NISTIR 7427
ISBN 1-886843-60-0

6th Annual PKI R&D Workshop
"Applications-Driven PKI"
Proceedings

William T. Polk
Kent Seamons

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

NISTIR 7427
ISBN 1-886843-60-0

6th Annual PKI R&D Workshop *"Applications-Driven PKI"* Proceedings

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

Kent Seamons
Brigham Young University

September 2007

U.S. DEPARTMENT OF COMMERCE
Carlos M. Gutierrez, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
James M. Turner, Acting Director

Foreward

NIST hosted the sixth annual Public Key Infrastructure (PKI) Research & Development Workshop on April 17-19, 2007. The two and a half day event brought together computer security experts from academia, industry, and government to share progress, and explore challenges, in PKI and related technologies. In addition to the nine refereed papers, this proceedings captures the essence of the workshop activities including the keynote addresses, invited talks, panels, a birds-of-a-feather session and the popular informal rump session.

Continuing a trend from previous years, the workshop addressed an increasingly broad range of topics beyond PKI. The theory and practice of identity management took center stage for much of the workshop. In keeping with this year's theme, "Applications Driven PKI", practical applications were the focus of many presentations as well. And as always, several presentations describing innovations in infrastructure design prompted spirited discussions.

The identity management theme was evident throughout the workshop, beginning with the opening keynote presentation. Carl Ellison presented on behalf of Microsoft's Identity and Access Architect, Kim Cameron. Cameron's seven Laws of Identity provided an excellent foundation for several subsequent presentations. Carl's presentation was immediately followed by a panel on identity management systems, with a lively exploration of their commonalities and differences. The second keynote, by Ken Klingenstein, explored an "attribute ecosystem" where the academic community's Shibboleth project might play a central role. A later panel discussion explored the lessons learned in identity systems that leverage both the Security Association Markup Language (SAML) and PKI.

Applications, both deployed and in development, were also a common theme. Projects from the European Union and Argonne National Laboratories were the subject of refereed paper presentations. A panel from the mortgage industry describing PKI-based mortgage applications generated extensive discussions with its intersection of signature technologies and the legal implications. A Federal Government panel presented applications from the Treasury Department, Food and Drug Administration, and the Drug Enforcement Administration.

Two presentations explored the intersection between applications and identity management. Yet another Federal Government panel explored the deployment of a PKI-based identity management credential, the Personal Identity Verification (PIV) Card, and the opportunities and challenges this project presents for application developers. The rump session featured a complementary presentation on PKI in Radio Frequency Identification (RFID) passports. PKI is applied to these identity credentials in entirely different ways, yet share the overall goal of identifying a credential holder.

New approaches to infrastructure design were also presented in the workshop. Three presentations addressed PKI for Grids, including two presentations from the United Kingdom. Other presentations included “one time private keys,” a “temporal key release infrastructure,” and a system for delegating client authenticated web access through proxy certificates. These schemes strive to establish PKI-based analogs to common practices in other arenas, such as sealed bids at auctions or password-sharing for privilege delegation.

In all, the workshop included the presentation of two invited keynote presentations, nine referred papers, five panel discussions, a rump session, and a birds-of-a-feather session. Once again, the workshop was an international event with 125 attendees representing a cross-section of the global PKI community, with presenters from the USA, United Kingdom, Israel, Singapore, Brazil, and Canada. Due to the success of this event, a seventh workshop is planned for March 4-6, 2008. Given the broad interest in identity management techniques regardless of the underlying technology, next year’s workshop will be renamed to reflect the de facto shift in focus.

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

2007 PKI R&D Workshop
Applications-Driven PKI

Gaithersburg, Maryland USA

April 17-19, 2007

http://middleware.internet2.edu/pki07/

(Pre-proceedings were distributed at the workshop)

Organizers

General Chair: Ken Klingenstein, University of Colorado
Program Chair: Kent Seamons, Brigham Young University
Steering Committee Chair: Neal McBurnett, Internet2
Local Arrangements Chair: Tim Polk / Sara Caswell, NIST
Scribe: Ben Chinowsky, Internet2

Program Committee

Kent Seamons, *Brigham Young Univ.* (chair)	Eric Norman, *University of Wisconsin*
Peter Alterman, *National Institutes of Health*	Tim Polk, *NIST*
Bill Burr, *NIST*	Scott Rea, *Dartmouth College*
David Chadwick, *University of Kent*	John Sabo, *Computer Associates*
Joe Cohen, *Forum Systems*	Ravi Sandhu, *GMU and TriCipher*
Carl Ellison, *Microsoft*	Krishna Sankar, *Cisco Systems*
Stephen Farrell, *Trinity College Dublin*	Stefan Santesson, *Microsoft*
Richard Guida, *Johnson & Johnson*	Frank Siebenlist, *Argonne Nat'l Laboratory*
Peter Gutmann*, University of Auckland*	Sean Smith, *Dartmouth College*
Russ Housley, *Vigil Security, LLC*	Von Welch, *NCSA*
Neal McBurnett, *Internet2*	Stephen Whitlock*, Boeing*
Clifford Neuman, *USC-ISI*	Michael Wiener, *Cryptographic Clarity*

Archival Sites

PKI 2006:	http://middleware.internet2.edu/pki06
PKI 2005:	http://middleware.internet2.edu/pki05
PKI 2004:	http://middleware.internet2.edu/pki04
PKI 2003:	http://middleware.internet2.edu/pki03
PKI 2002:	http://www.cs.dartmouth.edu/~pki02

This page has been left intentionally blank.

6th Annual PKI R&D Workshop Summary

http://middleware.internet2.edu/pki07/proceedings/workshop_summary.html
Ben Chinowsky, Internet2

Note: this summary is organized topically rather than chronologically. See http://middleware.internet2.edu/pki07/proceedings/ for the workshop program, with links to papers and presentations.

The workshop looked at three aspects of its main theme of "applications-driven PKI":
I. Identity systems and federations
II. Current or imminent applications
III. Advanced approaches to infrastructure
There were also some additional talks not directly related to the workshop theme.

I. Identity systems and federations

There is a surge of interest in an "identity layer for the Internet", and a change in how identity is conceived in many quarters. The old joke goes that on the Internet no one knows you're a dog; identity is about giving you control over who -- if anyone -- gets to know that you're a dog, and if so, what kind of dog you are. This notion of *identity as attributes*, instead of identity as identifier, is gaining momentum for both privacy and security reasons.

In that spirit, Carl Ellison, now working with Identity and Access Architect Kim Cameron at Microsoft, and speaking on his behalf, keynoted on **Identity Management.** The Internet was built without much attention to security; various approaches to addressing this -- like proliferating per-vendor passwords and Microsoft Passport -- are more and more showing their limitations. Kim Cameron, through extensive discussions with a variety of parties concerned with solving this problem, has proposed seven Laws of Identity:
1. ***User Control and Consent.*** *Technical identity systems must only reveal information identifying a user with the user's consent.*
2. ***Minimal Disclosure for a Constrained Use.*** *The solution that discloses the least amount of identifying information and best limits its use is the most stable long-term solution.*
3. ***Justifiable Parties.*** *Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship.*
4. ***Directed Identity.*** *A universal identity system must support both "omni-directional" identifiers for use by public entities and "unidirectional" identifiers for use by private entities, thus facilitating discovery while preventing unnecessary release of correlation handles.*
5. ***Pluralism of Operators and Technologies.*** *A universal identity system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers.* (Ellison drew a contrast between this situation and DNS, which he suggested only worked because it was implemented before anyone noticed.)

