a data structure signed by a Certification Authority (CA) and made available in an unsecured repository.

Various PKI systems can be distinguished and compared based upon the above six PKI characteristics. In Section 3, we categorize several examples of PKI systems with respect to these characteristics. It is particularly interesting to note that one of these examples makes use of

---

[1] Though the first version of Pretty Good Privacy (PGP) appeared in 1991, it wasn't until later that features consistent with a PKI were provided (see Section 3.2).

[2] An "authority" may be any specially designated entity, though an end-entity client may also act authoritatively.

[3] These roots form the axiomatic elements of direct trust for a client. Trust in other public keys is derived from these roots.

Diffie and Hellman's original concept of a secured repository (AADS; see Section 3.3), while the remaining examples use "certificates" with varying syntax.

3 PKI Examples

Over the past 10-15 years, there have been several examples of public-key technology solutions. Below, we focus on those solutions that offer the best contrasts within the PKI components identified above. The representative example names (X.509, PGP, AADS/X9.59, SPKI) are quite overloaded with varying descriptions, as they may refer to several standards or even several varying implementations of those standards. In our review below, we have tried to focus on those features that are independent of specific product implementations yet representative of distinctive features for each PKI example.

On a related note, it is also recognized that over such a time period, the solutions have each grown and matured greatly. Though we attempt to identify this growth, our main purpose is to identify the philosophical differences between the solutions, so that not all features of each PKI solution may be acknowledged.

3.1 X.509

The X.509 standard [X509-00] and its Internet profile [RFC3280] do well to represent the PKI components identified in the previous section. In most cases, implementations differ based upon the rigour with which they implement the suite of appropriate standards (e.g., see the exhaustive list of Internet standards for X.509 [PKIX-WG]). Below, we examine relevant components of an X.509 PKI.

- *Authority.* A Certification Authority (CA) issues X.509 certificates that bind the public key to a Distinguished Name (DN) identifier (although other name forms are also allowed), in addition to other information contained in the certificate. An Attribute Authority (AA) is similarly defined, and binds more general attributes to one another in an attribute certificate, and provides an optional link to a corresponding public key certificate.
- *Issuance process.* Typically, though not necessarily, certificate issuance involves a Registration Authority (RA) responsible for registering the user (including their identification, if performed). Traditionally, the DN and alternative name forms would be veronymous. However, neither the ASN.1 syntax nor the standard restricts this so that anonymous or pseudonymous name forms are fully supported.[4] Once issued by a CA, certificates require no further integrity protection and may be distributed amongst parties or made available in a repository. This repository is commonly an X.500 or LDAP directory, though various other repositories are typically supported now, including Web servers. Retrieval is predicated upon knowing the identifier of the certificate holder (typically the DN, although an email address contained in the certificate can also be used).
- *Termination process.* Certificates contain an expiry date that acts as a default termination date for the certificate. Certificates may also be "revoked" prior to their expiry, in which case the revocation information must be disseminated. There are traditionally two ways for this to be achieved: (i) by posting information regarding the revocation in a Certificate Revocation List (CRL), or (ii) by making the revocation information available through an Online Certificate Status Protocol (OCSP) responder [RFC2560]. The location of revocation information is typically included within the certificate that is being verified (e.g., as a URL for the CRL).

[4] See [Just03] for an example of an X.509 PKI with a pseudonymous certificate identifier.

- *Anchor management.* The standards describe protocols for retrieving trust anchors as part of the registration (and key update) process(es). Depending upon the implementation, a client may be able to trust a number of trust anchors simultaneously (as part of a certificate trust list). Traditionally, there are two forms of trust for X.509 certificates. In the first, the application software holds the public key of a root CA. All certificates that may be trusted by this client are issued, either directly or indirectly (e.g., within a hierarchy), by this CA. In the second form of trust, a party holds the public key of the CA that issued their own certificate.
- *Private key management.* The standards support protocols for renewing key material prior to the expiry of the corresponding certificate. They also support the backup of decryption keys (and, more importantly, their recovery). The standards also allow a separate lifetime for the private key itself, primarily in support of preventing the creation of signatures too close to the time of certificate expiry, though this lifetime value is also helpful to trigger a timely key update in support of uninterrupted client operation.
- *Binding validation.* Clients use their trust anchors and possibly chain building to establish certificate trust. Trust in certificates issued by other CAs may be obtained through cross-certification between the CAs, or possibly by the party importing or retrieving the certificates of the other CAs as necessary. There are numerous variations to these two simple trust models. Traditionally, clients would be required to retrieve and validate the entire chain of certificates, though recent standards have been developed to support the off-loading of some of these operations [RFC3379].

In the often-cited "browser-based PKI", it is important to recognize that certificates are issued to servers, while clients use those certificates to authenticate a server and establish a confidential communication channel.[5] Clients retrieve the server's certificate a part of the SSL protocol [Resc01]. The termination process supports expiry, though automated revocation support is minimal and inconsistent. Client anchor management is essentially static, and established by the version of Web browser being used, though users can manually update their trust anchors, if they so desire.

3.2 PGP

Though the first version of Pretty Good Privacy (PGP) appeared in 1991, its primary focus was in the support of public key cryptographic operations, not the provision of a PKI. Later versions supported notions such as key servers (see, for example, [PGPks]) thereby supporting an "infrastructure" for the management of public key information. In more recent times, PGP has highlighted its ability to also support features similar to X.509 [PGP99]. Traditionally, however, PGP has been distinguished by its distributed approach to key management. PGP certificates are formed of one or more "certifications", which bind keys to user information with a digital signature. There is great flexibility in the key and user information that can be conveyed in a single certificate.

- *Authority.* Traditionally, a PGP PKI avoided the need for any form of authority. As discussed below, trust relies upon a "web" of users. However, the syntax does not preclude "authoritative users" that might act in a similar fashion to a Certification Authority (CA). For many communities, the lack of an authority greatly eases initial deployment as small communities need only rely upon the bilateral sharing of certificates among users who wish to communicate securely.
- *Issuance process.* Certificates are created and populated by the key owner. They can be distributed via email, Web pages, key servers, and/or other means. The identifier is typically an email address, though there is nothing to preclude other identifiers.

[5] Though client authentication with certificates is supported by the standards, it is not often implemented.

- *Termination process.* Certificates can contain an expiry date (though no expiry date need be set) that acts as a default termination date for the certificate. The distribution of certifications is not managed, so that manual revocation would have to be performed, though revocation information can be made available in a similar fashion to publication of the certificate.
- *Anchor management.* Trust must be anchored in the *direct trust* of other users' certificates. Various mechanisms can be used to establish this direct trust base. For example, two users can exchange key information by email, but verify the authenticity of the exchange by exchanging message digests for the key information by phone. Such key information serves the role of PGP "roots" or "trust anchors."
- *Private key management.* Lacking a 3rd-party authority, private key management is the responsibility of the key owner. For example, key owners can backup private keys on a separate disk. It would be a simple task for software to remind users of the need to update their key material. Similar ease would allow an update of key material, since no communication with an authority is required, though updated key distribution would still be performed by the key owner.
- *Binding validation.* Based upon some initial, direct trust, there are a couple of options for indirectly extending trust to others. With hierarchical trust ("chain of trust"), you trust others that are trusted by people you trust. With cumulative trust ("web of trust"), you trust others only when they are trusted by a number of people you trust.

3.3 AADS/ ANSI X9.59

ANSI X9.59[6] is a financial industry standard for secure financial payments, based on AADS (Account Authority Digital Signature) [AADS99]. For our purposes, we use it as an example of a non-certificate-based public key solution.

A public key is stored and managed as part of a key owner's financial account, along with other identity or authorization information. So, the issuer is an authority that already has a relationship with the user, and thus should be able to easily identify said user.

- *Authority.* The authority, or maintainer of the binding, is the user account manager. The public key serves, quite simply, as an additional attribute to a user's financial account record.
- *Issuance process.* Users may be identified by their account manager, based upon shared financial secrets, or other account information. The public key is retained by the manager. For AADS, the public key need only be accessed by an authentic request and response with the account manager to retrieve the public key for signature verification. For other applications, this could be easily adapted to allow for public-key encryption. Note that by not relying on a certificate, the AADS solution is more similar to the ideas of Diffie and Hellman than those of Khonfelder (see Section 2.3), except that with AADS the repository of public keys is built, held, and used by the relying party alone, whereas with the original Diffie-Hellman proposal this repository was to be created for the use of the whole world.
- *Termination process.* There are no "certificates" and use of any public key is always initiated by a request to the account manager. Expiry or revocation occurs by removing or replacing the public key for a user's account. Therefore, a type of immediate, online validation is supported as part of the public key retrieval.
- *Anchor management.* Relying parties require a method by which they can trust the account manager. A method similar to server-authenticated SSL would suffice. An initial trust anchor could be retrieved when the user's key pair is generated, for example.
- *Private key management.* Updates to private key material may be managed and initiated by the account manager. In the case of AADS, since only digital signature operations are

[6] See http://www.ansi.org/

performed, there is no need to backup private key material. If encryption operations were supported with the user public keys, there may be a need for key backup support.
- *Binding validation.* Trust is isolated to the domain of a single account manager. As mentioned above, online trust validation is implicit with the retrieval of the public key.

Similar to SSL representing a client-server PKI implementation using X.509 certificates, SSH [SSH03] is representative of a certificate-less client-server PKI implementation. As part of the SSH transport protocol, clients establish trust in the server based on their trust in the server host key. Though there are options for how this trust might be established, including the client maintaining a store of trusted server host keys, or certification by a CA, SSH presents an interesting compromise whereby "the server name - host key association is not checked when connecting to the host for the first time" [SSH03]. Though introducing the potential for a middle-person attack, this novel variation offers great improvement for SSH bootstrapping. Once a server-authenticated, confidential channel is established, the client may authenticate to the server; this is often performed using password authentication.

3.4 SPKI

Simple Public Key Infrastructure [RFC2692, RFC2693] was developed in response to several criticisms of X.509. The major philosophical objection to X.509 surrounds its relation to X.500 naming. SPKI, more correctly an authorization infrastructure, relies upon the uniqueness of the combination of a pseudonym and a public key.

- *Authority.* SPKI focuses on the issuance of authorization information within certificates. Thus, an SPKI authority might be referred to as an authorization authority. With regard to an issuance authority, SPKI theory indicates that certificates may be generated "by any keyholder empowered to grant or delegate the authorization in questions." [RFC2692]
- *Issuance process.* In support of the authorization information, the SPKI certificate syntax uses an S-Expression, which is a LISP-like expression using parentheses. *Authorization certificates* bind authorization information to a key, while *attribute certificates* bind an authorization to a name. The use of names differs from the initial use of global names for an X.500 directory, as part of X.509, and was inspired by the use of SDSI's [SDSI96] local names. Combined with the (globally unique) hash of a public key, such a name can become globally unique.
- *Termination process.* Certificate lifetime is parameterized by a validity period so that certificates can be set to expire. Several options for certificate revocation are supported, including Certificate Revocation Lists (though they are not the preferred choice). Options for online revocation status checking are also supported. Preference is given to "positive statements" on certificate validity, so that a protocol returning an indication that a certificate is currently valid is favourable to one that returns a notice of invalidity.
- *Anchor management.* Details regarding anchor management are left open for developers so that, for example, protocols similar to those previously described could be used. For validation of authorization information, however, the relying party maintains an access control list (ACL).
- *Private key management.* The management of private keys depends upon the certificate issuer regarding issues of key backup; however, when used only for authorization purposes, the need for key backup is limited. Support for key updates does not appear to be standardized, so would be dependent upon the specifics of a particular implementation.
- *Binding validation.* The main difference with traditional X.509 is the use of the pseudonym for SPKI. Processing decisions for SPKI certificates are defined through "tuple reduction." [RFC2693].

3.5 Summary of Examples

The following table compares the solutions based upon the *validity of bindings* (see Section 2.3).

PKI Solution	Authority	Issuance Process	Termination Process
X.509	Certification Authority (CA) Attribute Authority (AA). The CA is the owner / definer of the namespace for the identifier.	ASN.1 syntax Traditionally available from X.500 or LDAP directories.	Certificate contains an expiry date. Revocations posted through revocation lists, or made available through an OCSP responder.
PGP	No external authority required. Key pair and certificate are self-generated. The user (end entity) is the owner / definer of the namespace for his/her identifier.	Made available to others by key owner (e.g. via Web page, email signature, or key server).	Certificates can expire. Termination performed by key owner. Dissemination of termination notice by key owner as with certificate publication.
AADS/ X9.59	User account manager. The relying party (the account manager) is the owner / definer of the namespace for the identifier (the acc't. #).	Public keys available in secured repository from account manager.	Public keys removed from repository when binding is terminated.
SPKI	No explicit authority is required as the authorization granter or delegator may issue certificates. The relying party is the owner / definer of the namespace for the identifier.	Issue authorizations based on pseudonymous identifier or SDSI names.	Similar to X.509, though "positive statements" through online validation are preferred.

The following table compares the solutions based upon the *use of bindings* (see Section 2.3).

PKI Solution	Anchor Management Process	Private Key Management Process	Binding Validation Process
X.509	Single or multiple roots. Standardized protocols support changes.	Standardized protocols support update, backup and recovery.	Client search of Directory for cross certificates. Delegated path discovery and validation services are being standardized.
PGP	Direct trust of other user certificates. Trust anchor is user's own key(s).	Manual update, backup, and recovery performed by user.	Chain of trust, or web of trust.
AADS/ X9.59	Trust in account manager is required.	Depends upon expiry policy. Backup and recovery not a concern when only digital signatures are used.	Only direct validation through trusted key retrieval.
SPKI	Open to developer. Trust anchor is the ACL at the relying party.	Open to developer. Fewer backup and recovery requirements when certificates used only for authentication or authorization.	Tuple reduction.

These fundamental PKI examples contribute to a greater understanding of the different options available for PKIs within what is mistakenly viewed as a "rigid" structure. In the following section, we further examine criticism of PKI, identifying those issues that are specific to the components of this infrastructure. In Section 5, we use the examples and the lessons learned from the criticism to capture the evolutionary definition of PKI.

4 Criticism of PKI

Over the past ten years, PKI has been the subject of criticism from various quarters. Some of this criticism has been beneficial, driving the evolution of this technology and leading to a deeper understanding and broader application of PKI. However, much of the criticism has been misdirected, aimed at PKI when the actual problem or challenge is either independent of this technology, or common to many technologies.

In this section we review some popular PKI criticisms to see which can fairly be applied to the current state of the art. While it is certainly not the only collection of criticisms, arguably the best known collection can be found in the paper by Ellison and Schneier [ElSc00]. We therefore use that paper as the basis for our examination of PKI, circa 2004.

"Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure" aims to explore some basic questions around PKI ("What good are certificates anyway? Are they secure? For what?") so that potential users of this technology can be made aware of some of the risks involved with its use. This is unquestionably a worthy goal and will serve the industry well, but only if the highlighted risks are accurate and fair (i.e., legitimate criticisms of PKI technology). Let us examine some of the risks discussed in that paper.

Risk #1 ("Who do we trust, and for what?") warns that the certificates issued by a CA should not be automatically trusted for a plethora of application-level purposes, such as making a micropayment or signing a million-dollar purchase order ("Who gave the CA the authority to grant such authorizations? Who made it trusted?"). Unfortunately, this criticism highlights only the misuse of PKI by some implementers; it is not a valid criticism of PKI itself. PKI is an authentication technology; authorization is an independent matter and may or may not be linked to authentication in any way. The authors suggest that "Many CAs sidestep the question of having no authority to delegate authorizations by issuing ID certificates." However, the issuance of ID certificates is the primary function of a CA (not a "sidestep"). In some environments, it may be natural for information other than an identifier to be linked to the public key by the CA; for such situations a variety of authorities for such information may be used in conjunction with the CA (these are discussed as part of the evolving PKI definition in Section 5). On a related note, while certificate policies appear to contain some notion of authorization, they are more properly viewed as statements regarding the "quality" of the key. For example, given the specific process used to generate the key pair, the rigour with which identification of the key holder was done, the care with which the private key will be safeguarded, and so on, the CA declares (by including a policy to this effect in the certificate) that the public key can be used for signing million-dollar purchase orders. But this is not a granting of authority. In a properly-implemented system, the signer must still prove that s/he is authorized to sign such a purchase order (and this authorization will typically come from some entity in the environment that is not the CA). Certificate policy may be viewed as a "fit for purpose" declaration: if the signer is allowed to sign such a transaction, then this key pair can be used to create and to verify that signature.

Risk #2 ("Who is using my key?") warns that the private key stored on your computer may not be secure (without physical access controls, TEMPEST shielding, air-gap network security, video surveillance, and so on). Clearly this is true, but is equally true of all technologies that store data on a computer. In order to address this, PKI has evolved to support both "soft token" solutions (in which the user retains the private key) and roaming solutions (in which the private key may be stored at a server). As always, there are security / convenience trade-offs for each. When stored at the user's computer, the user can authenticate with his/her private key to a server that has an authentic copy of the public key. This is arguably more secure than solutions that store either a clear-text or hashed version of a password at a server in support of password-based authentication (the latter is susceptible to brute-force attack). The discussion about PKI vendors "lobbying for laws to the effect that if someone uses your private signing key, then you are not allowed to repudiate the signature" does not reflect any of the myriad debates we have heard and read on this topic. (On the contrary, if there is any reasonable evidence that someone else has used your private key, this is precisely when you can repudiate a digital signature.) In any case, in recognition of the vulnerabilities associated with typical computing platforms, PKI has come to strongly support alternative devices, such as hardware tokens and smart cards, for storing private keys and trust anchors.

Risk #3 ("How secure is the verifying computer?") examines the insecure computer question (i.e., Risk #2) again, but this time from the side of the verifying machine. As above, this risk is shared by all technologies that use computers and is not specific to PKI. Again, alternative storage devices can be helpful here.

Risk #4 ("Which John Robinson is he?") warns that the name in a certificate may not be as valuable as it appears to be ("You may know only one John Robinson personally, but how many does the CA know? ... How many John Robinsons are in the New York City phone book, much less in a hypothetical phone book for the global Internet?") Additionally, the authors ask, "How do you find out if the particular John Robinson certificate you received is your friend's

certificate?" But one could equally ask how you find out if the e-mail you received is from your friend John Robinson or from some other John Robinson, or how you find out if the person on the other end of the telephone is your friend John Robinson, or how you find out if the postcard you received in your mailbox is from your friend John Robinson. Real life requires us to be able to resolve the potential ambiguity with regard to a name, and we do this all the time, but this is not a problem that is either created, or purported to by solved, by PKI. Users in the electronic world, as in the physical world, need to be able to do the mapping between name and identity whether or not PKI was ever invented. This is true of all authentication technologies. A PKI binds an identifier to a public key. Associating that identifier with an identity, or with entitlements within the context of the application being used, is outside of the scope of PKI; it always has been. Applications that rely upon PKI for authentication need to recognize that this issue is not solved by PKI. (More discussion of this topic can be found in Chapter 14 of [AL03]).

Risk #5 ("Is the CA an authority?") warns that the CA may not be an authority on what the certificate contains (e.g., the corporate name of the keyholder and the DNS name of the server in an SSL server certificate). There are authorities on DNS name assignments, and authorities on corporate names, but the CA is likely to be neither of these. This is quite true but, as stated above, it is not necessarily the job of the CA to create names, or even assign names to entities; its primary function (and its authority) is to bind an identifier to a public key. As time has passed, it has become more generally recognized that a CA may make use of other authorities in order to do this task. In particular, it will often collaborate with other naming authorities to ensure that the information in a certificate is as accurate as possible.

Risk #6 ("Is the user part of the security design?") warns that users will often make security decisions (such as whether to shop at a given SSL-protected Web page) without even seeing the certificate involved or knowing whether it has any relation to what is displayed. This is certainly true, but is equally true of many security technologies. If a security infrastructure provides a technical means by which application security decisions can be automated and enforced, and then these means are not used, this is not the fault of the infrastructure and is not a risk of the infrastructure. More accurately, the "risk" is that security software is often not implemented correctly or used properly, but this is true of all security software, everywhere, and has nothing specific to do with PKI. However, in general, PKI implementations do need to provide for more useful interaction with the user [WhTy99].

Risk #7 ("Was it one CA or a CA plus a Registration Authority?") warns that "the RA+CA model allows some entity (the CA) that is not an authority on the contents to forge a certificate with that contents." This is true, but authorities in any system can always abuse their power. This is not a risk specific to PKI, but is true for all systems, everywhere. Furthermore, even if an RA+CA combination can be less secure than a single CA, there are certainly environments in which placing all power into the hands of a single authority can also be a highly risky thing to do. As in any system, it is important to choose the authorities carefully or the system will not work as intended. Over the years, explicit statements of CA practices and policies (see, for example, the framework specified in [RFC3647]) have come to be used throughout the PKI community so that external auditors and inspectors can check whether a given CA is abusing its power in some way.

Risk #8 ("How did the CA identify the certificate holder?") warns that the CA may not use good information to check the identity of the entity applying for the certificate, or may not ensure that this entity really controls the private key corresponding to the public key being certified. This is true, but again is not a risk specific to PKI. If authorities in any system do not do their jobs diligently and with integrity, the system cannot be expected to work. This is not a failing of the system itself. Authorities in a PKI (in particular, the CAs) need to be chosen carefully and

trained well, but this is no different from any other system. As in Risk #7 above, auditable statements of CA practices and policies can be helpful to avoid problems in this area.

Risk #9 ("How secure are the certificate practices?") warns, in a nutshell, that "Certificates must be used properly if you want security." It is hard to imagine that anyone would argue with such a statement. But passwords must be used properly if you want security; biometrics must be used properly if you want security; smart cards must be used properly if you want security; and so on. Once again, this is not a risk specific to PKI; this basic statement holds true for all security technologies.

Risk #10 ("Why are we using the CA process, anyway?") warns that PKI does not solve all security problems, even though it is sometimes marketed and sold under that premise. This is in some ways a fair criticism, as some over-zealous marketing executives have sought to increase profits by stretching the truth in several directions. However, this is not a risk of PKI. All this highlights is a need to get accurate information out regarding what PKI actually is, and what it actually does. Things have improved significantly in this area in the past few years, but more can certainly be done.

In summary, we find that of the popular PKI criticisms voiced in the literature and at various conferences, many do not apply to PKI at all, and most of the rest apply equally to all security technologies. (As a measure of items related to implementing and deploying a PKI, however, they do highlight some specific concerns. And, as with the other security technologies to which they pertain, solutions – often outside the scope of a PKI – can be applied.) For the remaining criticisms that are accurate and valid, the evolution of this technology has come to understand and address these comments so that the current (at least theoretical) view of PKI no longer appears to be deficient in these ways. Such criticisms have therefore been very beneficial to the industry as a whole.

5 PKI Evolution and a Current Definition

The comparison of approaches in Section 3 makes it clear that a PKI can be instantiated today in many different ways. But ten years ago, several of the above instantiations would not have fit into the "accepted vision" of what a PKI was. Clearly, something has changed, but is it the essence of the definition, or the implementation details? Guided by the general definition of Section 2.3, we see that the essence remains intact; only our understanding of each of the components of that definition has evolved over time.

Definition 1994. In 1994, the six components of the general definition given in Section 2.3 were restricted in the following ways.

1. *Authority.* The authority was always and only a Certification Authority (CA). There was no place in the PKI for any other kind of authority.
2. *Issuance process.* The syntax was always and only an X.509 public key certificate which binds a public key to a Distinguished Name (DN) for the user. The certificate was made available to other parties through the use of an X.500 Directory.
3. *Termination process.* The termination process was always and only a Certificate Revocation List (CRL) which could be made available to other parties through the use of an X.500 Directory (perhaps pointed at using a CRL Distribution Point in the certificate).
4. *Anchor management process.* A user may trust the CA that is "closest" to him/her (i.e., the CA that actually issued the user's certificate) or may trust the root of a hierarchy of CAs

which includes the CA that actually issued the user's certificate. In either case, however, the client code was pre-installed with a single trust anchor and no changes were possible.

5. *Private key management process.* Very little of this was specified, although it was generally assumed that key generation occurred at the CA, registration occurred via an out-of-band, in-person process, and private keys were "safe" in the user's local environment (perhaps protected by a password).
6. *Binding validation process.* Client machines had to be configured with a large, special-purpose software toolkit that could understand all the details of certificate processing and could make validated public keys available to application code.

We now propose an updated definition of PKI.

Definition 2004. By 2004, after ten years of evolution that has resulted from extensive discussion, research, and implementation by various interested parties around the world, we find that each of the above six components of the definition has broadened considerably. However, interestingly, the same six components comprise the core of the definition. That is, the essential characteristics of the definition remain unchanged, even if the thinking about how to realize these characteristics has deepened and matured over time.

1. *Authority.* The notion of an "authority" has broadened from the CA that is "closest" to a user, to a CA that may be "farther away" (e.g., at the root of the user's hierarchy), to a CA that may be even farther away (e.g., at the root of a different hierarchy in another domain), to a CA that may be entirely independent of the user (e.g., one offered as a public service). In addition, the authoritative role of a CA might be performed by an end entity. Furthermore, it is now recognized that a CA may make use of other entities prior to issuing a binding. For example, an *Identification entity* (perhaps a Registration Authority, or some other entity altogether, such as a Naming Authority) may be used to properly determine the correctness of an identifier (on behalf of, or at the request of, a CA) before the identifier is bound to the public key. As well, PKI now recognizes the utility and value of other authorities in the environment that are not CAs, such as OCSP Responders, certificate path validation authorities, Attribute Authorities, and so on.
2. *Issuance process.* A number of different syntax proposals have been discussed and implemented over the years, and it is now well recognized that some environments will be more suited to a particular syntax than others. There is therefore a need for various ways of encoding the binding expressed by an authority. Similarly, options for the type of identifier (see Section 2.2), and the actual location of the trustworthy binding, have evolved as different choices. Certificate formats such as PGP, SPKI, SAML, XKMS Responses (see Section 6), and so on, all have a place in this broader definition. Furthermore, it is recognized that X.500 Directories are but one possible mechanism for making these bindings available to other entities, and many other technologies are now commonly in use to achieve this.
3. *Termination process.* Breaking the binding between a public key and an identifier can now use many more mechanisms than the traditional CRL. Online certificate status checkers (e.g., OCSP) were an early step in this direction, but even broader online services, such as delegated path validation [RFC3379] and XKMS servers, have also been envisioned and implemented. The use of on-line checking includes the option of online certificate retrieval, where only if the certificate is available, is it considered valid at the time of retrieval. The PKI community has come to realize that the information regarding a revoked binding needs to take different forms and use different delivery mechanisms for different environments.
4. *Anchor management process.* In support of the broader definition of authority, mechanisms for establishing how different parties accept the bindings issued by an authority have been defined and used, including trust root installation, cross-certification, trust lists, and so on. It

has become recognized that the trust anchors for a user will typically be a set of size greater than one, and that the members of this set will need to change over time.

5. *Private key management process*. Thinking in this area has broadened significantly over the past ten years as the need for flexibility in different environments became clear. To list a few examples, key generation may take place at the CA, at the client, or at a third party and protocols have been developed to handle any of the three options securely (along with protocols to allow secure backup and recovery of key material). Registration might need to be an online process (rather than an offline process) for efficiency and user convenience. As well, private keys might be stored in software, in hardware tokens, in smart cards, or in other formats, and might be stored with the user or at some 3rd-party server; protocols and interfaces have been developed over the years to handle all of these options.
6. *Binding validation process*. With respect to software, there was a growing concern that large, special-purpose toolkits were not the best alternative for some environments (for a number of reasons, including cost, size requirements, and complexity of upgrades). Interest in this area shifted to so-called "thin clients": toolkits that were very small for fast and easy download (e.g., in Java applet form). But there was also a growing realization that, in some environments at least, the ideal situation would be native applications (e.g., off-the-shelf browsers) that could properly understand all the details of certificate processing.

The current view of PKI, as expressed in "Definition 2004", is a reflection of the evolution that has occurred in this community over the past ten years. It benefits from the innovative thinking and fruitful technical discussion of researchers the world over, and has been steered greatly in more practical and useful directions by constructive criticism and numerous implementation efforts (see Sections 3 and 4 above). This definition, we believe, represents the "state of the art" in the understanding of PKI.

6 Moving From Theory to Practice

Reflecting – in an updated definition – the evolution (and the occasional revolution!) that has occurred over the years may be a useful step, but it is not sufficient. Clearly, this deeper understanding of PKI needs to be embraced in a real way in real implementations. This is not to suggest that a given PKI implementation should strive to be all things to all people. If we as a community have learned anything over the past decade, it is that the many options available for each component of the definition preclude any "one size fits all" PKI. However, even for a given set of choices, most (perhaps all) implementations can improve in both correct operation and suitability to a given environment. Many common implementation bugs and challenges have been summarized well by Gutmann [Gut, Gut02]. Specifically, Gutmann identifies issues regarding hierarchical naming, revocation, and certificate chain building. Current and prospective implementers of PKI technology would do well to look through some of this material.

One important realization of Gutmann is that original (and even some current) PKI implementations would "constrain the real world to match the PKI", as opposed to "adapt[ing] the PKI design to the real world." [Gut02]. It is hoped that we similarly capture this concern in our discussions above. In considering further issues that may remain regarding the deployment of PKI within some environments, a survey was recently performed by the OASIS PKI Technical Committee [OAS03]. The main impediments cited were the cost and lack of PKI support within client applications. Noteworthy in this regard are more recent PKI-related efforts that were motivated to address this specific concern. In particular, XML Key Management Services [XKMS03] have primarily been designed in order to abstract away some of the technical PKI detail in order to work with relatively simple clients. The generic protocol description for XKMS

should allow it to support any of the PKI examples discussed in Section 3. Key management issues are part of the key registration service component (X-KRSS), while key validation issues are part of the key information service component (X-KISS). A common Web services interface may go some way to aid the otherwise difficult process of integrating these components with existing software.

An area that is yet to be widely embraced in real implementations concerns the nature of the identifier used in a certificate. There are times when there is a legitimate need for this identifier to be veronymous, other times when a pseudonym would be preferable, and still other times when an anonym should be used (even within a single environment). Yet existing CAs are typically built to use only a single type of identifier (perhaps, if the CA is very flexible, in a range of formats). Standards, in their language and in their syntax, do not generally preclude the use of different identifier types, but history and tradition have made rigid interpretations and resulted in PKI deployments that are almost exclusively one type or another. More flexibility in this area (i.e., CAs that can bind keys to any of the three types, as required) would make PKIs more suited to many real-world requirements.

The goal of this paper has been to demonstrate that the PKI community has significantly broadened its understanding of this technology over the past ten years. The challenge now is to translate that understanding to real PKI implementations that solve authentication challenges in real, heterogeneous environments.

Acknowledgements

The authors wish to thank the reviewers for their numerous comments. They greatly improved the content and readability of this paper.

7 References

[AADS99] L. Wheeler, "Account Authority Digital Signature and X9.59 Payment Standard", slides presented at the *3rd CACR Information Security Workshop*, June 1999. (http://www.cacr.math.uwaterloo.ca/conferences/1999/isw-june/wheeler.ppt)

[AL03] C. Adams, S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition*, Addison-Wesley, 2003.