6. ***Human Integration.*** *The universal identity metasystem must define the human user to be a component of the distributed system integrated through unambiguous human-machine communication mechanisms offering protection against identity attacks.* (Ellison has been advocating such mechanisms for years, under the label "ceremonies".)
7. ***Consistent Experience Across Contexts.*** *The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies.* (Here is where the card metaphor comes in; as Cameron notes, "we must 'thingify' digital identities -- make them into 'things' the user can see on the desktop, add and delete, select and share...How usable would today's computers be had we not invented icons and lists that consistently represent folders and documents? We must do the same with digital identities.")

A great deal of detail on Cameron's Laws of Identity is available at http://www.identityblog.com/?page_id=354.

The way to obey these laws is to build an "identity metasystem". This means not only creating a system (like Microsoft's CardSpace) for presenting choices of identity to the user and conveying identities to relying parties, but also creating a way for different such systems to interoperate.

Ellison's keynote was immediately followed by presentations on three more identity systems, and a panel discussion of the relationships among the various identity systems and the identity metasystem. Eric Norman moderated the discussion.

Mike Ozburn introduced **OpenID.** The main idea behind OpenID is to use a URL to identify the individual; OpenID is the result of a recent pooling of effort by four separate projects that had been taking similar approaches. This approach solves the namespace problem by leveraging DNS, creating a "single point of contact, single point of control" for each individual. Ozburn stressed that "OpenID is NOT a trust system...but it can be PART of one". OpenID already has millions of users and over a thousand sites enabled to use it. OpenID is currently used mostly for low-risk applications like blogs and social networking, not commerce, education, or government.

Mike McIntosh introduced the **Higgins project.** Higgins is an open-source identity framework and API which can accommodate CardSpace, OpenID and other protocols; IBM and Novell have prominent leadership roles in the project. Like CardSpace, Higgins uses a card metaphor for user identities, is designed with interoperability with other identity systems in mind, and includes elements of an identity metasystem. McIntosh noted that he is working closely with Kim Cameron, and cited Law of Identity #5 -- Pluralism of Operators and Technologies -- as key to the effort. McIntosh also noted that the Eclipse Public License will allow incorporation of Higgins into proprietary code.

Scott Cantor gave an overview of the **Security Assertion Markup Language (SAML)** space, including the history and current status of the Shibboleth and Liberty Alliance projects. Cantor's most emphatic point, however, was the danger of a new generation of web applications being tied to particular identity systems: "BrokenWeb 2.0". Cantor stressed that "it's just as wrong to bind an app to SAML or OpenID" as to bind it to passwords; in a correct, scalable architecture, applications trust the web server. But, despite the broad emerging consensus on the need for an identity metasystem, the construction of BrokenWeb 2.0 is already well underway.

Eric Norman underscored this concern in his informal definition of the concept of an "identity layer" for the Internet: "application developers shouldn't have to write code to do identity stuff." Several points were made about how to achieve this:

- There was general agreement that identity should be conceived as a collection of attributes, rather than identity being counterposed to attributes. Carl Ellison stressed the distinction between an authenticator and the attributes bound to that authenticator. In order for an identity metasystem to work, this distinction needs to be rigorously maintained; attributes and authenticators should not be combined as with e.g. credit card numbers.
- Scott Cantor observed that one huge problem is that users don't see identity as a collection of attributes, and don't understand the nature and privacy implications of many of the low-level attributes. Work on user interfaces that address this is underway, but still at an early stage.
- Cantor also noted that there are two reasons that deployers can't make the PKIX libraries work: the implementations are poor and the protocol is too complicated to begin with. IETF Chair Russ Housley agreed.
- Rich Guida noted that privacy concerns can be more readily addressed in intra-enterprise or enterprise-to-enterprise communications, and that there is great need for, and benefit in, deploying identity systems even with this more restricted scope. There are many complexities associated with the attribute-centric approach, which helps explain why it is not prevalent today. Simpler approaches, where identity is tied to an identifier such as an employee ID number, are much more commonplace and may be perfectly sufficient for use within or between enterprises. This approach is what much of Microsoft’s Active Directory framework is based on.
- There was a discussion of duplication vs. specialization among identity systems. Mike McIntosh argued that some systems are better for some purposes than others; Cantor argued that most of them can serve most purposes. Nonetheless, there was general agreement that a plurality of systems is inevitable, so that an identity layer / identity metasystem -- not an attempt to standardize on one identity system -- is the right approach.

Continuing with the theme of attribute-centricity, Ken Klingenstein's invited talk explored the concept of an **"attribute ecosystem".** Klingenstein envisions a central role for Shibboleth in this system: providing real-time transport from identity providers to service providers for authorization decisions. Other "compile-time" means will also be used to ship attributes to service providers, and intermediate entities such as proxies and portals, as well as to the identity provider itself. Klingenstein's slides present a variety of scenarios for how the pieces could fit together. The user needs to be able to manage all of this, which is a significant challenge; the Autograph tool developed by the Australian Meta Access Management System (MAMS) project is one attempt to meet that challenge.

Klingenstein wrapped up by observing that if this sounds like PKI, that's because what he's trying to create is PKI with a few more degrees of freedom.

Klingenstein, Georgia Marsh, and Jens Jensen presented experiences with federations in a panel discussion moderated by Scott Rea, who posed the question, "how is the mix of SAML / PKI / other working out?" Marsh described the approach of the General Services Administration's **eAuthentication** project as to use commercial off-the-shelf (COTS) technologies to grow e-government; ease of use is key to this effort. EAuth uses SAML for lower levels of assurance and PKI for higher levels, and is not limited to government-issued credentials; in fact, Marsh noted, "the government really does want to get out of the credential-issuing business". EAuth has been operational since October 2005; there are currently 46 relying parties and six credential service providers. Business aspects remain more challenging than technical aspects; lessons learned so far include "federate business not technology" and "align the data to the business process".

Jensen gave an overview of lessons learned in running a **Shibboleth federation in the UK.** He stressed the importance of policies in setting other federations' members' expectations; at the same time, keeping them consistent is difficult, as updating policy requires you to then prod everyone to update their procedures.

In his **global federations survey,** Klingenstein noted that the UK federation aims to encompass all of K-12, higher education, and continuing education -- a much broader scope than any US federation. Klingenstein also cited privacy guidelines from the UK (http://www.ukfederation.org.uk/library/uploads/Documents/recommendations-for-use-of-personal-data.pdf), including the mandatory provisions of the UK Data Protection Act. Klingenstein noted that federations are being rapidly adopted by collaborative applications, wikis in particular; his slides also list an impressive variety of other current and planned uses.

II. Current or imminent applications

Uri Resnitsky presented and demonstrated his **Directory-Enabled PKI Appliance.** The Appliance stores users' digital-signing keys and leads them through the signing process via a simple graphical interface; the signing key never leaves the Appliance. This is production technology; Resnitsky's paper provides details on its ongoing use in a variety of settings.

Nick Pope described the **application of OASIS Digital Signature Services (DSS) to e-invoicing in Europe.** The fact that the Value Added Tax (VAT) is applied at every stage of a commercial process, with rebates available for tax paid in a different jurisdiction, presents rich opportunities for fraud; here DSS is applied to stop such fraud. An implementation is in progress and is expected later this year. Pope noted that, although the specification has only recently been ratified, DSS is based on a style of operation already in use for years, e.g. in Thales SafeSign and the Norwegian BankID. See http://www.oasis-open.org/committees/dss/.

David Salbego described how Argonne National Laboratories combined Microsoft Certificate Services, KX.509 and Sun's Java Enterprise Suite to enable **certificate-based access to applications**. The resulting access manager has been open-sourced at http://www.opensso.dev.java.net/.