[DH76] W. Diffie, M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory,* Vol. 22, No. 6, November 1976, pp. 644.

[ElSc00] C. Ellison, B. Schneier, "Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure", Computer Security Journal, vol.XVI, no.1, 2000.

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

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

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

[Kohn78] L. Kohnfelder, "Towards a Practical Public-key Cryptosystem", *MIT Thesis,* May. 1978.

[OAS03] P. Doyle, S. Hanna, "Analysis of June 2003 Survey on Obstacles to PKI Deployment and Usage", report of the *OASIS PKI Technical Committee*, v1.0, 8 August 2003. Available at http://www.oasis-open.org/committees/pki/pkiobstaclesjune2003surveyreport.pdf.

[PGP99] J. Callas, "The OpenPGP Standard", slides presented at the *3rd CACR Information Security Workshop*, June 1999. (http://www.cacr.math.uwaterloo.ca/conferences/1999/isw-june/callas.ppt)

[PGPks] The MIT PGP Key Server; see http://pgp.mit.edu/

[PKIX-WG] IETF Public-Key Infrastructure X.509 (PKIX) Working Group; see http://www.ietf.org/html.charters/pkix-charter.html

[Resc01] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, 2001.

[RFC2560] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol – OCSP", Internet Request for Comments 2560, June 1999.

[RFC2692] C. Ellison, "SPKI Requirements", *Internet Engineering Task Force (IETF) Request for Comments (RFC) 2692*, September 1999.

[RFC2693] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylönen, "SPKI Certificate Theory", *Internet Engineering Task Force (IETF) Request for Comments (RFC) 2693*, September 1999.

[RFC3280] R. Housley, W. Polk, W. Ford, D. Solo, "Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile, Internet Request for Comments 3280, April 2002.

[RFC3379] D. Pinkas, R. Housley, "Delegated Path Validation and Delegated Path Discovery Protocol Requirements", Internet Request for Comments 3379, September 2002.

[RFC3647] S. Chokhani, W. Ford, R. Sabett, C. Merrill, S. Wu, "Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practices Framework", Internet Request for Comments 3647, November 2003.

[SDSI96] R. Rivest, B. Lampson, "SDSI – A Simple Distributed Security Infrastructure", 17 September 1996, http://theory.lcs.mit.edu/~rivest/sdsi10.html

[SSH03] T. Ylonen, D. Moffat, "SSH Protocol Architecture", *Internet Draft*, October 2003. Available at http://www.ietf.org/internet-drafts/draft-ietf-secsh-architecture-15.txt.

[WhTy99] A. Whitten, J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.", in *Proceedings of the 9th USENIX Security Symposium*, August 1999.

[X509-00] ITU-T Recommendation X.509. *Information Technology – Open Systems Interconnection – The Directory: Public Key and Attribute Certificate Frameworks*. March 2000 (equivalent to ISO/IEC 9594-8:2001).

[XKMS03] P. Hallam-Baker, "XML Key Management Specification", W3C Working Draft, v2.0, 18 April 2003. Available at http://www.w3c.org/2001/XKMS/.

An Examination of Asserted PKI Issues and Proposed Alternatives

John Linn, RSA Laboratories, Bedford, MA, USA
Marc Branchaud, RSA Security Inc., Vancouver, BC, Canada

15 March 2004

1 Introduction

Since the 1980s, public-key infrastructures (PKIs) have been widely anticipated as a primary means to make entities' keys available to others in a trusted fashion, thereby enabling a qualitative improvement in the protection and assurance of communications and transactions carried out over the Internet. Certificate-based authentication has become common practice in certain contexts, particularly in conjunction with SSL-protected web sites. In recent years, however, many commentators have lamented the fact that PKI has not achieved more pervasive adoption and deployment. Some, like [Clar01], [ElSc00], and [Gutt02], have concluded that PKI is a failure or does not address users' primary security needs. Opinions differ on the reasons for these results, but most can be distilled into a few general categories:

- A belief that demand for the services offered by PKI, in terms of PKI-integrated applications and/or security-oriented use cases for those applications, has not yet emerged to a degree sufficient to motivate deployment of a trust infrastructure.
- A belief that characteristics of current PKI architectures and implementations make them unnecessarily difficult to deploy, and/or that those characteristics render them incapable of delivering value which alternate approaches could achieve.
- A belief that deployment of PKI technology intrinsically implies and enforces a higher assurance environment than is appropriate or cost-effective in many operational contexts.

A 2003 survey undertaken by the OASIS PKI Technical Committee [Hann03] on obstacles to PKI deployment and usage suggests a mix of factors spanning each of these categories. If increased PKI adoption is taken as a goal, the first interpretation suggests a strategy of promoting applications and usage modes that would make use of certificates. Existing PKI technologies would stand ready to satisfy the demand if and as it emerges. While incremental changes might remain necessary to satisfy integration requirements, fundamental PKI architectures could safely remain intact. Questions of candidate applications and usages for PKI technology are interesting and important, but lie outside this paper's scope.

The second and third interpretations imply criticisms of elements within the PKI technology base, and motivations to revisit and modify those aspects of PKI that are considered to be contentious or problematic. Different commentators have expressed concerns about different elements of PKI technology, and have proposed different alternatives as a result; the goal of this paper is to examine a range of perceived issues and suggested approaches,

not to assert that all are equally valid or appropriate. Following this introduction, we characterize various perceived problem areas. Then, we examine several proposed approaches, seeking to characterize them in terms of the goals that they address, and the properties and value that they offer. We conclude by assessing asserted problems, and the contributions that suggested solutions make towards those problems.

This paper focuses on architectural and functional aspects of PKI. It is not primarily concerned with encoding alternatives, such as choices between ASN.1 and XML representations for protocol objects. For purposes of discussion, we assume the following elements as aspects of the contemporary PKI baseline, and therefore do not consider them under the category of candidate future variations:

- Support for hierarchic and non-hierarchic trust models
- Support for certificate revocation via Certificate Revocation Lists (CRLs) and via basic on-line status query mechanisms such as OCSP
- Syntactic support within certificates for a range of name forms, such as X.500 Distinguished Names, Internet-form names within AltName extensions, and pseudonyms.

While particular enhancements can be considered within many of these areas, their general premises have been widely presented and adopted, so do not constitute qualitative shifts from current accepted practice.

2 Contentious Aspects of PKI

In this section, we discuss several aspects of PKI technology and its operation that have attracted criticism and controversy.

2.1 Difficulty in Retrieving Keys and Certificates

To perform operations using public keys, those public keys must be available at the point where the operations are to be performed. In a conventional certificate-based PKI, this implies that a sender cannot encrypt a message for a recipient unless the recipient has already obtained a certificate and has made the certificate available to the sender (whether by direct transfer or posting on an accessible repository). If off-line operation is required, the appropriate certificates must be obtained in advance, when connectivity is available. Since large-scale directories have not become widely available to serve as certificate publication vehicles, interest has grown in approaches that enable public-key encryption to be performed without first satisfying these preconditions.

2.2 Questionable Value of Certified Key Representations

Certificates' usage practice reflects characteristics of environments for which they were originally developed, where it was considered inappropriate or impractical to rely on on-line availability of trusted servers. A primary goal of certificates' design was to represent keys and their bindings to named principals in an integrity-protected form, whose content could be stored safely on unprotected repositories or transferred across unprotected channels. Retrieval of a certificate requires that a suitable repository be available, but use of signed representations abstracts away the need to depend on that repository for security

properties other than availability. If, instead, keys are stored and retrieved from trusted servers, some of the rationale for representing them within signed certificate objects becomes superfluous. Channel-level mechanisms can protect a key from attackers while in transit between a server and a client, and can assure the client that it is receiving a key from a securely identified source.

2.3 Certificate Processing Complexity

PKI technologies have been criticized as being difficult to integrate with the applications that could make use of their services, requiring significant PKI-specific security expertise on the parts of application writers and maintainers. Today's X.509 certificates, e.g., have evolved into complex structures, with processing semantics that are far from trivial; this is primarily a matter of the information they carry, although it also involves its representation and encoding. Formalization and simplification of these semantics may represent a valuable area for investigation.

Some of the complexity in certification results from a desire for a certificate to include a comprehensive set of ancillary information so that it can be used for off-line processing, without consulting other trusted entities on an interactive basis. Increasingly, however, PKI models are evolving to include on-line components, which can offer alternative information sources to complement the certificates themselves.

Revocation mechanisms have long been recognized as a complex element in PKI, and path construction also introduces complexity [Elle01]. Despite the design attention that has been paid to revocation, it appears today that only a relatively small proportion of accepted certificates are actually checked for revocation status on an ongoing and timely basis.

2.4 Costly Certificates

Many assumptions about certificate usage have been based on a premise that certificates are expensive, and therefore that they can only be issued sparingly and infrequently. Some enrollment methods strive to provide confidence commensurate with high-value transactions and high-assurance client implementations, entailing high monetary costs and/or cumbersome registration processes. While this practice is appropriate for some types of technology (e.g., one-time placement of a user's long-term certificate into a smart card), and may be necessary to provide high levels of accountability, it need not be an intrinsic characteristic associated with the use of PKI methods. Imagine, by comparison, how computing might have developed if it had become accepted practice that an independent organizational authority needed to be consulted (and, possibly, paid) whenever a file was to be created. Most likely, only a subset of information, perhaps associated with a subset of critical users, would be deemed to warrant file representation. Other data would be stored and shared using different objects without the constraints associated with files. For a PKI, even when high levels of administrative assurance are not required, certification paradigms can be retained and adapted rather than developing or applying separate types of infrastructures to bind principals, keys, and attributes.

Dynamic issuance of certificates, which may be short-lived to avoid the need for separate revocation infrastructures, may allow new and innovative PKI models to be constructed.

In the Security Assertion Markup Language (SAML) [Male03], e.g., assertions bearing the Holder-of-Key confirmation method can take the form of signed objects carrying public keys, used to enable the corresponding private keys' holders to gain access to resources. Servers are expected to issue such assertions frequently, as needed to support authentication or resource access operations; no laborious procedures are required when an assertion is coined. Further, a number of on-line PKI key registration protocols (e.g., CMP [AdFa99], XKMS's X-KRSS [W3C03]) have been defined, which can provide the basis for interactive certification. The form of the resulting object, whether X.509, XML, or another format, need not imply or dictate the scope of procedural processing that is appropriate before the object is issued.

2.5 Problematic Cross-Domain Trust Management

The prospect of applying PKI technology to establish trust across heterogeneous domains can be daunting, both in administrative and technical terms. Some PKI architectures have sought to provide a sufficient basis to allow parties in different jurisdictions to engage in high-value transactions with one another, without prior shared knowledge beyond that manifested in the PKI. Few other technologies have attempted such ambitious goals, and it is debatable whether other approaches would necessarily achieve greater success in solving such a fundamentally challenging problem. In cases where the level of required assurance can be constrained, it may become easier to achieve (and benefit from) PKI-enabled interoperability.

PKI technologies can be applied to manifest trust relationships rooted at remote entities. Some (e.g., [DoEl02]) have argued, however, that users' trust is primarily local, and should be based on direct personal knowledge of other individuals. If this premise is accepted, reliance on remote roots is not considered practical or useful, and the ability to represent such trust relationships offers only irrelevant complexity.

Meaningful algorithmic translation of policies across domain boundaries is a significant challenge; often, the mapping between different organizations' policy elements can be based on administrative practices and interpretations that are difficult to encode. Management of inter-domain validation and trust relationships within a relatively small set of entities (e.g., bridge CAs, domain-level Delegated Path Validation (DPV) servers interacting with their peers representing other domains) may help to contain and simplify some aspects of the problem.

2.6 Naming Semantics

Naming plays an important role in PKIs, as public keys are typically bound to named entities. Conventional PKIs have been criticized for seeking to manifest a global naming structure that some view as fundamentally unrealistic. As with trust, some view naming as intrinsically local; further, given duplications among human individuals' names, ambiguities can arise in identifying a particular person based on his or her location in a distributed namespace. In some alternate approaches, e.g., SDSI [RiLa96], entities are named in a relative manner extending from one principal, and then can be linked to other principals through intermediary hops.

Another aspect of PKI entity names is the degree to which a name form matches or resembles names that people and software use on a regular basis. This has a direct bearing

on how useful the name is to the user or application that is trying to accomplish a security goal. Some PKIs – such as Pretty Good Privacy (PGP) [Call98] and the DNS security extensions (DNSSEC) [East99] – employ name forms that match their environments (or, rather, they adopt the name form of their environment). X.509 is an example of a PKI that started out adopting the name form of its environment (X.500 Distinguished Names), but then grew to accommodate application-specific names (through the Alternative Name extensions). A SDSI "well-defined" name – one that links a local name space to a particular principal, such as (using SDSI's "syntactic sugar") `jim's john's joe's jack` – is only meaningful to the SDSI PKI. However, each individual local name is an arbitrary string, and so can be meaningful to an application. For example, `10.1.1.1` might be a local SDSI name assigned to a VPN server whose IP address is, presumably, 10.1.1.1. PKIs with PKI-specialized name forms require applications to translate between their native name form and the PKI's, a process that can be error-prone and introduce security risks.

PKI	Name Locality	Name Form Utility	
		Application	PKI
PGP	Low	High	Low
DNSSEC	Low	High	High
X.509	Low	High	Low
SPKI/SDSI	High	Medium	High

Table 1 - PKI naming properties

A third property of PKI names is the degree of utility that the name has to the PKI itself. By "degree of utility" we mean the efficiency with which the PKI can use the name to obtain and validate a public key. PKIs provide keys by discovering and validating paths between entities, and so the PKI-efficiency of a name can be measured by the amount of path information that it encodes. Well-defined SDSI names are an extreme example of a name form that is almost entirely devoted to expressing path information, so much so that a (non-global) SDSI name is usually only meaningful to a single entity. DNSSEC names also encode a large amount of path information. In contrast, PGP names are email addresses, which are completely devoid of any PGP PKI path data. X.509's names – all of them – also contain no X.509 path information whatsoever.

Table 1 summarizes the naming properties for various PKIs. SDSI scores highly for name form PKI utility because of its well-defined names, but only moderately for application utility because although an individual local name can be an application-meaningful string, there are no conventions for an application to reliably extract a meaningful local name from a SDSI certificate. A VPN client, for example, has no way to tell that the `10.1.1.1` name in a SDSI certificate is supposed to be the IP address of a VPN server.

Recent PKI proposals have emphasized certificate processing and cryptographic methods rather than naming. A viable naming strategy seems to be a factor in a PKI's success, but it is not clear what combinations of properties (per Table 1) offer most value. Naming strategies do appear to require some consideration, and yet they remain relatively unexplored. Some of the questions that arise include:

- Are there any other useful naming properties?
- Is it necessary or desirable to rank highly in all of these properties?
- Have approaches to naming had an impact on PKI deployment? We note, for example, that an X.509 certificate in fact has two names – Issuer and Subject

– which together provide a small amount of path information. Would more (or less) path information in the certificate help or hinder widespread deployment of an X.509 PKI?

2.7 Use with Insecure Clients

Some PKI architecture premises were developed in anticipation of widespread security features at user clients, e.g., smart cards encapsulating users' private keys and cryptographic processing capabilities so that the keys need never be exposed elsewhere. Such implementations are particularly desirable when the keys mediate access to particularly sensitive data or resources, or when strong accountability (i.e., a non-repudiation service) is tied to their use. While such environments are gradually becoming more common (as with use of SIMs and other cards), most candidate PKI user applications continue to reside on platforms that offer limited security. From an attacker's viewpoint, the strength of a cryptographic algorithm can become irrelevant if its keys can be obtained by attacking a weak platform. Where high assurance is required, these arguments motivate approaches that perform cryptographic processing in other entities, whether protected devices or shared services, and/or distribute the processing with such entities.