R.J. Schlecht introduced the **mortgage industry panel,** noting that PKI can be applied at several different steps in the mortgage-approval process. This process involves interaction among entities of widely differing resources, and is heavily regulated; the industry has been working on PKI for about four years. Yuriy Dzambasow introduced **SISAC** (http://www.sisac.org/), "a fully owned PKI subsidiary of the mortgage industry." SISAC certifies accredited issuing authorities to provide certificates to mortgage industry entities (not customers). Schlecht observed that pursuing consistency with FPKI has helped a lot, as parts of the mortgage industry are part of the government.

Francois Leblanc and Jim Bacchus introduced two users for SISAC certificates: **"eClosing and eVaulting"** and **"eNotary"** respectively. Leblanc noted that the goal of MERS, the Mortgage Electronic Registration System, is to register every mortgage loan in the US; they are currently at 80%. Bacchus noted that while the law requires you to notarize in person, the result of the process is an electronic document that can go in a digital safe deposit box.

The mortgage panel generated extensive discussion.
- Carl Ellison pointed out that signings are just "little point samples" of the extremely complicated human process of getting a mortgage, and expressed concern that if we automate the document process we'll end up throwing out the human process. Schlecht countered that the intent is not to change the human process, but to reduce the opportunity for error inherent in basing that process on paper documents, e.g. retyping things and losing things.
- John Sabo pointed out that in most signature-fraud cases, what is in dispute is not whether something was signed, but whether the signer was properly informed; he asked how technology could address this. There was general agreement that, while this is more a legal issue than a technical one, document signing could be helpful in creating auditable documentation that the signer was properly led through the process.
- Somewhat more prosaically, Leblanc noted that having electronic copies of documents often makes it possible for the customer to review them individually ahead of time, instead of being confronted with a mountain of documents all at once when visiting the mortgage office.

Peter Alterman led two panel discussions of PKI in the Federal Government. The first concerned current applications:
- Jim Schminky presented **the Treasury Department's Secure Extranet Gateway,** used for secure access to Treasury applications by business partners, remote Treasury users, and other Government agencies.
- Cindy Cullen discussed **SAFE digital signatures and the FDA Electronic Submissions Gateway (ESG).** Cullen noted that the FDA wants to get away from "semi trucks full of submissions" and is making a big push for submissions to be made

electronically. On the pharmaceutical-industry side, the Regulatory Affairs department of AstraZeneca has been involed in the ESG pilot.
- Alterman stood in for Chris Jewell in presenting the **Controlled Substances Ordering System (CSOS) at DEA.** CSOS has been a great success, with over 33,000 certificates issued and over two million line items ordered so far. Working closely with industry has been key to this success. See http://www.deaecom.gov/ for more information.

The second Federal PKI panel addressed issues around the August 2004 Homeland Security Presidential Directive 12 (HSPD-12), which mandates Personal Identity Verification (PIV) cards, i.e. smartcards, for Federal employees and contractors.
- Judith Spencer gave an **update on HSPD-12 implementation.** She noted that there have been many queries from states, industry, and foreign governments about how HSPD-12 applies to them. The short answer is that it doesn't, but many people want to be compatible with it anyway; FIPS 201 is seen as "the gold standard". All Federal employees with less than 15 years service are to have PIV cards by October 27, 2007; all Federal employees and contractors should have them by October 27, 2008. (In the mortgage panel, Jim Bacchus, a Marine Corps reserve officer, noted that he didn't use his smartcard for the first six years he had it, but since HSPD-12 he's had to use it every time he sends email.)
- Debb Blanchard gave an overview of the implementation of **HSPD-12 at the Veterans Administration.** They have issued about a thousand smart cards so far, and need to issue about 400,000 more before the October 2008 deadline.
- Tim Polk discussed **PIV-enabling applications,** noting that many applications -- both COTS and custom -- have limited compatibility. Polk's presentation noted "Six Deadly Sins" of PIV-Enabling: hardwiring to current cryptography modules, failing to allow for large certificates, overloading key-usage extensions, processing only the common name rather than the full name, assuming that a valid path means a valid user, and relying on a single type of certificate status information.

In the discussion, Peter Alterman (altermap@mail.nih.gov) asked the group for its help in documenting issues that are inadequately addressed in the Common Policy (http://www.cio.gov/fpkipa/documents/CommonPolicy.pdf). There was strong interest in seeing better documentation of PIV-enabling best (and worst) practices; Polk noted his certainty that the list of Deadly Sins will grow.

There was also a short rump-session presentation by Simon Godwin, discussing **PKI in RFID passports.** US passports have been completely redesigned to include RFID chips with biographical and biometric data (the latter is mostly the photo). Godwin noted that, in the US, "PKI as it relates to passports is really all about signing data" -- there is no document-specific keypair on the passport, just the public key used to sign the data on the chip. Sean Smith noted that Singapore is pursing a more advanced scheme, with passports carrying keypairs. Godwin also reassured the group that there are no plans to remove humans from the passport-inspection process.

III. Advanced approaches to infrastructure

Tan Teik Guan discussed **digital signatures via One Time Private Keys (OTPK).** This scheme builds on the practice, already common in Singapore and Hong Kong, of using one-time passwords for everyday banking transactions. With OTPK, a separate private key is created, used, and deleted for each signature; the key never leaves the client. A demo is at http://www.demo.com/demonstrators/demo2006fall/79808.php; a toolkit and pilot project are planned.

Wills, bids at auction, and bids for government contracts are examples of documents commonly put in sealed envelopes in order to ensure that they are not read until after a certain point in time. Observing that there is currently no electronic equivalent to this process, Ricardo Felipe Custódio introduced the proof-of-concept **Temporal Key Release Infrastructure.** Custódio also further developed the analogy with the sealed envelope, e.g. noting that TKRI provides a functional equivalent of a window in the envelope. Custódio's team currently has a working prototype.

Nicholas Santos discussed **Limited Delegation for Client-Side SSL.** As evidenced by the endemic practice of password-sharing, users really need a way to delegate their privileges to other users; Santos noted that this concept is well-understood everywhere except in traditional PKI. His solution, developed as a student of Sean Smith at Dartmouth, involves a non-standard use of X.509 proxy certificates, together with dynamically loadable modules in Mozilla Firefox -- and, unlike password sharing, allows delegation of a subset of privileges, not just all or none.

Paul Rabinovich made the case for **standardizing Kerberos names in X.509 certificates,** in order to facilitate their use for cross-domain authentication. He outlined four possible approaches to doing this, advocating the most vendor-neutral of the four. This talk generated lively discussion, including a suggestion that at least one of the approaches be written up as an RFC, but overall there was little support for Rabinovich's position. In particular, all four approaches were resoundingly rejected by Russ Housley, who argued that the time for standarding X.509 on Kerberos names has come and gone.

Three of the five rump-session talks also fell into the "advanced infrastructure" category:
- Doug Engert discussed **using PIV smartcards on Linux for authentication to Active Directory.** You can test this today; see http://opensc-project.org/.
- Olga Kornievskaia presented her work on **PKINIT,** which provides initial Kerberos authentication via X.509 certificates. This work is further described in standards-track RFC 4556, and CITI is working toward including it in MIT Kerberos 1.7. See http://citi.umich.edu/projects/pkinit/ for more information.
- Kent Seamons presented the work of his graduate student, Timothy van der Horst, on **Simple Authentication for the Web.** SAW is inspired by the common practice of using email to help a user who has forgotten their password. SAW introduces a variant of this approach as the primary means of authentication, resulting in a system that removes the

need for passwords at many web sites. The goals of this work are convenience and security. Complete details are available in a paper to be published at the *3rd International Conference on Security and Privacy in Communication Networks* in September 2007 (see http://isrl.cs.byu.edu/publications.php). The extension of this approach to IM and SMS was discussed at length in the Wednesday night BoF.