There are many cases, however, where the assurance level of commercial platforms is an adequate basis to support useful, interoperable security. Use of PKI need not also imply use of specialized, higher-security technologies by clients; higher assurance requirements may be warranted at CAs, as misuse of a single CA private key can compromise an entire community. Today, it is common practice to store user keys in a password-encrypted form. It is arguable that passwords used to unlock private keys may warrant higher quality or tighter protection than other passwords, as the keys they release can enable direct authentication to multiple entities rather than just to a single system, but user convenience may conflict with such measures.

2.8 Privacy Compromises

It has been observed, e.g., in [Bran99], that conventional PKI is unfriendly to privacy, as its certificates provide persistent, widely visible linkages between keys and principal identifiers. This property is appropriate in contexts where authorizations or signatures depend on individuals' authenticated identities, but not all possible uses of public-key technology fit this model. Even if data messages are encrypted, patterns of certificate acquisition and usage can reveal identities of principals and their communicating peers; a certificate validation server could be particularly well placed to collect such information. Certified pseudonyms can provide a partial countermeasure, but do not satisfy all privacy goals; if a fixed pseudonym is used to represent a principal to multiple sites for an extended period, the sites can use it as the basis to collect an extensive behavior profile which may then be associated with an individual.

Use of X.509 certificates to hold principal attributes other than identifiers has been proposed and considered for some time, recently in [FaHo02], though has not yet achieved wide adoption. Attribute statements within SAML assertions are another form of attribute representation within a signed object corresponding to a principal. Both have the property of disclosing an aggregate set of attributes to their certifier and to the parties that rely

on the certified object, even if not all of these entities necessarily require the full set of information.

3 Proposed Approaches

In this section, we examine approaches that have been proposed as extensions or alternatives to conventional PKI technologies, addressing one or more of the concerns identified in the preceding section.

3.1 IBE and Related Work

The concept of Identity-Based Encryption (IBE) has been considered in the cryptographic community for some time, and recent work has yielded a variety of methods realizing variations on the concept. Some, but not all, approaches in this group allow a sender to prepare a protected message for a recipient without first obtaining a certificate for the recipient. This section considers some of their properties.

3.1.1 Identity-Based Encryption

IBE, surveyed in [Gagn03], enables senders to encrypt messages for recipients without requiring that a recipient's key first be established, certified, and published. The basic IBE paradigm allows a sender to determine the key to be used to encrypt for a particular recipient based on the recipient's identifier; the recipient derives the corresponding decryption key through interaction with a Private Key Generator (PKG) system. While the sender must determine the PKG corresponding to a particular recipient, and must obtain domain-level parameters associated with that PKG, it need not obtain information specific to an individual recipient before encrypting a message. The basic IBE approach implies intrinsic key escrow, as the PKG can decrypt on behalf of the user. Variant approaches ([AlPa03] [Gent03]) cited below apply some aspects of IBE, but seek to avoid the escrow characteristic.

3.1.2 Certificateless Public Key Cryptography

This approach, proposed in [AlPa03], incorporates IBE methods, using partial private keys so the PKG can't decrypt on behalf of the user. These are combined with secret information held by the recipient, yielding a public key that the recipient can publish and/or transfer directly, but for which no certification is required. Would-be senders must, however, first obtain that key through some means in order to encrypt a message for a recipient. Publication of a key for this method may not prove significantly easier than publishing a conventional PKI certificate. In fact, the publication problem could become significantly worse, since use of the approach might imply a need for frequent republication in lieu of a revocation mechanism.

3.1.3 Certificate-Based Encryption

This approach, proposed in [Gent03], incorporates IBE methods, but uses double encryption so that its CA can't decrypt on behalf of the user. A sender must obtain a recipient's certificate in order to encrypt a message for a recipient. In order for a recipient to decrypt successfully, he/she must have both a current CA-issued certificate and a personal secret key; use of IBE methods in certificate generation means that the same certificate used by

the sender to encrypt is also used by the recipient as part of the decryption process. Frequent certificate updates are performed, so that senders need not separately check revocation status of the certificates they obtain.

3.2 PKI Augmented with On-Line TTP

Some properties similar to those of IBE can be achieved by augmenting conventional PKI with an on-line trusted third party (TTP) system. Two classes of TTP-based operations can be considered:

- Encryption using a TTP's public key rather than one associated with an individual recipient; in this case, a recipient could request that the TTP perform decryption services on his/her behalf, or a message could be routed to the TTP which would then decrypt it and forward the result to the recipient. This eliminates the need for recipients to register individual key pairs, and for senders to obtain per-recipient keys; it implies that the TTP can decrypt all recipients' traffic and requires involvement by the TTP in order to process each of their messages. [DeOt01] provides examples and discussion of this type of approach.
- Encryption using an individual recipient's public key, which the sender would request from the TTP. For already-registered recipients, a TTP (such as that suggested in [Dier03]) would provide their existing keys or certificates. Additionally, such a TTP could revoke keys or certificates by removing them from its store. If no public key or certificate existed for the recipient at the time of the request, the TTP would generate one dynamically, provide the public component to its requester, and make the corresponding private key available to the recipient. In this model, the TTP's possession of recipients' private keys need not be more than temporary in nature, pending their retrieval by the corresponding recipient.

The second type can be considered as a example of a general class which has previously been considered in various contexts but has not become part of the PKI mainstream, that of "on-the-fly PKI" approaches where certificates are signed dynamically as needed rather than being generated by a CA in advance as a prerequisite to secure operation. Such certificates and the keys they certify can be short-lived, enabling particular operations or use of a session while becoming disposable thereafter. Some other examples include the delegation certificates that represent login sessions within Digital Equipment Corporation's Distributed System Security Architecture (DSSA) as proposed ca. 1990 [GaMcD90], and recent IETF-PKIX contributions on proxy certificates [Tuec03].

3.3 Distributed Computation

Methods have been developed (see, e.g., [Gold02]) that distribute cryptographic operations so that the cooperative contribution of a number of entities is required in order to perform an operation such as a signature or a decryption. Use of such measures could help to ameliorate the risks associated with insecure client platforms; even if such a client's keys were compromised, they would be insufficient to impersonate the client's associated user.

Analogous to the case with IBE, some similar properties can also be achieved without specialized cryptography by holding a user's keys at a server, which would perform operations on behalf of the user upon receipt of an authenticated request. This strategy can take advantage of tighter protection at servers vs. clients, but implies that the users must fully trust the servers to apply their keys appropriately.

3.4 Alternative Validation Strategies

PKI's original Certificate Revocation List (CRL) mechanisms implied significant storage, communications bandwidth, and processing overhead, yet could only provide revocation with significant time latency. Newer on-line approaches, such as OCSP, SCVP, and XKMS, address many of these concerns, but introduce requirements for trusted on-line servers to process certificates and for connectivity between the servers and their relying parties. Their effective revocation latencies can vary, as a result of caching and when information updates are available only on a periodic basis. These approaches' capabilities, and the extent to which clients must trust the servers, increase as the scope of server-based processing extends from revocation checking on single certificates to acquisition and validation of full certification paths, and from independent, self-contained validation servers to distributed networks of cooperating validators. More broadly, however, the extent of trust required should correspond to the value of the information that the underlying certificates protect. Further discussion of validation alternatives and their prospects and implications can be found in [BrLi02].

Hash-tree approaches (e.g., [Mica02] [NaNi98]) have been proposed, offering compact, protected representations of the status of large numbers of certificates. Their value is most apparent for PKIs operating at extremely large scale; in smaller contexts, such as within typical enterprises, their benefits relative to CRLs appear less compelling. Like CRLs, they reflect certificate status information only at fixed intervals, rather than with the immediacy that on-line status queries can offer.

Levi and Caglayan [LeCa00] propose the concept of "nested certificates" in order to avoid some of the performance burdens associated with verification of long certification paths. Several variations are suggested, but a general premise is that a hierarchy's higher-level CAs certify not only their immediate descendants but also directly certify members of more distant generations. While this approach can indeed reduce the number of certificates in a validated path, it appears to suffer from a serious flaw. Among other reasons, CA hierarchies are constructed in order to distribute certification responsibilities, and to place them in hands close to the principals they certify. In a condensed hierarchy, higher-level CAs would need to be involved in enrollment of remote generations, and potentially to generate very large numbers of certificates. In the limit, a CA hierarchy could be flattened to a single CA, making any hierarchy below it moot, but such an approach is unlikely to be attractive from a technical or policy perspective.

3.5 Key Servers

Given today's generally high level of connectivity, and widespread interest in simplifying client-side operations, an emerging approach is to use servers to perform some, or all, certificate processing. Clients would delegate certificate path discovery and/or validation to a trusted server (see [PiHo02]).

DPV, in particular, changes the basic PKI model. A DPV server assumes the primary responsibilities of a traditional CA, from the client's perspective. That is, the client relies upon the server to ensure correct correspondences between principals and their public keys.

This approach has implications for assurance and availability, especially when a DPV server relies on other DPV servers (see [BrLi02]). However, once the premise of trusting an on-line server for certificate retrieval and validation is accepted, it is only an incremental step to relying on the server to provide the bare key over a secure channel – eliminating the need for the client to process certificate formats entirely. Such an approach is one of the models supported within XKMS's X-KISS [W3C03].

The full impact of delegating key validation and acquisition to servers has yet to be investigated. The benefits to PKI client applications for smaller, simpler code are apparent, but it is not yet clear what effects delegated key servers will have on a PKI's policies and procedures, or what levels of assurance are enabled (or disabled).

3.6 Privacy Protection

Some PKI privacy implications can be ameliorated by reducing the amount of principal-related information bound within a single certificate or other signed object. Certified pseudonyms can easily be supported, and are appropriate and sufficient in many operational contexts. Further, use of attribute certificates (ACs) can offer privacy advantages over placement of attributes within public-key identity certificates (PKCs). Even in the common case where an AC is bound to a PKC for use, implying a linkage to the PKC within the AC, the PKC's contents need not disclose all of the ACs that may be used with it. This modularity allows the attributes within ACs to be disclosed selectively, when needed in order to support a particular access request, and to remain confidential otherwise. To take advantage of this capability, it is desirable for accessors to present ACs selectively along with requests rather than posting them for general access within a directory or other repository.

Use of on-line certificate validation services introduces the prospect of user tracking, if the validation service can identify the set of locations from which a certificate's status is queried. Aggregation and/or anonymization of status requests can help to mitigate this concern.

Stefan Brands, in [Bran99], proposes cryptographic certification techniques which address privacy goals outside the scope of traditional PKI models, and which imply different assumptions and paradigms for PKI protocols and interactions. Brands' techniques seek to allow certificate holders to disclose certified attributes selectively in a general manner, and to limit the extent to which presentation of certified attributes can be proven to third parties by recipients. Cryptographic blinding is used for certificate issuance, so that not all of the attributes represented within a certificate need be visible to a particular issuing CA. These approaches can provide privacy assurance unavailable in conventional PKIs, particularly in terms of constraining the scope of trust that a certified user must place in a CA and of countering use of certificates as a means to aggregate data. Their operational models would require changes in certificate-based protocols, one factor which would likely complicate their deployment.

4 Conclusions

Any concrete system can suffer in comparison with a hypothetical, ideal alternative. PKI has been a particularly attractive target, perhaps partly because it has sometimes been perceived and promoted as a general panacea, intended to solve even organizational issues outside the realm of technology, rather than as a technical answer to clearly understood and practically achievable requirements. Variations to many aspects of PKI are possible and worthy of consideration, but an appropriate comparison between practice and proposal requires a specific alternative and an understanding of its impact on the system as a whole.

Certificates have been criticized for a variety of reasons, particularly:

- Processing complexity and overhead, including both the contents of certificates and the usage of signed representations to carry those contents; many of these characteristics derive from design assumptions which presumed off-line certificate processing without reliance on trusted servers, and use of such servers may allow significant simplifications.
- Association with operational models that imply high costs for certificate issuance; here, the use of a signed key-bearing object should properly be distinguished from a particular type of deployment. Public-key methods can be used to construct a wide variety of useful approaches with different assurance, semantics, and dynamics.

PKI has also been criticized on the basis that it fails to render the problems of securely interconnecting different entities and trust domains simple. These problems are fundamentally difficult, for organizational as well as technical reasons. Few proposals outside the realm of PKI have attempted to satisfy these concerns comprehensively, though trust management research activities [Blaz99] have proposed various supporting mechanisms. Generally, PKIs' trust management capabilities should be evaluated in terms of their supporting contributions to distributed security, rather than against an expectation that all such requirements should be satisfied solely by PKI or any other technology.

Much cryptographic research activity has concerned forms of IBE, applied to avoid the need for senders to retrieve certificates from repositories. Unfortunately, many proposed alternatives substitute different publication requirements, or introduce implicit key escrow properties. Other computational methods can distribute processing, mitigating some of the impact of key compromise at weakly protected clients. These cryptographic innovations provide elegant approaches, but many of their properties can also be achieved by using trusted third parties with conventional cryptographic algorithms.

Fundamentally, PKIs exist to provide public keys that correspond to principals, in a fashion enabling other parties to rely on their correspondence. This function is an essential basis on which to construct secure distributed computing environments, and necessarily implies some form of infrastructure. Many PKIs seek to provide high levels of technical and procedural assurance, particularly at CAs, but some of these measures may not be necessary for environments where the ability to communicate with at least some level of protection takes precedence over especially strong security guarantees. Naming is a cen-

tral element in PKI, and further research focused on aspects of alternate naming methods may warrant attention.

Certificates are a convenient, self-sufficient means of representing keys, but their use may become superfluous in server-centered environments. Further, new PKI models can evolve based on signed key-bearing assertions; these objects can provide the same functions as certificates, but are emerging unbounded by existing assumptions about how certificates must be created, processed, and managed. Generally, it seems that PKI suffers today from a perception that it can assume only a particular, monolithic form; to satisfy a broad range of applications and environments, it must be possible for its underlying methods to be composed and applied in a variety of ways.

5 Acknowledgment

The authors would like to acknowledge this paper's anonymous reviewers for comments helping to improve its final version.

6 References

[AdFa99] C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure Certificate Management Protocols", Internet RFC-2510, March 1999.

[AlPa03] S. S. Al-Riyami and K. G. Paterson, "Certificateless Public Key Cryptography", IACR Cryptology ePrint Archive paper 2003/126, 2 July 2003.

[Blaz99] M. Blaze, J. Feigenbaum, J. Ioannidis, A. D. Keromytis, "The Role of Trust Management in Distributed System Security", in *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*, Springer-Verlag Lecture Notes in Computer Science State-of-the-Art Series, pp. 185-210, Berlin, 1999.

[Bran99] S. Brands, "Rethinking Public Key Infrastructures and Digital Certificates – Building in Privacy", PhD Dissertation, University of Utrecht, October 1999.

[BrLi02] M. Branchaud, J. Linn, "Extended Validation Models in PKI: Alternatives and Implications", 1st PKI Research Workshop, Gaithersburg, MD, April 2002.

[Call98] J. Callas, et al., "OpenPGP Message Format", Internet RFC-2440, November 1998.

[Clar01] R. Clarke, "The Fundamental Inadequacies of Conventional Public Key Infrastructure", Proceedings, ECIS'2001, Bled, Slovenia, June 2001. Available at http://www.anu.edu.au/people/Roger.Clarke/II/ECIS2001.html. (Date of access: 26 November 2003.)

[DeOt01] T. Dean, W. Ottaway, "Domain Security Services Using S/MIME", Internet RFC-3183, October 2001.