Also in the BoF, Massimiliano Pala discussed his work on **an OCSP-like protocol for PKI resource discovery.** See https://www.openca.org/projects/libprqp/ for more information.

There were three talks on PKI for Grids:
- Jens Jensen presented a **PKI for the UK National Grid Service,** complementary to but independent of the UK Shibboleth deployment. Jensen noted that the UK e-Science CA is the world's second-largest Grid CA, behind only the US Department of Energy Grid CA.
- Hoon Wei Lim outlined a **Certificate-free Grid Security Infrastructure Supporting Password-based User Authentication.** This work is a variation on the Gentry and Silverberg approach to identity-based cryptography, using IBC hierarchies that match the hierarchies of virtual organizations.
- Stephen Langella discussed **Enabling the Provisioning and Management of a Federated Grid Trust Fabric.** In Langella's work with the Cancer Biomedical Informatics Grid (caBIG), a major problem he encountered was how to know which CAs to trust. The approach developed to address this problem uses a single trusted CA to bootstrap the process of identifying other trustworthy CAs.

Organizational and miscellaneous

John Sabo introduced the **OASIS Identity and Trusted Infrastructure (IDtrust) Member Section,** formerly the PKI Member Section. IDtrust oversees the Enterprise Key Management Infrastructure (EKMI) Technical Committee, which is concerned with symmetric key management, and the PKI Adoption Committee. Sabo's slides note that "PKI is resurgent, driven by applications needing signatures, esp. for paperless transacting." See http://www.oasis-idtrust.org/.

Sara "Scout" Sinclair gave a short rump-session presentation on **the Dartmouth PKI Lab's planned PKI Census.** Where the OASIS survey focused on qualitative barriers to adoption, this will focus on quantifying the status of PKI as it exists today, and in particular on how many people are using it for each of its many current applications. Send questions you'd like included in the Census, and suggestions for people to send the Census form to, to geetha.wunnava@dartmouth.edu and scout.sinclair@dartmouth.edu.

Conclusion

PKI (more precisely, the distribution and use of X.509 digital certificates for authentication, digital signatures and encryption) is not happening the way we originally expected -- by reaching a sudden tipping point -- but it's happening nonetheless, via a patchwork of deployments, for a wide variety of purposes, taking a correspondingly wide variety of approaches. The broadening of this year's program to include discussion of identity systems and metasystems in general, reflects this turn of events.

The wrap-up discussion revealed authorization, delegation, and the delegation of authorization as additional areas of particular interest for next year's workshop. Kent Seamons noted that while technical paper submissions were up over last year, we still want a greater volume of submissions for next year. The submissions deadline is expected to be sometime in October.

In program committee discussions shortly after the workshop, it was agreed that the scope of next year's meeting will be broadened to "identity and trust". This will formalize the trend established at PKI07. Please join us at PKI08, March 4-6, 2008; watch http://middleware.internet2.edu/idtrust/ for details.

REFEREED PAPERS

The OASIS Digital Signing Service and its Application to E-Invoicing in Europe

Nick Pope – Thales e-Security, UK
(nick.pope@thales-esecurity.com)

Juan Carlos Cruellas - Universitat Politècnica de Catalunya, Spain
(cruellas@ac.upc.edu)

Nick and Juan Carlos co-chair the OASIS Digital Signature Services TC and
have work together over a number of years on European Electronic Signature standards
including most recently the ETSI specification on "Signatures for Digital Accounting"

Abstract

This paper presents two related activities, the first is the OASIS Digital Signature Services (DSS) standard, the second is the application of digital signatures to electronic invoicing as recognised under recent European legislation. DSS can be used to support a range of signature formats including the binary "cryptographic message syntax" and XML signatures, as well as related extended formats for "advanced electronic signatures" defined in European standards. The DSS standard is built around the general XML web based services structure and can be used with HTTP and SOAP transport protocols. The paper describes how DSS supports the needs of eInvoicing signature creation and verification, minimising the per user installation costs, improving security and reducing the need for revocation. It also describes how DSS verification greatly simplifies the complexity of user systems and facilitates centralised management of security within an organisation. Finally, the paper considers the requirements for maintaining the verifiability of signed invoices stored over a period of around 10 years and how this can be met by DSS verification services with time-stamping and / or archive services.

E-Invoicing in Europe

A directive was issued in 2001 with the aim of harmonising the requirements relating to "Value Added Tax" (VAT) in Europe [VATDirective]. This tax is a form purchase tax but is applicable to all sales including supplies of goods between companies to which value is added (hence the name value added tax). VAT legislation requires invoices be produced and recorded on all sales to which VAT is applicable and there are pan European rules on how this tax is itemised to facilitate auditing of the tax collection.

The recent directive on VAT harmonisation defines further rules for the form of VAT invoices, including requirements for the security of electronic invoices. It states that:

> *"Invoices sent by electronic means shall be accepted by Member States provided that the authenticity of the origin and integrity of the contents are guaranteed ..*"

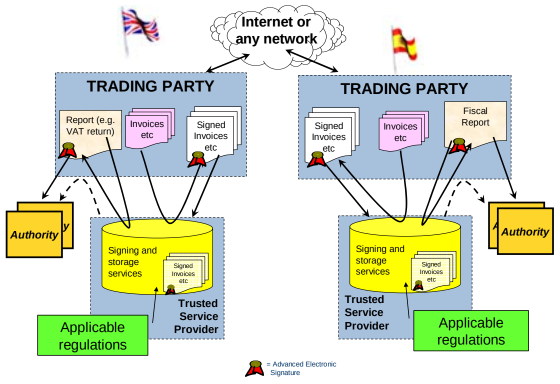
The VAT directive then goes on to identify alternative solutions to providing such protection including protection using a form of digital signature based on a PKI (referred to in EU legislation as an "advanced electronic signature").

Records of these signed e-invoices need to be kept for a number of years, varying from country to country, but can be up to 10 years or more. It has been generally taken that the requirement for authentication and integrity extends for this period. So it can be necessary for signatures to be verifiable 10 years after the invoice was created, possibly long after any public key certificates involved in their creation have expired or have been revoked.

Recognising the need to establish techniques for maintaining the validity of signatures over the long term, a European based workshop operated by CEN (Comité Européen de Normalisation) has included as part of its CEN Workshop Agreement on E-Invoices and Digital Signatures [CWA 15579] guidance on the maintenance of signatures over the long term. This identifies the need for controls to ensure that the information needed to verify the signature at the time of signing is maintained. This can be through use of technical measures such as the preservation of relevant certificates and revocation information such as OCSP (Online Certificate Status Protocol [RFC 2560]) or CRL (Certificate Revocation List [X.509]), along with a signature time-stamp, augmented by archive time-stamps where the algorithm strength dictates cannot be guaranteed for the whole storage. Alternatively, this can be through use of organisational measures using, for example, trusted notaries to check signatures and maintain records. It is recognised that a range of solutions exist depending more or less on cryptographic technology, use of archive media such as WORM drives, and organisational controls. This requirement is discussed further below.

Similar requirements can be applied not only to the e-invoices, but other documentation for company accounting and auditing, such as the quarterly regular VAT reports required by the tax authorities, as well as other company reports required to support needs for secure accounting such as those imposed through the Sarbanes-Oxley Act in the U.S.

The work of CEN has been taken further by ETSI (European Telecommunications Standards Institute) in a specification to be published shortly on “policy requirements for trust service providers signing and/or storing data for digital accounting” [ETSI TS 102 573]. This identifies best practice for third parties providing signing and storage of invoices and other accounting documents. It outlines the security controls that should be applied to ensure appropriate security on signing services and insure integrity of signed documents over the period of storage. It can be used as requirements external to an organisation, or internal services provides for use within large organisations. The model used in the ETSI standard is illustrated below:



Storage of Signed Invoices

The issue of particular concern when storing signed e-invoices is that when retrieved at a later date (say after 5 years) the signature may need to be verified by an auditor when looking back at old records. After such a period, it is likely that the certificates used in signing the invoice have expired and it is possible that one or more of the certificates used have been revoked. The auditor looking at an old signed document needs to be able to know that the signature was valid at the time the signature was applied.