[Dier03] T. Dierks, "Re: Fwd: [IP] A Simpler, More Personal Key to Protect Online Messages". Message posted to Cryptography electronic mailing list, ar-

chived at http://www.mail-archive.com/cryptography@metzdowd.com/msg00409.html. (Date of access: 4 December 2003.)

[DoEl02] S. Dohrmann, C. Ellison, "Public-key Support for Collaborative Groups", 1st PKI Research Workshop, Gaithersburg, MD, April 2002.

[East99] D. Eastlake, "Domain Name System Security Extensions", Internet RFC-2535, March 1999.

[Elle01] Y. Elley, et al., "Building Certification Paths: Forward vs. Reverse", NDSS-01, San Diego, 2001.

[ElSc00] C. Ellison, B. Schneier, "Ten Risks of PKI: What You're Not Being Told about Public Key Infrastructure", Computer Security Journal, Vol. XVI, No. 1, 2000. Available at http://www.schneier.com/paper-pki.html. (Date of access: 4 March 2004.)

[FaHo02] S. Farrell, R. Housley, "An Internet Attribute Certificate Profile for Authorization", Internet RFC-3281, April 2002.

[GaMcD90] M. Gasser, E. McDermott, "An Architecture for Practical Delegation in a Distributed System", Proceedings, IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, May 1990.

[Gagn03] M. Gagné, "Identity-Based Encryption: a Survey", RSA Laboratories Cryptobytes, Vol. 6, No. 1, Spring 2003.

[Gent03] C. Gentry, "Certificate-Based Encryption and the Certificate Revocation Problem", EUROCRYPT 2003, LNCS 2656, pp. 272-293, 2003.

[Gold02] O. Goldreich, "Secure Multi-Party Computation (Final (Incomplete) Draft Version 1.4)", 27 October 2002. Available at http://www.wisdom.weizmann.ac.il/~oded/pp.html. (Date of access: 19 December 2003.)

[Gutt02] P. Guttman, "PKI: It's Not Dead, Just Resting". Available at http://www.cs.auckland.ac.nz/~pgut001/pubs/notdead.pdf. (Date of access: 5 March 2004.)

[Hann03] S. Hanna, ed., "Analysis of August 2003 Follow-up Survey on Obstacles to PKI Deployment and Usage", OASIS PKI Technical Committee, 1 October 2003. Available at http://www.oasis-open.org/committees/pki/pkiobstaclesaugust2003surveyreport.pdf. (Date of access: 4 March 2004.)

[LeCa00] A. Levi, M. Caglayan, "An Efficient, Dynamic, and Trust Preserving Public Key Infrastructure", Proceedings, IEEE Computer Society Symposium on Research in Security and Privacy 2000. IEEE, Piscataway, NJ, USA, pp. 203-214.

[Male03] E. Maler, P. Mishra, R. Philpott, eds. (2003), "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1." OASIS Standard.

[Mica02] S. Micali, "Novomodo: Scalable Certificate Validation and Simplified PKI Management", 1st PKI Research Workshop, Gaithersburg, MD, April 2002.

[NaNi98] M. Naor, K. Nissim, "Certificate Revocation and Certificate Update", 8th USENIX Security Symposium, San Antonio, January 1998.

[PiHo02] D. Pinkas, R. Housley, "Delegated Path Validation and Delegated Path Discovery Protocol Requirements", Internet RFC-3379, September 2002.

[RiLa96] R. Rivest, B. Lampson, "SDSI – A Simple Distributed Security Infrastructure", 30 April 1996.

[Tuec03] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, "Internet X.509 Public Key Infrastructure Proxy Certificate Profile", work in progress, IETF PKIX working group, 2003.

[W3C03] World Wide Web Consortium, "XML Key Management Specification (XKMS)", Version 2.0, W3C Working Draft, 18 April 2003.

Private Revocation Test using Oblivious Membership Evaluation Protocol

Hiroaki Kikuchi
Tokai University
Dept. of Information Media Technology,
1117 Kitakaname, Hiratsuka, Kanagawa, 259-1292, Japan
kikn@tokai.ac.jp

Abstract This paper presents a cryptographic protocol for the authenticated dictionary, namely, an untrusted directory provides a verifiable answer to a membership query for a given element. In our protocol, a user is able to retrieve whether or not a target element belongs to a database that the directory has without revealing which element he/she wishes to know against the untrusted directory. Our protocol requires linear exponentiations to the number of elements in the database, but achieves a constant size communication complexity between a user and a directory. The privacy of query is assured under the Φ-hiding assumption introduced by Cachin.

1 Introduction

1.1 The PKI Issue

Certificate revocation is a current topic of interest in public-key infrastructure (PKI). Traditionally, a list of revoked certificates (CRL) has been used to represent the periodic distribution of revoked information. To improve the bandwidth consumption of the entire CRL transmission, some mail agents have begun supporting an online protocol for providing users the status of a target certificate alone, instead of the full CRL. The Online Certificate Status Protocol (OCSP)[6] is the standard protocol now in common use. There have been several attempts to improve the efficiency and security of CRLs. Kocher proposed a hash-tree based revocation protocol known as CRT[8], Micali presented a linear linking scheme with $O(1)$ communication cost (CRS)[7], and Naor and Nissim formalized the problem as an authenticated dictionary [9] in which a B-tree is used to balance the tree while the tree itself is skewed while updating the database.

1.2 Privacy Issues

As these online protocols are now in widespread use, a new privacy issue has arisen. The OCSP method uses the following steps. Each time a digitally signed mail is received, then the mail agent picks up a certificate from the mail and automatically sends a query to check if the certificate is revoked to a server specified in the certificate. Hence, the server, known as the *CRL distribution point*, acquires the significant statistics of the PKI – who sends a message to whom, how often, and, even worse, a digital signature, which is often used when we send significant messages whose privacy we wish to preserve the most.

1.3 Privacy Information Retrieval

To overcome the privacy issues of revoked certificates, the private information retrieval (PIR) method is a suitable technique for a user to be able to retrieve a target data item from a database while hiding the identity of the target item from the server. The notion of a PIR was introduced by Chor, Goldreich, Kushievitz, and Sudan [4], and has already improved retrieval in terms of its communication and computation costs. One of the recent results by Beimel, Ishai, and Malkin [5] archives, for a given constant, $k \geq 2$, and the number of items in a database n, a k-server protocol with $O(n^{1/(2k-1)})$ communication, and $O(n/\log^{2k-2} n)$ computations at the server. The servers, however, are considered as untrustworthy parties in the PKI model because servers must be online and, thus, have greater chance of being compromised by an intruder. Therefore, the behavior on the server side is not guaranteed to be correct. In addition, the average user may have poor computational power and narrow bandwidth, with even just one server. Thus, a single server protocol making the cost at user side as small as possible is preferable for solving the CRL distribution problem.

1.4 Our Contribution

In this paper, we present a simple solution to the problem. Given an element, $x \in X$, a user performs a membership test if x is in a subset $L = \{x_1, \ldots, x_n\} \subset X$, requesting a query for a single non-trusted server who manages L steps without revealing x to the server. Our proposed protocol achieves a single server PIR with an optimal communication cost of $O(1)$ between a server and a user, and an optimum computation cost of $O(1)$ at the user side. To prevent the server from answering an improper response, a verification protocol that authorizes the answer from an authority is also provided.

2 Preliminaries

2.1 The PKI Model and Requirements

We have three types of parties: The *source* S or *Certification Authority* (CA), which is a trusted party that certifies the list of revoked certificates, *directories* D is non-trusted party who maintains the list and answers the questions that a target certificate is still active, instead of CA, and *users*, U, who wish to keep in touch with the current status of the certificates via the non-trusted D.

The CA is a source of information of revoked certificates and has the replication of the information distributed among directories. The directories of Ds work as carriers of the revoked information and are thus not responsible for the integrity of the database provided from the source. In terms of security, the directories have no secret information inside so that, even if one of directories is compromised, any rebuild of PKI is not necessary. The directories have a powerful computational power, e.g., the state-of-the-art CPUs, a secure coprocessor, and broad-bandwidth connections to each party. Since the directories are widely distributed over the network, we assume the risk that some of directories might perform an analysis of the access log from the end users using the data mining techniques. A user U communicates with one of the directories and checks if a target certificate is revoked or not and examines the integrity of responses from the directory server. We assume that some of the users may have limited computational power and a poor link of limited bandwidth. (In particular, this can happen when the user is mobile and with a PDA).

Oblivious Membership Evaluation:
Let X be the universal set of identities of certificate (64-bit serial numbers are often used in actual services), and $L = \{x_1, \ldots, x_n\}$ be a subset of X. The S gets L distributed among directories D. Given an element $x \in X$, U performs a membership query to D whether or not, $x \in L$ without revealing x to D.

The requirements of oblivious membership evaluation should satisfy are as follows:

1. **Privacy of query.** From a membership query of $x \in L$, D learns no information about x.

2. **Authenticity of source.** From the response from D, U verifies that the result of membership query is authorized by S and that D follows the steps properly.

3. **Efficiency.** The sizes of both query and answer should be independent of the number of PKI users, to which the size of CRL n seems to be proportional, and we want the sizes to be as small as possible. The computational costs at users should be also minimized.

2.2 Dynamic Accumulator

The RSA accumulator is proposed by Benaloh and de Mare[10], where a set of values are accumulated into a single object for which a witness that a given

value was incorporated into it is provided. Camenish and Lysyanskaya improves the RSA accumulator so that dynamic operations of insertion and deletion are feasible with independent cost from the number of values[12]. Goodrich, Tamassia, and Hasic show the pre-computations of witness reduces the computation overhead at the directory with the cost of communication consumption[11].

Informally, the RSA accumulator works as follows. The source picks strong primes p and q and publishes $N = pq$. Let L be a set of primes $\{x_1, \ldots, x_n\}$, representing identities (of the revoked certificates in PKI). The source then computes *accumulator*

$$A = a^{x_1 x_2 \cdots x_n} \pmod{N},$$

where a is a public constant that is relatively prime to N and publishes A together with digital signature $\sigma_S(A)$ on A. To prove an element $x_i \in L$, the directory computes *witness*

$$A_i = a^{x_1 \cdots x_{i-1} x_{i+1} \cdots x_n} \pmod{N}.$$

The user verifies witness by $A_i^{x_i} \pmod{N} \stackrel{?}{=} A$. Under the strong RSA assumption[12], the directory, which does not have the knowledge of factorization of N, is able to compute the witness A_i only when x_i belongs to L.

2.3 Φ-Hiding Assumption

Cachin present an efficient secure auction protocol that an oblivious party blindly compares two inputs bit-by-bit under the the ϕ-hiding assumption (Φ HA)[13]. Informally, the ΦHA states that it is computationally infeasible to decide whether a given prime divides $\phi(N)$, where m is a composite number of unknown factorization.

We say modulus m *hides* a prime p if N is a composite number $p'q'$ such that $p' = 2pp_1 + 1$ and $q' = 2q_1 + 1$ with primes p_1, q_1. Note that N hides p if and only if $p|\phi(N)$. The Φ HA states that, for a randomly chosen $N \in Z_N^*$ and primes p_0, p_1 such that N hides p_0 but does not hide p_1, the (N, p_0) and (N, p_1) is computationally indistinguishable.

An integer x is a *p-th residue* modulo m if there exists an α such that $\alpha^p = x \pmod{N}$. Let $R_N(p)$ denote a set of all p-th residues in Z_N^*. Then, note that only the party that knows the factorization of N and thus $\phi(N)$ is able to test if any given integer is a p-th residue by

$$a^{\phi(n)/p} \equiv 1 \pmod{m},$$

which holds if a is a p-th residue modulo m.

2.4 Proof of Conjunctive Knowledge

Cramer, Cramer, Damgard, and Schoenmakers presents an efficient zero-knowledge proof of conjunctive propositions [1]. By $PK\{(\alpha) : y_1 = g_1^\alpha \wedge y_2 = g_2^\alpha\}$, we denote a proof of knowledge of discrete logarithms of elements y_1 and y_2 to the

bases g_1 and g_2. Selecting random numbers r_1 and $r_2 \in Z_q$, a prover sends $t_1 = g_1^{r_1}$ and $t_2 = g_2^{r_2}$ to a verifier, who then sends back a random challenge $c \in \{0,1\}^k$. The prover shows $s = r - cx \pmod{q}$, which should satisfy both $g_1^s y_1^c = t_1$ and $g_2^s y_2^c = t_2$.

3 Oblivious Membership Evaluation

3.1 Overview

Our construction is based on Φ HA in order for users to blindly query a membership to a directory that has the list L. A user generates a modulus m that hides a prime x specifying the identity of a given certificate, and then sends a query consisting of non x-th residue c. The directory D then raises c to the power of all primes in S modulo m and sends the answer back to U, who then performs an x-th residue test using secret knowledge of factorization of m. In addition, we need a verification protocol to prevent a dishonest directory from cheating users. The witness in RSA accumulator cannot be applied here because the directory does not know which element is to be tested. Instead, we employ a zero-knowledge proof technique to show that the directory has raised a base to the power exactly the same exponents to that used by accumulator A.

3.2 Accumulator Setup

We begin with a set up protocol in which a source S notifies to the directories the list of currently revoked certificates.

1. The S picks strong primes P and Q and publishes $N = PQ$. For the list of revoked certificates $L = \{x_1, x_2, \ldots, x_n\}$, where x_i are small primes corresponding identities of revoked certificates, S computes accumulation

$$A = a^{x_1 x_2 \cdots x_n} \pmod{N},$$

   where a is a public constant that is relatively prime to N and publishes L, A, a together with a digital signature $\sigma_S(A, a, t)$, where t is the current time interval.

2. On receiving the list L and accumulator A periodically, every directory D updates the current (at a time t) database by L after it verifies the digital signature and accumulator $a^{x_1 x_2 \cdots x_n} = A \pmod{N}$.

3.3 Membership Test

Given a certificate to be examined, a user performs the following membership test protocol with one of the directories.

1. Given a target certificate specified by prime x, U chooses strong primes p and q such that $m = pq$ hides prime x. U picks an integer c that is not

p-th residue modulo m. U sends a query of the form (c, m) to one of the directories, D.

2. Then, D computes an answer

$$z = c^{x_1 x_2 \cdots x_n} \pmod{m}$$

and responds to U the answer z together with the accumulator A and digital signature $\sigma_S(A, a, t)$.

3. Finally, U locally performs the membership test

$$z^{\phi(m)/x} \pmod{m} = \begin{cases} 1 & \text{if } x \in L, \\ 1^{1/x} & \text{otherwise.} \end{cases}$$

Note that answer z becomes the x-th residue when there is an element in L that is equal to the target x.

3.4 Authenticity of the Source

To prevent a dishonest D from cheating users with improperly computed z, we require D to provide the proof of accumulating every element L into A by the form

$$PK\{\beta : a^\beta = A \wedge c^\beta = z\},$$

where private information β is ℓ defined by $\ell = x_1 x_2 \cdots x_n$ (note that this is not a modular multiplication), for which both $z = c^\ell$ and $A = a^\ell$ are satisfied. In other words, ℓ is a witness for which accumulator A is consistent with the answer z. Since D does not know the factorization of N nor m, we need the modified version of the proof of conjunctive knowledge mentioned in Section 2.4.

1. The D randomly picks r that is properly large (but is less than N and m) and computes

$$T = a^r \pmod{N}, \quad V = c^r \pmod{m}.$$

For T and V, D applies a secure cryptographic hash function H with properly large range to obtain a challenge $d = H(T||V)$, and computes (not modular arithmetic)

$$s = r + d\ell$$

and sends the proof (T, V, s) to U.

2. Then, U computes $d = H(T||V)$ and verifies that

$$a^s / A^d \stackrel{?}{=} T \pmod{N},$$
$$c^s / z^d \stackrel{?}{=} V \pmod{m}.$$

4 Evaluation

4.1 Security

Under the assumption of a secure digital signature scheme used by the source, the accumulator A at the time t is unable to be forged. Consider a dishonest directory that is trying to manipulate z to z' so that the membership test will fail for z' when x is in L. To convince users that the answer was correctly computed, the directory has to predict s that satisfies the above-mentioned equations for proof of knowledge. The probability of passing the test is negligibly small.

4.2 Privacy

If a malicious directory is able to determine which prime is hidden by a given m and c, it can immediately distinguish two composite numbers m_0 and m_1 that hide distinct primes, which contradicts the Φ-hiding assumption. Therefore, D is not able to learn the target x under the Φ HA. Moreover, D does not even know the result of the membership test at all.

4.3 Efficiency

The proposed scheme has the following performance:

- a size of query (c) sent from user to directory is $|m|$;
- a size of answer (z) sent from directory to user is $|m| + |N| + |\sigma|$ (without proof of knowledge);
- a size of proof (T, V, s) is $O(n)$ (since the magnitude of ℓ is linear to n);
- a number of modular exponentiations at the user is 1;
- a number of modular exponentiations at the directory is n.

Without the knowledge of $\phi(m)$, the size of ℓ increases with the number of elements in L; thus, the verification at the last step in the scheme requires $O(n|m|)$ modular multiplications, which is impractically heavy when n is too large.

One more inefficiency we should address is the key generation cost to the user, who should always pick a new modulus that hides the given prime.

5 Conclusions

We have presented a protocol for oblivious membership evaluation using the Φ-hiding assumption. The proposed protocol is efficient in terms of directory-and-user communication with $O(1)$, preserves the privacy of a query as to which certificate is to be examined, and provides verification steps that result in the membership query being correctly computed. Future studies include an efficient

verification independent of n and an improvement of n-size modular exponentiations at the directory.

References

[1] R. Cramer, I. Damgard, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in Proc. of *CRYPTO '94*, pp.174-187, 1994.

[2] J. Camenisch, and M. Michels, "Proving in zero-knowledge that a number is the product of two safe primes," in Proc. of *EUROCRYPT '99*, pp. 107-122, 1999.

[3] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *EUROCRYPT 1997.*

[4] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan, "Private information retrieval," in Proc. 36th IEEE Symposium on Foundations of Computer Science (FOCS), 1995.

[5] Amos Beimel, Yuval Ishai, and Tal Malkin, "Reducing the servers computation in private information retrieval: pir with preprocessing," in Proc. CRYPTO '00, LNCS vol. 1880, pp. 550, 2000.

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

[7] S. Micali, "Efficient certificate revocation", Technical Report TM-542b, MIT Laboratory for Computer Science, 1996.

[8] P. Kocher, "On certificate revocation and validation," in Proc. of Financial Cryptography'98, *Springer LNCS 1465*, pp. 172-177, 1998.

[9] M. Naor, and K. Nissim, "Certificate revocation and certificate update," in Proc. of Seventh USENIX Security Symposium, pp. 217-228, 1998.

[10] J. Benaloh, and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in Proc. of *EUROCRYPT*, LNCS vol. 839, Springer, pp. 216-233, 1994.

[11] M. Goodrich, R. Tamassia, and J. Hasic, "An efficient dynamic and distributed cryptographic accumulator," in Proc. of Information Security Conference (ISC 2002), LNCS Vol. 2433, Springer, pp. 372-388, 2002.

[12] J. Camenisch, and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in Proc. *CRYPTO 2002*, 2002.

[13] C. Cachin, "Efficient private bidding and auctions with an oblivious third party," ACM Conference on Computer and Communications Security (CCS), pp. 120-127, 1999.

"Dynamic Bridge" Concept Paper

Ken Stillson
Mitretek Systems

Written for presentation to the
3rd Annual PKI R&D Workshop

Table of Contents

Problem Background 3
Solution Background 5
Dynamic Bridge Concept 6
Proposed Benefits 9
Challenges and Unresolved Issues 10
Request for Feedback 11
References 12

Abstract

Current cross-certificates based PKI trust mechanisms suffer from a scaling problem. Even given the topological simplification of bridge CAs, as cross certificate meshes grow in size and complexity, the number of possible routes between points increases very quickly, and the time required for path discovery can increase beyond a tolerable delay for real-time operation. This paper proposes a "dynamic bridge," which is an automatically created transformation of a cross certificate topology, designed to reflect the same trust arrangements and constraints, but in a simplified structure. Creation of dynamic bridges should not require centralized coordination or infrastructure, and use of them for speed-enhanced validation should require clients to implement only a subset of standard path discovery and validation logic.

Problem Background

The standards that govern PKI, primarily [X509] and [RFC3280], envision a mechanism for a recipient, or "relying party" in one PKI domain to accept credentials from a sender or signer in another. The recipient has one or more "trust anchors." These are certificate authorities ("CAs") that the recipient trusts completely. These CAs can create cross-certificates, which indicate other CAs that this CA trusts. This process can be repeated multiple times, resulting in a chain of trust from the trust anchor to the sender's certificate. [pathbuild]

This process involves three distinct areas: "path discovery," "object location," and "path validation."

Path discovery entails finding the possible chain(s) of certificates between the sender and the trust anchor(s). Path discovery would be challenging enough even if all the certificates in the world were immediately available to select from. However, generally the only inputs to path discovery are the end-points: the sender's certificate, available because it is included with the signed message being validated, and the trust-anchor(s), which are part of the relying party's configuration. Path discovery therefore involves an iterative process sniffing out each "next possible link" – building the chain one link at a time.

Object location is the challenge of retrieving the certificates and cross certificates needed to feed the path discovery algorithm. Object location suffers from competition between several different mechanisms, none of which are very mature, and each of which assume they are the global solution. The separate mechanisms do not easily build upon each other. This challenge is not fatal, as the software of a relying party can support all of the contending mechanisms.

Path validation takes a candidate chain created by path discovery, and confirms that all the rules of trust transfer are followed within the chain. For example, cross-certificates can stipulate constraints that add requirements to the overall chain, or to parts of it some distance away from the certificate adding the constraint. Path validation checks all the constraints of each certificate against the others. Path validation also generally includes an "on-line status check" to confirm that no certificate in the chain has been revoked. Path validation is a computationally expensive process, but is well defined and reasonably well understood.

The focus of this paper is a scaling problem with path discovery.

Although developed independently, the concept is an extension of [Sun1], which envisions hierarchical root CAs issuing certificates directly to their n-level subordinates, thus flattening the hierarchical structure. The dynamic bridge extends this concept into the space of multi-domain PKIs linked by cross certificates, and handles the complexity of policy and constraint mapping introduced by such an extension.

In figure 1, a path discovery algorithm starts with the sender's certificate[1]. An object location mechanism should easily find CA 1, as CA 1's name is stated in the sender's certificate. The next step will be for the discovery algorithm to ask the location mechanism to find all certificates issued to CA 1. This could result in any number of certificates, indicated by the multiple green arrowheads around CA 1. As we have the picture already laid-out, we can see that the link to CA 2 is the correct choice. However, there is no way for the discovery algorithm to know this. It may well select the link that leads into cloud X, and one can imagine that if cloud X contains a large and complex mesh, it may be some time before the path discovery algorithm realizes it made a wrong turn at CA 1, and tries the alternative path leading to CA 2.

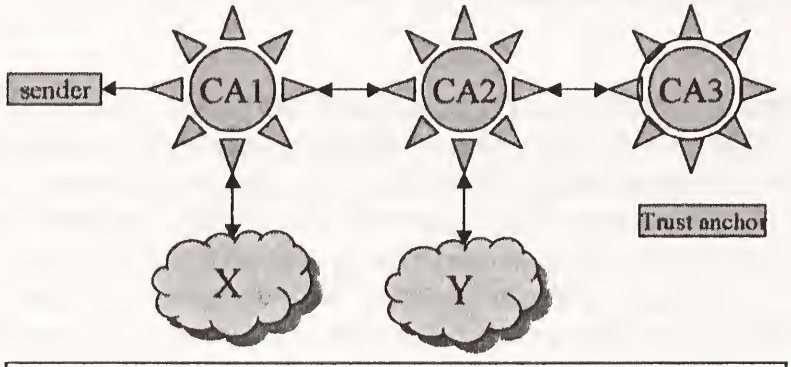


Figure 1

One response to this problem has been the implementation of PKI "bridge CAs." [pathbuild][FBCA]. A bridge CA is like a trust "hub"; it re-organizes the cross-certificate "mesh" topology into a star shape, which shortens the number of links in chains.

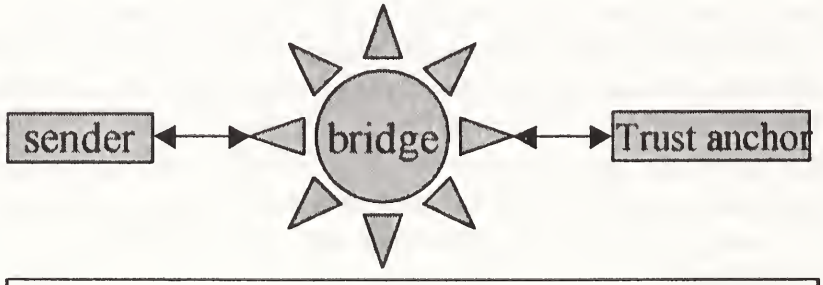


Figure 2

[1] There is an alternate technique [RFC3280] that begins at the trust anchor(s) and builds the path towards the sender. This difference is not relevant to the scaling problem that the example is building towards.

If an all-encompassing central bridge linked the entire world, there would be no path discovery problem. The cross-certificate from each trust anchor to the bridge would contain enough information to be located immediately from the bridge.

However, it has become clear that there will be no global bridge in the near future. No organization appears to have the combination of desire, funding, expertise, and ubiquitous acceptance that would be required. Rather, individual arenas are creating bridges that cover their natural scope. For example, the US Federal government has established the Federal Bridge CA [FBCA], Canada has a national bridge underway, and EDUCAUSE (a consortium of educational institutions) has created the Higher Education Bridge CA [HEBCA], etc.

These bridges are slowly being linked together. The result will likely be a cluster of large bridges, surrounded by a constellation of smaller bridges, surrounded by individual PKI domains. While this arrangement is an improvement on an unstructured mesh, in that the *average* path length will be lower, the problem discussed above in figure 1 remains. In-fact, in figure 1, if CA1 and CA2 were both bridges, the number of possible "wrong" routes would likely increase drastically, compared to individual CAs.

Basically, path discovery suffers from a lack of a "sense of direction." [PKIconcepts]

Solution Background

A common approach to resolve the path discovery sense of direction problem involves a process scanning the entire cross-certificate mesh, and pre-processing the results in some way to make discovery of a specific path faster once the actual endpoints are known.

However, there is a general complication to this technique. One cannot "cache" pre-validated *partial* paths. This is because of the ability of any certificate in a chain to add constraints applying to other (non-neighboring) certificates. Specifically, if a chain from CA1 ↔ CA2 ↔ CA3 is found, and is believed to be a common component to complete chains, it would make sense to cache it as an available component for building larger chains. However, once CA4 is added, CA4 may contain a property forbidden by a constraint in CA2, or vice-versa. As this is true all the way through to the end-user certificate, no partial path is safe from elimination by constraints from certificates not included in the partial path.

This is a complication rather than a fatal flaw as it can be resolved by careful separation of path discovery and path validation. In other words, partial paths can be cached as "discovered," so long as path validation is performed on the complete path once assembled, to make sure that constraints of the additional certificates are respected.

An example of a path discovery assistance system is the "intermediate store solution," envisioned and implemented in proof-of-concept by the FBCA's "path discovery and validation working group (PD-VAL)." [2]

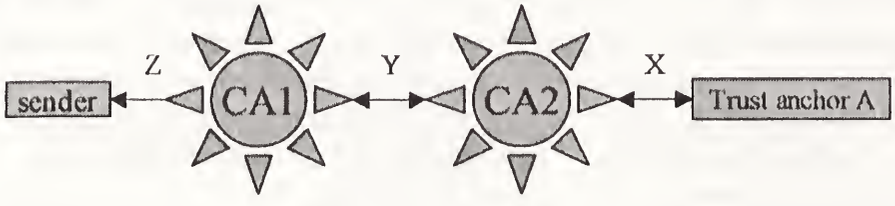
This intermediate store solution uses a "cross certificate spider" [HHSspider] to map out entire cross certificate topologies, and retrieve all the directly linked certificates and cross-certificates. This program essentially takes on the object location challenge, and retrieves all objects possibly needed to a server. The certificates are then stored into a distributable form (a PKCS7 file), and published to a web-server. A program running on end-user Microsoft workstations then regularly downloads the PKCS7 file, and adds its contents to the Microsoft Windows "intermediate store." This removes the iterative piece-by-piece part of path discovery, leaving only the problem of selecting a valid chain given the pre-collected universe of certificates, a capability that is built-in to most modern versions of Windows.[3]

This approach shows some promise, in homogenizing differences between versions of Windows (its original purpose), solving the object location problem, and simplifying the path discovery problem. [CAI-POC] However, like most pre-caching solutions, it requires proprietary software running on the desktop to utilize the cache – in this case, the program that downloads and installs the PKCS7 cache file.

Dynamic Bridge Concept

The idea of the dynamic bridge is to actively search out paths with a path length greater than 1 hop, and "condense" them by creating direct cross-certificates to reduce the path-length to one.

For example we consider the path

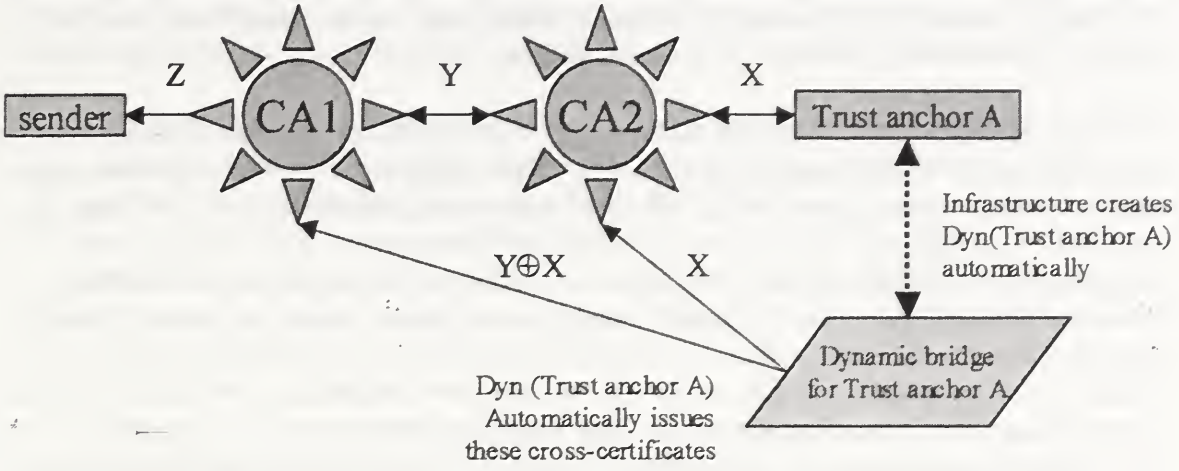


Here, X, Y, and Z are constraints, policies, policy mappings, and other attributes that effect a path's validity.

[2] "PD-VAL" was established by the FBCA Operational Authority to research challenges on bridge-aware path discovery. PD-VAL members include representatives from NIST, the Federal PKI technical working group, various Federal agencies, PKI vendors, and the author.