Two basic solutions have been identified, The first is to use a trusted service that stores all signed documents along with a trusted statement that the signature was verified to be correct at the time when first placed in storage. Such a trusted service would require the use of secure databases or other forms of trusted storage and procedural controls to ensure that the appropriate checks are carried out when placing data in storage.

The second solution is to use technical measures to establish:

a) The time when a valid signature existed,
b) The certificates and revocation information (e.g. Certificate Revocation List or OCSP response) required to verify the correctness of the signature at that time.

This second solution allows an auditor to later verify the signature.

The time of when a valid signature is known to existed may achieved by verifying a signature just before first storing and marking the signature with this time.

The most widely accepted solution to getting assurance of the time is to use time-stamping produced by an server such as defined in RFC 3161. If applied over the signature this can be used to demonstrate that the signature occurred before the time-stamp. Further assurance of the signing time can be achieved by including the signing time within the signed data and then apply a time-stamp afterwards. However, from recent work on profiles for "advanced electronic signatures" based on a survey of current practice [ETSI TS 102 704 and ETSI TS 102 904] indicates that a time-stamp applied after signing is generally considered sufficient. It is recognised that other mechanisms, such as secure audit logs, can be used to prove the time that the document was signed / stored.

Possibly the surest way of making the certificate and revocation information available is by storing it with the signed document. This, however, can be very inefficient requiring significant amount of storage with much common information repeated across documents. The alternative of depending on the certification authority (CA) to store all the historical information and make it readily available for access for any date for 10 years is beyond the capabilities of most CAs.

The solution adopted in European "advanced" electronic signature format standards ([ETSI TS 101 733], [ETSI TS 101 903]) is to recommend that signatures are time-stamped on receipt, before placing in storage, and either the relevant certificates and revocation information is included with the signature by value or reference.

When storing documents for very long periods when the strength of the public key algorithm may not be assured (say longer than 10 years), additional protection needs to be applied. Over such periods the integrity of the signed document needs to be extended, for example, by additional signatures, signed time-stamps or secure storage mechanisms.

Since in Europe electronic documents are not generally required to be kept for longer than 10 years, this paper primarily considers the requirements for storage of documents in the 1 to 10 year time-scale. The mechanisms to protect documents for longer than ten years are not generally necessary for eInvoicing.

Application of Conventional PKI to E-Invoicing

The application of conventional PKI techniques to signing e-invoices stored for a period of around say 10 years is illustrated in the following diagram:

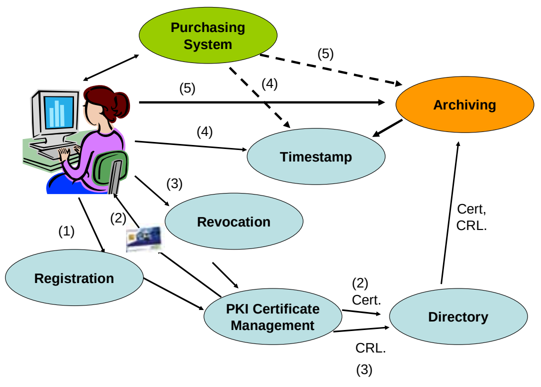


Fig 1: *Conventional PKI solution for e-Invoicing*

As illustrated each user needs to interact with several services to set up and apply digital signatures and maintain all the necessary records for e-invoicing documents produced by the purchasing system:

1) When first setting up the capability to create digital signatures, the user needs to interact with registration services which will carry out the necessary authentication and authorisation checks.
2) Some time later when the necessary checks have been completed the registration service interacts with the PKI management services to produce the necessary keys and certificates. The certificates are placed in a directory system and the key is passed back to the user, for example in a smart card device.
3) If after using the signing key the user changes role and is no longer authorised for signing, or the security of the key is compromised, for example due to loss of the smart card in a public place, the user needs to interact with revocation services to revoke use of the key.
4) When applying the signature in order to obtain a trusted indication of the time when the signature was applied the user (or purchasing system) timestamps the signature using a timestamp server.
5) In order to preserve the evidential value of the signed invoice the signed document is passed, either by the user or purchasing system, to the archiving system. The archiving system is required to maintain the integrity of the signature including the necessary CRLs and certificates which may be retrieved from the directory.

Some of the above steps may be simplified or handled by the purchasing application on behalf of the user (e.g. archiving and time-stamping). However, much of the complexities of the PKI system (registration, revocation handling) are inherent in the design of client based PKI.

In many European countries the need for digital (electronic) signatures is not limited to purchasing. Many of the accounting reports (for example quarterly tax returns summarising total VAT paid and collected, yearly corporate accounts) also have to be protected by signatures, widening the need for support for digital signatures.

DSS Based Solution

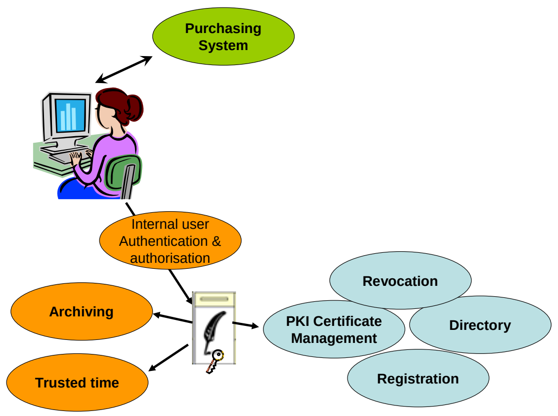

Purchasing System
Internal user Authentication & authorisation
Revocation
Archiving
PKI Certificate Management
Directory
Trusted time
Registration

Fig 2: *DSS Based Solution for e-Invoicing*

An alterative solution to signing company documents, such as e-invoices, would be to employ a signing server which signs document for authorised users within the organisation.

Such a server based solution can build on the existing user authentication and authorisation system to control the use of the signing function and, if individual user signing keys are required, identify the appropriate signing key. All the complexities of the PKI Management are handled by the server without the need for any user involvement. The signing service can, in addition, be extended to provide the necessary archiving functions to maintain the signatures over the lifetime of the document. Also, trusted time functions can be used to provide the necessary evidence of the signing time.

The OASIS DSS Protocol

The basic aim of the OASIS Digital Signature Service (DSS) draft standard [OASIS DSS] is to define protocols for a networked web service to support digital signatures. It also supports a variety of variations on basic digital signature services such as time-stamping.

DSS is designed to support a range of signature formats. Not only does DSS support the World Wide Web consortium XML Signature [W3C XMLDSig], but also the widely used Cryptographic Message Syntax (CMS) binary signed data format [IETF CMS]. It can even be extended to support other forms of signature such as PGP. The protocol is also designed to be easily extensible to enable support of advanced forms of CMS and XML based signatures such as defined by ETSI [ETSI TS 101 733] & [ETSI TS 101 903].

DSS supports two basic protocols one for the creation of digital signatures, the other for verification of signatures. The basic operation of a DSS sign and verify requests are illustrated below:

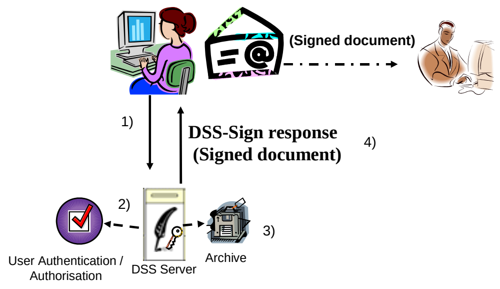


Fig 3: *DSS Sign Protocol*

1) The user sends the request for the document to be signed through a secure channel that authenticates the user (e.g. SSL + client authentication using one time password).
2) The server checks that the authenticated user is allowed to sign the document and if acceptable signs the document on behalf of the user with a corporate signing key or a key which the server holds on behalf of the user.
3) If required, the server can be extended to archive the document, signature and appropriate supporting information (CRLs, certificates, signing time).
4) The server returns the signed document to the user back through the same secure channel.

Having obtained the signed document from the DSS server the user can then pass it on to one or more recipients who may verify the signature themselves or use the DSS verify protocol.

The recipient may verify the signature himself or use the DSS verify protocol as indicated below:

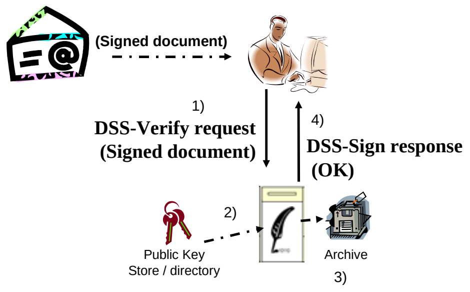


Fig 4 *DSS Verify protocol*

1) The user sends the request for the signed document to be verified through a secure channel (e.g. SSL).
2) The server verifies the validity of the signed document including checking the validity and revocation checks on any keys or certificates as necessary.
3) If required the verifying organisation may keep its own copy of the signed document and verification information (CRLs, Certificates, time at which document was verified to be correct).
4) The results of this verification is returned back to the user through the same secure channel.

The DSS protocol removes from the user all the burdens normally associated with digital signatures. There is no need for the management of large numbers of keys distributed throughout the organisation, and no special cryptographic code or keys are needed on the client system. Where it is necessary to authenticate the client existing mechanisms can be used. All the problems of maintaining the security of the keys and cryptographic functions associated with digital signatures can be managed by the organisation through centralised controls.

DSS servers can be used to maintain an audit record to confirm that signatures are verifiable at the time of receipt, and through use of time-stamping ensure that the validity of archived signed documents can be assured long after the applicable keys have expired as described below.

DSS specification set structure

The DSS specification set is formed by the so-called core document ("Digital Signature Service Core Protocols, Elements and Bindings") and a number of additional documents defining specific profiles of the aforementioned core protocols.

The core document defines the (XML-based) syntax and semantics for the basic services, namely: signature generation and signature verification. This includes:

- Definition of four basic messages: SignRequest, SignResponse, VerifyRequest and VerifyResponse. They are defined to easily manage the most common signatures formats, ie, [XMLSig] and [CMS].
- Definition of an extensibility mechanism that allows the clients to further qualify or even increase the extent of the requests through optional inputs. It also allows the servers to answer with extended responses through the corresponding optional outputs.
- Definition of a XML format for a time-stamp token, fully based on XML signatures as specified in [XMLSig].
- Definition of mechanisms for managing generation and verification of digital signatures carrying time-stamp tokens (both CMS-based as defined in [RFC 3161] and the XML-based specified in the core document itself) computed on the signatures themselves (signature time-stamps).
- Definition of bindings for transport and security. The first ones specify how DSS messages are encoded and carried over the most popular transport protocols (it defines bindings for HTTP –through HTTP POST exchanges- and SOAP 1.2). The security bindings establish rules for providing confidentiality, authentication and integrity to the transport binding; TLS 1.0 support is mandatory and SSL 3.0 support is optional. In this way clients may use wide-spread tools that do not jeopardize their implementation.

The profile documents further develop the basic messages so that they may be easily tailored to meet the requirements of a specific application or use case. Profiles may restrict the values ranges of certain message elements, or, if required, extend the basic core protocols defining new optional inputs, outputs and/or bindings.

The final result is not only a set of protocols targeting a number of relevant scenarios but also a set of generic protocols which may be easily further profiled as new uncovered use cases are identified.

Variations and Profiling DSS

The DSS protocol supports a number of variations in this protocol. For example, the signature may be passed back to the user on its own, detached from the document to which it applies, or placed within the document to which it applies. Another variation is that the document is reduced to a simple hash fingerprint for sending to the server instead of the document for either signing or verification, thereby reducing bandwidth requirements and reducing the opportunity for the confidentiality of the document to be compromised.

When signing a document the DSS server may add additional attributes or properties to the signature such as the claimed signing time or a time-stamp against the content applied immediately before signing.

Due to the number of variations a specific set of options can be selected in the DSS protocol to support a particular mode of operation or application requirement. This selection from the DSS protocol is defined in separate DSS profile specification. The DSS protocol is also designed to facilitate extensions and so DSS Profiles may also extend the protocol, as well as selecting specific options, defining its own profile specific input or outputs for profile specific attributes of a signature.

A number of profiles have been defined for DSS. This includes:

a) **Time-stamp profile**

As described above, including support for XML format time-stamps.

b) **DSS Entity Seal Profile**

This profile is a variation on a signed time-stamp, where the signed object includes not only the time but the identity of the authenticated user requesting the "entity seal". This provides further traceability and provides a form of "proxy" signature where the signature is produced on behalf of another identifiable party.

c) **Advanced Electronic Signature Profile**

This profile produces signatures that have the attributes needed for legally qualified and long-term signatures

d) **Code signing Profile**

This profile is designed to support the signing of code authorised for distribution with an organisational signature indicating its authenticity.

e) **Electronic (Digital) Post Mark Profile**

This profile is for providing an electronic post mark used confirm authenticity of email, as promoted by the Universal Postal Union [UPU-EPM].

f) **Signature Gateway Profile**

This profile supports the creation of signatures at a gateway from a form only recognised internally to a standard form which can be recognised externally.

Authentication and Authorisation for Signature Creation

The DSS services decouple the authentication / authorisation of the signing request from the authentication in the signature. This significantly simplifies the management of identities and authentication in the case of e-invoicing, where the signature is generally applied on behalf of a company, either as a corporate signature or as the signature of an individual who signs as a person responsible for the company, such as a chief executive.

The authentication required to authorise a signature request within an organisation, can be based upon internal security controls. Internal user identities can be assigned as part of the normal internal user authentication and authorisation controls, there is no need to interact with external registration services to set up each individual user that may be authorised to sign. Furthermore, where more complex work flow processes are involved with authorisation of invoices this process can be controlled independent of the application of the signature. Finally, any changes in personnel or removal of access rights, need not affect external revocation. Any authorisations to sign documents can be removed immediately without any impact on external revocation services.

If required the method employed for user authentication to a DSS need not involve any installation of security devices on the user PC. For example, simple challenge response systems using hardware tokens may be used to request signing by the DSS server through a simple web interface without the need for special security installation. Thus the common difficulties with installing security devices such as smart card readers can be avoided.

The centralised management of corporate signing keys are also facilitated through the use a signing server. Strict organisational controls can be applied to the server. If necessary it can be held in a physically secure area. Dual control / split keys can be applied so that the signing key can be used under strict controls. Thus the probability of compromise, and hence the need for external revocation, is minimised.

This ability in DSS to centralise signing capability not only improves security but also can reduce costs by minimising the per user installation costs.

Signature Verification for Stored Signatures

As mentioned above, to assure that signatures are verifiable for the period they are to be stored, it is considered necessary to establish the time that they are known to be valid and the certificate and revocation information (e.g. Certificate Revocation List, OCSP response) used to confirm that validation.