[3] Some versions of Windows do have the capability to perform iterative path discovery, but they rely on an object location solution that makes assumptions about certificate "AIA fields" that are not universally followed. By separating the object location function into the "cross-certificate spider," which supports multiple object location mechanisms, this removes the Windows AIA requirement.

Dynamic bridge infrastructure would transform the above situation as follows:



Specifically– A new CA, "Dyn(A)," with its own self-signed certificate, has been created, and that new CA has issued a series of unidirectional cross certificates. The new cross certificates reflect exactly the same trust arrangements between their subjects and Dyn(A) as the initial CA, but they have been "flattened" to combine intermediate hops. All trust is transferred directly from Dyn(A) to the targets with new 1-hop cross certificates.

This "condensation" of multiple hops into a single one is similar to [Sun1], except that rather than flattening hierarchical chains, the condensed hops here include cross-certificates between distinct PKI domains. This creates a significant new complication – trying to condense the constraints of the cross certificates.

In the case of the trust transfer from A -> CA2, with the constraint X, nothing has changed. However, in the case of the path A -> CA2 -> CA1 with the constraint X between A and CA2, and the constraint Y between CA2 and CA1, this path of length two has been reduced to a path of length one, directly from Dyn(A) to CA1, with the new constraint "Y⊕X", which is the ordered combination of constraints Y and X.

For example, if Y requires policy1, and X requires policy2, then Y⊕X would require both polices in a single constraint. If Y maps policy1 to policy2, and X maps policy2 to policy3, then Y⊕X map policy1 directly to policy3. It is asserted that all the constraints possible in cross certificates can be combined by a fully implemented ⊕ function, and that therefore the types of certificates and chains that can be condensed is not limited.

A dynamic bridge infrastructure would automatically create Dyn(A) by using a cross certificate spider to locate all paths that lead back to A. Once this process is completed, a user that previously used A as their trust anchor would change their trust anchor to Dyn(A). They would now find that all paths leading back to A are now immediately available directly from Dyn(A) with path length 1.

Furthermore, it is proposed that all cross certificates created with the ⊕ function also add a path length constraint indicating that at most one further hop is permitted. This would effectively prevent iterative path discovery. If a user uses only Dyn(A) as a trust anchor, then the process of path discovery is reduced to selecting the cross certificate from the issuer of the sender's certificate to the trust anchor.

Walking through a typical path discovery algorithm: in typical implementations, discovery starts with the sender's certificate, which contains the name of its issuer. An object location query is executed for all certificates whose subject matches the issuer of the sender's certificate. In this case, only CA1 will be returned. CA1 is not a trust anchor, so the algorithm iterates. An object location query is run for all certificates whose subject is CA1. This will return numerous certificates, including the one issued directly from the dynamic bridge. As the dynamic bridge is a trust anchor, path discovery is concluded. The key is that the path discovery algorithm never had to make a "guess" as to which certificate to select. The very first query that returned multiple certificates including a direct link to the trust anchor. This avoids the need for a "sense of direction."[4]

There is an interesting implication for revocation checking. The dynamic bridge path has "short circuited" CA2. A correctly implemented ⊕ function would ensure that CA2's constraints are respected, however in the original topology, CA2 also has the capability to revoke it's cross certificate to CA1, thus breaking the path. The dynamic bridge path does not pass through CA2, and thus removes CA's revocation capability.

There is a solution. The expiration date for the dynamic bridge's certificate from Dyn(A) to CA1 should be set to the earliest CRL "next update" time for any of the CA's that have been "flattened" from the path. i.e. The earliest CRL update time has become the cross certificate expiration time. In this way, a dynamic bridge cross certificate is valid only up until the time that a CRL update could have invalidated part of the consolidated path.

The dynamic bridge must continuously re-generate its cross certificates as they expire, and obviously will re-perform full path validation before re-consolidating paths, thus giving the intermediate CA's the opportunity to revoke paths.

Clearly the dynamic bridge is going to be a busy system. Its internal database conceivably consists of certificates for every CA in the world, it must continuously scan for new CA's (new paths to add), and perform the above re-validation of existing paths each time a CRL expires. However, a clever implementation could filter on changes to previous queries to detect additions, and queue re-validation carefully for only expiring paths. The assertion is made that this implementation is feasible.

---

[4] Note that it is possible that multiple paths existed through the original mesh between the sender's CA and the trust anchor. In this case, there would be multiple single-hop cross certificates from dynamic bridge to the issuer's CA. Although there is a "choice" as to which of these multiple paths will be selected, this does not change the claimed advantage. Whichever single hop cross certificate is selected by the path processor, it results in immediate path to the trust anchor with no further iteration, there is no "heading off in the wrong direction" regardless of which certificate is selected.

Proposed Benefits

It is believed that this arrangement would lead to benefits in each of the three primary areas of cross certificate-based trust building.

For path discovery, the need for iterative discovery is removed. The path length constraints added by the ⊕ function ensure that no "wild goose chases" will occur by the path discovery algorithm taking a wrong turn; as no "turns" are allowed.

For path validation, the process of validation has become simpler, both because path lengths are reduced, and because the cumulative effects of the chained constraints have been pre-calculated.

For object location, the dynamic bridge process has already collected and consolidated all the objects that the relying party will require. Assuming the dynamic bridge's cross-certificates are all stored in a single directory, the recipient's object location system needs only to be directed to that location.[5] If the dynamic bridge's object location mechanism supports multiple of the competing retrieval techniques (e.g. DN searching against X.500, AIA, LDAP referral trees, etc), the dynamic bridge's user's software need only support the mechanism that leads it to the dynamic bridge's consolidated repository.

Construction of a dynamic bridge requires no particular privilege, nor the cooperation of the mesh participants, other than CAs posting their cross-certificates into locations that the dynamic bridge's object location techniques can find. Any enterprise, or even end-users, can establish their own dynamic bridge(s). If multiple trust anchors are in use, either multiple independent dynamic bridges can be constructed (if selection as trust anchors must be separately selectable), or a dynamic bridge could merge multiple meshes by seeding the "condensing" process from multiple trust anchors.

While only those that utilize the dynamic bridge's trust anchor will receive the above benefits, the existence of the dynamic bridge is non-harmful to those that do not utilize it.

Finally, utilization of a dynamic bridge does not require any specialized software. Any standards-compliant path discovery system will be able to gain the advantages of a dynamic bridge just by re-selecting their trust anchor(s). In-fact, only a small subset of the standards-required capabilities are needed; some PKI software that is not fully compliant now (due to lack of iterative path discovery, or limited object location capabilities) would be made fully capable by using a dynamic bridge.

---

[5] Another interesting possibility is that the dynamic bridge could set AIA and/or SIA fields in the cross certificates that it generates. An example of a possible advantage -- the dynamic bridge will have multiple object location techniques that it may utilize when searching the cross certificate mesh, storing the technique that worked in an SIA field of the cross certificate could simplify object location of the target certificate for path building techniques that start with the trust anchor and work towards the sender.

Challenges and Unresolved Issues

Firstly, while the dynamic bridge does not create new key pairs[6], it does automatically create cross certificates which its users will trust. Frequently, CAs are kept "offline" to improve security, but due to the automated and continuous nature of its processing, the dynamic bridge must be "online.[7]"

Secondly, the entire concept hinges on the ⊕ function – the ability to take a series of constraints in separate certificates along a path and condense them into a single set of constraints, stored in a single X.509 compliant cross certificate constraint. Intuitively, this should be possible. A sample "permitted and forbidden sub-trees" algorithm is specified in [RFC3280], and although that algorithm is designed to be run iteratively during path construction, it should also be possible to run it during the cross certificate crawl. Furthermore, it appears that the constraints specification system is adequately expressive that transitive combinations of constraints can be simplified to a single constraint, but until ⊕ is successfully implemented, caution is needed.

Thirdly, Microsoft Window's build-in path discovery system ("CAPI") remains a challenge with respect to object location. CAPI does not support specification of a "default directory" or an "AIA[8] of last resort," which could be used to point to the dynamic bridge's directory. CAPI builds from the sender towards the trust anchor, so addition of AIA fields to dynamic bridge certificates does not help, as the sender's certificate's AIA will not lead to the relying party's dynamic bridge directory. Again, this could be overcome by loading the dynamic bridge output into the relying party's intermediate store, but this would require proprietary software on the desktop.

Finally, there is an issue with respect to the object location algorithm used by the spider process that builds the dynamic bridge. The crawl must start at the known trust anchor(s), and spread outwards. This direction is "the hard way" for object location.

When AIA fields are not present, the object location problem is generally solved via an LDAP search for DN's matching the subject field of the desired objects. When building a path from the sender's certificate towards the trust anchor(s), each certificate contains the DN of it's issuer, so the next possible steps can be queried by searching for certificates with the subject that matches the issuer of the current DN. However, certificates do not have an "issuee" field, so this technique cannot proceed in the opposite

---

[6] The creation of cross certificates involves signing an existing key with an existing key, it does not generate new key pairs.

[7] Technically, only a border directory containing the cross certificates must be fully on-line. The dynamic bridge must be able to push cross certificates onto its border directory, but other than this action, can be well isolated. While an "air gap" around CA's is "better," it is not unusual for production CA's to have a live one-way connection to their border directories.

[8] AIA stands for "authority information access," and in this context, gives a location to obtain all certificates whose subject matches the issuer of that certificate. AIA fields may be used by object location algorithms to obtain the "next step" in path discovery. An "AIA of last resort" specifies a general technique for finding any certificate's issuers given it's DN. Use of this type of mechanism is one of the competing object location solutions referred to in the first section.

direction. [X509] defines the "SIA" extension (the converse of the AIA field) for this purpose, but SIA is not widely populated.

There is a saving grace – bi-directional trust. When A issues a cross certificate to B, the matching reverse cross certificate is usually issued by B to A. The first iteration of the "subject that matches the issuer" technique will locate the reverse certificate, thus revealing the DN of the CA one step further from the trust anchor. The next iteration of the "subject that matches the issuer" technique will then return the forward cross certificate, allowing the crawl to spread outwards. This procedure has been shown to work in an implemented spider [HHSspider], but it does rely on bi-directional trust (or SIA fields) to discover the entire mesh.

Request for Feedback

Mitretek Systems is presenting the dynamic bridge concept to the PKI community without intent to assert patent protection, in hopes that its utility may be assessed, and discussions started concerning the possibility of implementation.

Those interested in providing feedback, or joining discussions on the topic, are asked to contact the paper's author, Mr. Ken Stillson of Mitretek Systems, at stillson@mitretek.org, or 703-610-2965. If there is sufficient interest, a mailing list or similar discussion mechanism will be established.

References

[CAI-POC] K. Stillson, *HHS CAI Proof-of-Concept Results*. Prepared by Mitretek Systems for the Dept. of Health and Human Services, Sept. 2003 [Available upon request from HHS]

[FBCA] CSRC/NIST *Federal Bridge Certification Authority* See http://csrc.nist.gov/pki/fbca/welcome.html

[HEBCA] Higher Education Bridge Certification Authority, web site. see http://www.educause.edu/hebca/

[HHSspider] K. Stillson *The Certificate Spider – A Simplified Technical Description* Mitretek Systems. June, 2003. Prepared for Mark Silverman of the department of Health and Human Services [Available upon request from HHS]

[Pathbuild] M. Cooper, Y. Dzambasow, P. Hesse, S. Joseph, R. Nicholas, *Internet X.509 Public Key Infrastructure: Certification Path Building* (Internet Draft) IETF PKIX Working Group. December 2003. See http://www.ietf.org/internet-drafts/draft-ietf-pkix-certpathbuild-03.txt

[PKIconcepts] K.D. Stillson *Public Key Infrastructure Interoperability: Tools and Concepts* The Telecommunications Review, Mitretek Systems, December 2002. See http://www.mitretek.org/publications/2002_telecomm_review/stillson_07.pdf

[PKIX] Stephen Kent, Tim Polk (co-chairs) *Public-Key Infrastructure IETF Working Group*. See http://www.ietf.org/html.charters/pkix-charter.html

[RFC3280] R. Housley, W. Polk, W. Ford, D. Solo *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* IETF Network Working Group. April 2002 http://www.faqs.org/rfcs/rfc3280.html

[Sun1] R. J. Perlman, Sun Microsystems, Inc *System and method for shortening certificate chains* U.S. Patent Office, application #20020147905 October 10, 2002. See http://appft1.uspto.gov/netahtml/PTO/srchnum.html (#20020147905)

[X509] ITU-T (formerly CCITT). *Recommendation X.509: The Directory—Public-Key and Attribute Certificate Frameworks.* 2000.

Identifying and Overcoming Obstacles to PKI Deployment and Usage

Stephen R. Hanna
Sun Microsystems, Inc.
steve.hanna@sun.com

Jean Pawluk
Inovant
jpawluk@visa.com

Abstract

Public Key Infrastructure (PKI) is a fundamental security technology used in many applications. Nevertheless, PKI deployment has been slow. Why? In June and August 2003, the OASIS PKI Technical Committee conducted two surveys aimed at identifying the top obstacles to PKI deployment and usage and soliciting suggestions for how these obstacles can be overcome. This paper presents the results of those surveys and summarizes the PKI Action Plan that the PKI TC has developed in response.

1. Introduction

Around the world, security threats are escalating and the demands that business and personal information be safeguarded are mounting. Business, governments, and consumers want access to their information in a mode that is easy to use, yet secure.

Public Key Infrastructure (PKI) is a fundamental security technology used in many applications to provide those security assurances. For a number of years, the promise of PKI has been challenged by its complexity and the costs of deployment.

The OASIS PKI Technical Committee was formed in January 2003 to tackle the issue of how to successfully deploy and use Public Key Infrastructure. As early adopters of PKI technology, many members of the committee have first-hand experience with the challenges of implementing PKI technology. As a result of their combined experiences, the committee decided that an impartial survey was needed to further identify the critical obstacles to widespread use of PKI.

A short, multiple-choice web-based survey was prepared and hosted on the group's web site in June 2003. Invitations to participate in the survey were distributed to standards and industry groups as well as security vendors and their customers around the globe.

After reviewing the June 2003 survey results [1], the OASIS PKI Technical Committee prepared a second survey to gather more detailed data about specific obstacles. This second survey was publicized to the participants in the original survey during August 2003 [2].

The data gathered through these surveys provides a clear view of the obstacles impeding PKI deployment and usage. The survey respondents also provided specific suggestions for addressing these obstacles with a clear consensus emerging from the many responses.

Based on this consensus, the OASIS PKI Technical Committee developed a PKI Action Plan [3] with five specific action items addressing the top five obstacles identified in the surveys. After several months of public review and comment, the committee has published the PKI Action Plan and begun implementation.

Implementing the plan will require cooperation from many parties: vendors, customers, standards groups, etc. If these groups can overcome their differences and work together, the obstacles to PKI deployment may be greatly reduced.

2. Review of Previous Work

For several years, starting in 1997, the "Year of PKI" was proclaimed by vendors selling the promise that public key infrastructure would revolutionize security by safeguarding electronic transactions. While PKI has been very successful in certain realms (secure web browsing), the full scope of these declarations is yet to be fulfilled.

According to the findings of Burton Group research originally published in 2001 and in late 2002 [4], progress in PKI deployment has been made over the past decade, but very slowly. "While public key security potential is vast, public key infrastructure (PKI) continues to struggle with interoperability,

complexity and application integration issues that slow customer adoption. PKI's sophistication hasn't translated into mass enterprise deployments."

The Burton Group researchers state that "The major applications using PKI today remain web-based authentication and virtual private networks (VPN), though the use of digital signature based electronic forms applications continues to grow".

They also stated, "Much of the complexity retarding PKI arises because a complete PKI requires multiple products from multiple vendors" including the PKI enabled application, a certificate authority vendor, a directory services vendor, and the vendor specific software for hardware clients and servers. A functional PKI may also include scenarios "that include smart cards and other cryptographic devices, professional services or system integration services, access management portals, certificate validation services... and more".