The DSS verification service can take the burden of obtaining and maintaining the supporting evidence for "long term" signatures away from the user. Two basic solutions are envisaged. One is to extend the signature structure adding the signature time-stamp and references / values of the certificates and revocation information employed in validating the signature (as described in [ETSI TS 101 733] and [ETSI TS 101 903]). The other is to include a trusted time and relevant certificate and revocation information in a secure audit log. In either case all the complexities are taken from the user and handled by a trusted server.

A further advantage of using a DSS server for signature verification is that all the complexities of validating the certificate path are taken from the user. This can be particularly onerous where multiple certificate policies are involved or the trusted root certificate authority of the organisation where the signature was created is different from that where the signature is verified. By placing such functionality in the server the appropriate cross domain policy controls can maintained and easily updated under central control.

In general the ability of DSS verification server to be placed under central control enables all the appropriate security measures to be applied and maintained. The security management authorities for an organisation can ensure that the procedures applied are secure and up to date. There is no need to depend on users to properly apply signature verification policy and there is no need to distribute up to date security software and information around the organisation.

DSS Within an eInvoicing Architecture

The use of servers for signing and verification of signatures fits naturally with the basic architecture of many e-invoicing systems. Generally, back office systems are used to handle invoices as part of the process flow. Whilst individuals may need to be accountable for the creation of invoices within the organisation, from the external viewpoint the signature belongs to the organisation, or in some countries a senior executive who represents the company.

As illustrated below in the case of invoices the creation of signatures may be initiated by an invoicing and accounting system which prepares and issues invoices under control of accounting clerks. The system is already trusted to properly maintain and control the creation of accounting information. The private key used in creating the invoice signatures can be managed centrally under clear security controls.

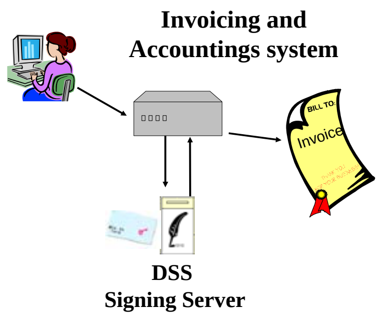


Fig 5 *eInvoicing System with DSS Signing*

Similarly, the verification of incoming invoices may be initiated by the accounting system. The information required for future verification of the stored invoice can be maintained in two ways. This can be done the use of a secure audit log maintained by the server containing the relevant validation information for later retrieval when subsequently re-verifying a stored document. Alternatively, by use of advanced electronic signature structures the document signature can be augmented with the information necessary to later re-verify the signature. In either case, the database used to store the invoices do not need to be secure to ensure the integrity of the signatures as this is provided through the DSS verification server.

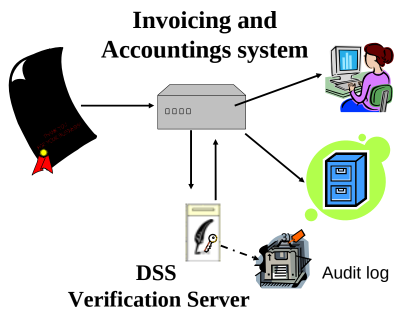


Fig 6 *eInvoicing System with DSS Verification*

DSS Implementations

The first version of the DSS core working draft dates back to October 2003. In 2004 and first half of 2005, the DSS Technical Committee developed a version of the core protocol incorporating most of the features of the current version, as well as the most important profiles. In 2006, the document went for public review. The TC received several comments that proved the attention attracted by his work. In parallel several members of the DSS TC have started an interoperability initiative for assessing the protocols under a practical perspective. At time of writing this paper specifications are in the final stage for ratification as OASIS specification early in 2007.

Over the last few years several systems have been deployed, which adopted DSS style of operation. As the specifications matured they attracted the attention an increasing number of manufactures of such a kind of systems. In the end, DSS specifications provide a standard way of operation for centralized services for electronic signatures generation and verification, which ensures interoperability.

2007 will likely start a period of extensive deployment of DSS-compliant applications. A number of organizations exist interested in providing centralized services for generation and verification of electronic signatures, which have decided to build a DSS-compliant application from the scratch. In addition, owners of platforms based on proprietary protocols are evolving towards DSS-compliant implementations.

One of the first major deployments using DSS specifications is the PSIS [PSIS]: a platform for identification and signature services, conceptualized, deployed and run by the Agència Catalana de Certificació (CATCERT) [CATCERT]. CATCERT is the CA for public administration agencies in Catalunya, Spain. Along with provision of different types of certificates (among which the personal certificate for Catalan citizens), CATCERT also offers this platform to Catalan governmental agencies, local administrations and private companies that have to securely exchange electronic information with them. This platform offers centralized services of signature generation, signature verification, encryption, and decryption. In addition to that CATCERT also provides access control tools (that use Liberty Alliance protocols) based on unique authentication or identity federation, to those organizations that want to integrate these services in their own applications. As for the DSS, this platform implements the DSS-core, the management of XML time-stamps, the DSS-AdES profile and the DSS time-stamping profile. It is able to perform semantic validation of certificates, CMS, XML-Sig XAdES and CAdES signatures, indicating their validity and the security level associated to the signing certificate (this is important because each type of electronic transaction with Catalan public administrations requires that the signing certificate has a pre-determined level of security).

In Norway a consortium of banks and CAs offer an optional lightweight web based signing, of a style similar to DSS, to over 600,000 banking customers [BankID] [EEMA-Award] with the aim to extend this to 2.3 million.

The UPU EPM, adopted by several postal service organisations, has been working closely with OASIS to incorporate DSS verification services in its global digital post mark system [UPU DPM].

Also in Spain the “Ministerio de Trabajo y Asuntos Sociales” (Ministery of Labour and Social Affaires) [MTAS], runs a centralized system that verifies digitally signed labour accidents reports. Within the framework on the currently on going initiative for a Spanish electronic ID card [DNIE], the “Ministerio de Administraciones Públicas” (Ministery for Public Administration) [MAP] also runs a platform offering, among others, a centralized service for electronic signatures validation. These two platforms were firstly developed using a proprietary protocol. Without no doubt, all these platforms deployed in different governmental agencies will have to evolve and become DSS-compliant for the sake of interoperability.

By the time this paper is written, the authors know of several commercial systems DSS-compliant that are offered to both private sector and public administrations all over Europe, which demonstrates the timeliness of the effort done by the DSS TC.

Conclusions

The work of OASIS in developing the standard for Digital Signature Services has provided a fruitful alternative to conventional client based PKI systems. The approach has been demonstrated to significantly reduce the cost of the per user installation, whilst features inherent in this approach can improve security.

The use of DSS signing servers have significant advantages. By detaching the authentication of internal end users from security of external keys requirements for revocation can be minimised. Also, by placing the server under central control proper administrative control can be applied to ensure the security of signing keys.

The use DSS verification servers provides straightforward verification of signed documents both on receipt and when retrieved from storage several years later. It can be used to remove the burden of complex certificate path validation from users, and maintain information required preservation of signatures over a period of up to around 10 years.

The features of DSS make it particularly suited to meeting the requirements of applications such as eInvoicing. DSS matches the need for invoice signing to be controlled on an organisation basis and handles the requirements for verification of stored signed documents.

The DSS specification is at its final stages of ratification. Interoperability trials have been run between separate implementations, and several major implementations are expected to appear over the next year.

Within Europe, and globally, there is significant interest in the use of web based and trusted third party services for the creation of digital signatures. Electronic invoicing is one of the major applications requiring digital signatures which is likely to be a major driver for a cost effective solution to digital signing.

By implementing DSS, the power of digital signatures can be provided without the headaches of installing PKI capabilities at every user system and ensuring signing keys and devices are managed securely.