These concerns about PKI are reflected in numerous similar articles and papers in the trade press, conferences, and workshops [5], [6].

3. June 2003 Survey Results

In the June 2003 survey conducted by the OASIS PKI Technical Committee [1], the participants were asked to rate the importance of several common PKI applications and the importance of commonly cited obstacles to PKI deployment and usage. They were also asked to provide demographic information, which was used to check for survey bias and correlations between demographics and opinions. Finally, they were asked to list applications and obstacles missing from the survey.

3.1. Survey Sample

The June 2003 survey was open to anyone with an opinion on PKI obstacles, but aimed at people with expertise or experience in this area. Therefore, the survey invitations were sent to organizations and email discussion lists dedicated to PKI.

The 216 survey respondents were found to be a group of experienced group of industry professionals with serious PKI experience.

A large variety of job titles and functions were found among the respondents. Many of them had both technical and business functions included within their scope of their job duties. More than 75% of the respondents had at least 5 years of experience in Information Security / Privacy.

With over 90% of the respondents having either deployed or developed PKI software, they were very experienced with PKI. The majority of the participants were from the USA and Canada (60%) however over 30 countries were represented with many participants from Europe or Asia.

3.2. Analysis of Applications

Survey respondents were asked to rate various PKI applications as Most Important, Important, or Not Important to them. Respondents were also able to enter their own application area under Other (such as Identity Management, Non-Repudiation, and Document Encryption) and rate its importance.

For analysis, these ratings were combined into a weight by assigning 2 points for each respondent who rated an application Most Important and 1 point for each rating of Important. By computing these weights, the applications can be ranked by importance (as indicated by the respondents).

As shown in Table 1, most applications were found to be important but no one application stood out as the most important.

Applications	Most Important	Important	Not Important	No Answer	Weight	Weight Rank
Document Signing	43%	47%	6%	3%	1.38	1
Web Server Security	42%	48%	6%	4%	1.37	2
Secure Email	40%	46%	8%	6%	1.33	3
Web Services Security	34%	53%	9%	4%	1.26	4
Virtual Private Network	33%	50%	11%	6%	1.24	5
Electronic Commerce	34%	48%	13%	5%	1.22	6
Single Sign On	28%	56%	12%	4%	1.17	7
Secure Wireless LAN	25%	48%	19%	8%	1.06	8
Code Signing	20%	50%	22%	8%	0.98	9
Secure RPC	6%	40%	40%	13%	0.61	10
Other Application	9%	3%	7%	81%	0.21	11

Table 1: Application Weight Rank

Application weights are shown graphically in Figure 1.

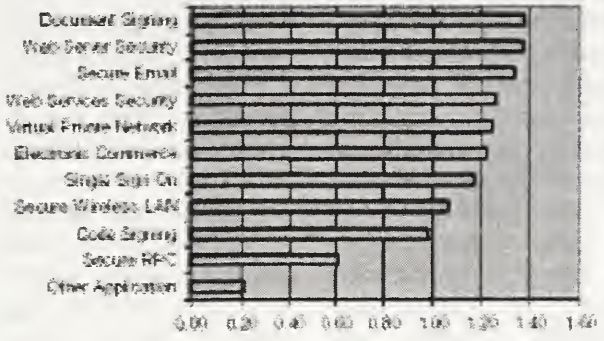


Figure 1 PKI Application Weights

These results affirm the view that PKI is a foundational technology used in many applications.

As business, governments and consumers all have different PKI needs, they also have different concerns about the importance of the listed applications. The survey results showed strong correlations between respondents' employment sector and their rating of applications. Government sector respondents ranked Document Signing 10% higher and Code Signing 11% lower than the total sample. In contrast, respondents in the Computer-related Manufacturing sector ranked Code Signing 12% higher than the total sample and Document Signing 10% lower. This is not surprising, since governments produce a lot more documents than code and computer firms typically do the opposite.

3.3. Analysis of Obstacles

In a manner similar to the rating of applications, respondents were presented with a list of possible obstacles to PKI deployment and usage and asked to rank each one as a Major Obstacle, a Minor Obstacle, or Not an Obstacle. Respondents were also able to describe an obstacle under Other and rate it in the same way.

Weights were computed by assigning 2 points to Major Obstacles and 1 point to Minor Obstacles. Using these weights, ranks were computed. The results are shown in Table 2.

The PKI Obstacles weight ranking is shown graphically in Figure 2.

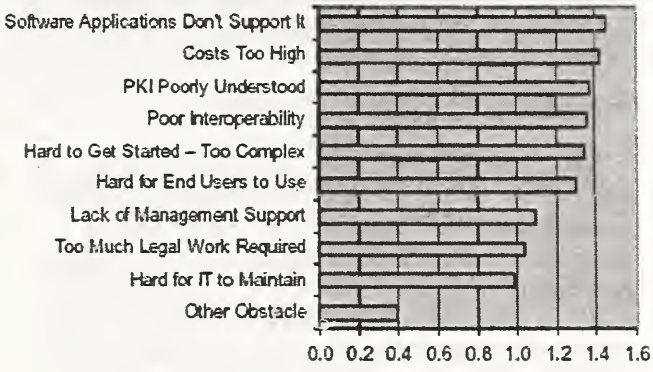


Figure 2 PKI Obstacle Weights

Many survey participants listed other obstacles to PKI deployment and usage. Here is a list of the obstacles that were cited by several respondents:

Insufficient ROI/business justification/need
Enrollment too complicated
Smart card problems (cost, driver and OS problems, readers rare)
Revocation hard
Standards (too many, incompatible, changing, poorly coordinated)
Too much focus on PKI technology, not enough on business need
No universal CA
Too complex
Insufficient skilled personnel
Poor implementations

Obstacles	Major Obstacle	Minor Obstacle	Not an Obstacle	No Answer	Total	Weight	Weight Rank
Software Applications Don't Support It	54%	33%	10%	3%	100%	1.45	1
Costs Too High	53%	34%	12%	2%	100%	1.42	2
PKI Poorly Understood	47%	41%	11%	1%	100%	1.37	3
Poor Interoperability	46%	39%	12%	3%	100%	1.35	4
Hard to Get Started – Too Complex	46%	39%	13%	2%	100%	1.34	5
Hard for End Users to Use	43%	42%	13%	3%	100%	1.30	6
Lack of Management Support	30%	44%	21%	5%	100%	1.09	7
Too Much Legal Work Required	25%	50%	22%	3%	100%	1.03	8
Hard for IT to Maintain	20%	55%	21%	4%	100%	0.99	9
Other Obstacle	18%	3%	5%	74%	100%	0.39	10

Table 2: PKI Obstacles Weight Rank

Unfortunately, the outcome of the survey question on obstacle ratings was inconclusive. Many obstacles had similar weights. Obstacles were broadly defined so it was not clear what respondents meant. In addition, several obstacles cited as Other Obstacles were noted by multiple respondents, indicating that the list of obstacles was incomplete. Therefore, the OASIS PKI Technical Committee Survey decided to conduct a followup survey to clarify the obstacles and ratings.

4. August 2003 Survey Results

The OASIS PKI Technical Committee's August 2003 survey [2] introduced a new points-based rating system that allowed respondents to clearly indicate priorities. It added "Other" obstacles cited by multiple participants in the June 2003 survey. It asked several questions designed to refine the broad categories used in the June survey. Moreover, it asked respondents to suggest ways that the obstacles could be addressed.

4.1. Survey Sample

The OASIS PKI Technical Committee sent invitations only to people who responded to the June 2003 Survey and provided an email address. This allowed us to use the previously gathered demographic data in analyzing the results while avoiding the need to ask for such data again. We found that the respondents to the August 2003 survey were similar in demographics and opinions to the earlier respondents.

4.2. Analysis of Obstacles

Instead of asking respondents to rate obstacles as a Major Obstacle, a Minor Obstacle, or Not an Obstacle, the August 2003 survey asked respondents to allocate 10 points among the obstacles listed, giving points to each item according to its importance. This allowed respondents to heavily weight items that were especially important to them. The results are shown in Table 3.

The point-based rankings reveal a substantial difference between the top five obstacles, which account for about 60% of the points, and the remaining ten obstacles. This does not mean that the lower-rated obstacles are not important. Most of them were rated as Most Important or Important by a majority of the respondents to the June 2003 survey. But the top five obstacles are just *more* important to the survey respondents.

The results were carefully checked for any sign that a small number of respondents might be skewing the results by throwing more votes than average to one item. This was not found to be true. In fact, the obstacle rankings were consistent across many demographic lines (experience, geography, industry sector, etc.). This was true for almost all opinions expressed in both surveys (except application ranking, as noted above).

Perhaps the most valuable part of the Follow-up Survey was the textual responses. For each of the top obstacles identified in the June 2003 Survey, respondents were asked to describe in their own words what causes these obstacles and what the PKI TC or others could do to address the obstacles. Certain themes were repeated over and over by many respondents. These themes pertain to several of the top obstacles. They are:

- Support for PKI is inconsistent. Often, it's missing from applications and operating systems. When present, it differs widely in what's supported. This increases cost and complexity substantially and makes interoperability a nightmare.

- Current PKI standards are inadequate. In some cases (as with certificate management), there are too many standards. In others (as with smart cards), there are too few. When present, the standards are too flexible and too complex. Because the standards are so

Obstacle	Average Points	Rank
Software Applications Don't Support It	1.76	1
Costs Too High	1.26	2
PKI Poorly Understood	1.06	3
Too Much Focus on Technology, Not Enough On Need	1.01	4
Poor Interoperability	.90	5
Hard to Get Started – Too Complex	.68	6
Lack of Management Support	.66	7
Hard for End Users to Use	.59	8
Enrollment Too Complicated	.35	9
Too Much Legal Work Required	.33	10
Smart Card Problems	.32	11
Hard for IT to Maintain	.30	12
Insufficient Need	.29	13
Revocation Hard	.25	14
Standards Problems	.25	15

Table 3: PKI Obstacles Point Rank

flexible and complex, implementations from different vendors rarely interoperate.

5. PKI Action Plan

The two surveys conducted in June and August 2003 allowed the OASIS PKI Technical Committee to identify the primary obstacles to PKI deployment and usage and to develop a PKI Action Plan [3] to address the obstacles. Here is a brief synopsis of that Action Plan.

5.1. Call for Industry-Wide Participation

The OASIS PKI Technical Committee recognizes that it cannot act independently in implementing this Action Plan. PKI involves many parties: customers and users, CA operators, software developers (for applications, PKI components, platforms, and libraries), industry and standards groups, lawyers, auditors, security experts, etc. This PKI Action Plan was developed based on input from all of these parties. The OASIS PKI Technical Committee calls on these parties to assist in its implementation.

5.2. Action Items

Develop Application Guidelines for PKI Use

For the three most popular PKI applications (Document Signing, Secure Email, and Electronic Commerce), specific guidelines should be developed describing how the standards should be used for this application. These guidelines should be simple and clear enough that if vendors and customers implement them properly, PKI interoperability can be achieved.

PKI TC members will contact application vendors, industry groups, and standards groups to determine whether such guidelines already exist and if not who could/should work on creating them. In some cases, standards may need to be created, merged or improved. If application guidelines already exist, the PKI TC will simply point them out.

Who: PKI TC Guidelines Subcommittee, Application Vendors, and Industry and Standards Groups

When: Spring 2004 for initial work

Increase Testing to Improve Interoperability

Provide conformance test suites, interoperability tests, and testing events for the three most popular applications (Document Signing, Secure Email, and Electronic Commerce) to improve interoperability. Certificate management protocols and smart card compatibility are also a concern. Branding and certification may be desirable. The PKI TC will work with organizations that have demonstrated involvement in or conduct of PKI interoperability testing or conformance testing to identify and encourage existing or new efforts in this area. Interoperability has many aspects. See the PKI Interoperability Framework white paper at http://www.pkiforum.org/whitepapers.html for details.

Who: PKI TC Testing Subcommittee with Industry and Standards Groups

When: Spring 2004 for initial work

Ask Application Vendors What They Need

OASIS PKI TC members will ask application vendors for the three most popular applications (Document Signing, Secure Email, and Electronic Commerce) to tell us what they need to provide better PKI support. Then we will explore how these needs (e.g. for quantified customer demand or good support libraries) can be met.

Who: PKI TC Ask Vendors Subcommittee, in cooperation with application vendors

When: Spring 2004 for initial work

Gather and Supplement Educational Materials on PKI

Explain in non-technical terms the benefits, value, ROI, and risk management effects of PKI. Include specific examples of PKI applications with real benefits and ROI. Also explain when PKI is appropriate (or not). Educational materials should be unbiased and freely available to all. If these materials already exist, the PKI TC will simply point them out. Otherwise, it will develop them in cooperation with others.

When: January – August 2004

Explore Ways to Lower Costs

Encourage the software development community (including the open source community) to provide options for organizations to conduct small pilots and tests of PKI functionality at reasonable costs–in effect reducing cost as a barrier to the use of PKI. Of course, operating a production PKI involves many costs other than software acquisition so an effort will be undertaken to gather and disseminate best practices for cost reduction in PKI deployments around the world.

Who: PKI TC Lower Costs Subcommittee, software development community, customers, etc.

When: Initial efforts in 2004

6. Conclusions

The results of the surveys conducted by the OASIS PKI Technical Committee identify the primary obstacles to PKI deployment and usage, as judged by the survey respondents. They also provide suggestions for addressing those obstacles.

Based on these results and on feedback from many PKI users, vendors, and other stakeholders, the OASIS PKI Technical Committee has prepared a PKI Action Plan to address the obstacles identified. Implementing the PKI Action Plan will be challenging but it provides some hope that PKI deployment will be easier and the benefits of PKI (strong and scalable security) will be widely realized.

7. Acknowledgments

Without the hard work and dedication of the members of the OASIS PKI Technical Committee, the results documented here would never have come to light. The survey respondents are thanked for their hard-won insights, which serve as a primary source for this paper. In addition, the PKI Action Plan reviewers and supporters are thanked for their assistance in hopes that it will lead to the successful completion of the PKI Action Plan.

Thanks to OASIS for granting permission for portions of the PKI Action Plan and survey analyses to be reproduced in this paper.

Thanks to The Burton Group for allowing us to quote one of their research reports.

8. References

[1] OASIS Public Key Infrastructure Technical Committee, "Analysis of June 2003 Survey on Obstacles to PKI Deployment and Usage", August 8, 2003
http://www.oasis-open.org/committees/pki/pkiobstaclesjune2003surveyreport.pdf

[2] OASIS Public Key Infrastructure Technical Committee, "Analysis of August 2003 Follow-up Survey on Obstacles to PKI Deployment and Usage", October 1,2003
http://www.oasis-open.org/committees/pki/pkiobstaclesaugust2003surveyreport.pdf

[3] OASIS Public Key Infrastructure Technical Committee, "PKI Action Plan", February 2004
http://www.oasis-open.org/committees/pki/pkiactionplan.pdf

[4] Dan Blum and Gerry Gebel, "Public Key Infrastructure: Making Progress, But Many Challenges Remain, V2", Directory and Security Strategies Research Report No. 612, Burton Group, Utah, February 13, 2003, pp. 5-7, http://www.burtongroup.com

[5] United States General Accounting Office, "Status of Federal Public Key Infrastructure Activities at Major Federal Departments and Agencies", (GAO-04-157), Washington D.C., December 2003
http://www.gao.gov/cgi-bin/getrpt?GAO-04-157

[6] Peter Gutmann, "PKI: It's Not Dead, Just Resting", IEEE Computer, August 2002, pp. 41-49

9. Copyright Notices