Acknowledgement

The authors wish to acknowledge the significant contribution made to the members of the OASIS DSS Technical Committee and the ETSI Technical Committee on Electronic Signatures and Infrastructures in developing the ideas represented in this paper.

References

[VAT Directive] Council Directive 2001/115/EC of 20 December 2001 amending Directive 77/388/EEC with a view to simplifying, modernising and harmonising the conditions laid down for invoicing in respect of value added tax.
http://europa.eu.int/eur-lex/pri/en/oj/dat/2002/l_015/l_01520020117en00240028.pdf

[CWA 15579] CEN Workshop Agreement "E-Invoices and Digital Signatures" http://www.cenorm.be/CENORM/BusinessDomains/businessdomains/isss/cwa/electronic+business.asp

[OASIS DSS] OASIS Digital Signature Services Technical Committee http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dss

[BankID] Bank ID http://www.bankid.no

[EEMA-Award] The eema Award for Excellence in partnership with Infosecurity Today http://www.eema.org/static/isse/awards.htm

[ETSI ESI] ETSI Electronic Signatures and Infrastructures public home page http://portal.etsi.org/esi/el-sign.asp

[UPU-EPM] Electronic Post Mark http://www.globalepost.com/

[W3C XMLDSig] XML-Signature Syntax and Processing, W3C Recommendation 12 February 2002 http://www.w3.org/TR/xmldsig-core/

[IETF CMS] Cryptographic Message Syntax (CMS) IETF RFC 3852, R. Housley, July 2004

[RFC 2560] "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP", M. Myers et al, June 1999.

[ETSI TS 101 733] Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)

[ETSI TS 101 903]. XML Advanced Electronic Signatures (XAdES)

[ETSI TS 102 734] Profiles of CMS Advanced Electronic Signatures based on TS 101 733 (CadES)

[ETSI TS 102 904] Profiles of XML Advanced Electronic Signatures based on TS 101 904 (CadES)

[ETSI TS 102 573] policy requirements for trust service providers signing and/or storing data for digital accounting.

All ETSI documents are available from: http://pda.etsi.org/pda/queryform.asp

[X.509] ITU-T Recommendation X.509: Information Technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks (Latest version 08-2005)

[CATCERT] Agència Catalana de Certificació http://www.catcert.cat

[PSIS] Plataforma de serveis d'identificació i signatura http://www.catcert.cat/web/cat/1_4_3_plataforma.jsp

[MTAS] Ministerio de Asuntos Sociales. http://www.mtas.es

[MAP] Ministerio de Administraciones Públicas. http://www.map.es

[DNIE] DNI Electronico http://www.dnielectronico.es/ http://www.dnielectronico.es/seccion_aapp/cata.html

The Directory-Enabled PKI Appliance: Digital Signatures Made Simple, Approach and Real World Experience

Uri Resnitzky
Chief Scientist
Algorithmic Research (ARX)
uri@arx.com
http://www.arx.com

Abstract

We present a novel approach for a PKI based digital signature system for documents in an enterprise setting. A centralized appliance securely stores users' private signing keys. The appliance interfaces with the existing enterprise directory to automatically provision users' keys and certificates. Users authenticate to the appliance using their existing directory credentials in order to access their signing keys. Client applications send document hash values to the appliance to be signed therefore the signing keys themselves never leave the appliance. Streamlined user interface methods enable easy acceptance by users, while streamlined management enables minimal ongoing investment by IT staff. Real world experience with the described system is presented and shows successful deployment in a variety of organizations and markets.

1. Motivation and Related Work

In recent years the market demand for electronic signature solutions in the enterprise market has increased substantially (Gartner estimated in July 2006 between 5 to 20 percent current market penetration and less than 2 years for mainstream adaptation of electronic signatures. In July 2005 the estimate was 2-5 years [Gartner], while the 2004 report did not even include electronic signatures). This is due to increasing transitions from traditional pen and paper solutions to efficient paperless processing systems, as well as the advent of regulatory requirements in certain markets.

By and large traditional PKI has yet to deliver on its promise to fully answer these requirements in the mass market. Traditional PKI systems are based on distributing private keys to the end-users, which aside of security concerns [Marchesini], creates a high burden in logistics, cost, help-desk support and user acceptance [Whitten] and also introduces education obstacles [Nielsen]. Management of a large distributed system of any kind is extremely hard and PKI is no exception.

On the other hand, simplistic approaches for non-standard electronic signatures are increasingly being adopted [Gartner]. These solutions range from so-called click-wrap signatures and use of static signature images to proprietary keyed hash solutions. For many organizations expedited business processes, cost reduction and user-friendly systems – rather than the security concerns of signer authenticity, data integrity and non-repudiation – drive the decision to use electronic signatures [Gartner].

Unless low cost and easy to use and manage PKI-based systems are developed, the electronic signature market in general will leave PKI technology behind, or at best, PKI based systems will be deployed only by a relatively few enterprises that can afford the demanding costs.

Some related work aimed at making PKI systems easier to deploy and use has been presented in the past: A Plug and Play approach to PKI [Gutmann], Password Enabled PKI [Sandhu], Cryptographic Mobility Solutions [Gupta], Hardware secured Credential Repository [Lorch], Delegated Cryptography [Perrin] and putting CAs on the RA desk [Ellison].

Specifically for digital signatures, recent developments such as the OASIS Digital Signature Service (DSS) specification draft [OASIS] [Pope] - a specification for digital signature processing by web services - and the support in Adobe Acrobat 8.0 for so-called "Roaming Credential" servers [Landwehr] shows some promise.

We aim to make further advances in simplifying PKI deployments for digital signature purposes in enterprises of any size.

2. Design Criteria and Goals

Simplify the PKI problem domain by concentrating on digital signatures only and on deployments characterized by a well-known set of registered users defined in a directory (such as applications for internal enterprise employees as well as some business to consumer scenarios).

The system should *be as transparent and invisible to the end-user as possible* in order to increase user acceptance levels and reduce the need for training. There should be minimal or no direct end-user involvement in PKI-specific tasks (which may be difficult for end-users to perform) such as key generation, certificate enrollment or certificate renewal. The system should natively support file formats and applications users are already familiar with. Graphical User Interface (GUI) elements for signing and verification should be simplified. Graphical signature images should be used to enable the user to associate the digital signature with the traditional pen-and-paper signature.

Minimize the overhead of PKI management by not assuming or requiring that the administrator has background or training in PKI. Such assumptions may be incorrect, especially in small and medium-sized businesses, and may lead to deployment frustrations or increased manpower and training related costs. The administrator should not be required to perform ongoing PKI-related per-user or per-certificate management tasks. The system installation should be as painless as possible with defaults that cover most deployment scenarios.

The provided system should *contain all the required components*. There should be no need for additional 3rd party components such as a CA service contract, private key tokens, signature capture pads or electronic signature software / plug-ins.

3. Our Approach

Secure appliance with Central Storage of Signing Keys – The system is based on a secure, centrally installed, network-attached appliance that provides all the required features and manages the private signing keys and certificates. The signing keys in this system are RSA [RSA] private keys with modulus size ranging from 1024 to 4096 bits. The appliance meets the security requirements for Host Security Modules (HSMs) [FIPS140] and smartcards including a hardened operating system and tamper resistance. Signing keys and user certificates are stored in a database within the appliance and are encrypted using a key which is erased upon physical tampering of the appliance. Users can securely (see "Application Integration" below) access their signing keys from whatever computer they are working on, but the signing keys themselves are never exposed outside the appliance. In essence the secure repository can be regarded as a multi-user network-attached smartcard.

During a signature operation, once a user is authenticated (see the next section), the document's hash value (i.e., the result of applying the SHA-1 [SHA1] cryptographic hash function to the document's content) is sent to the appliance and is then signed within the appliance using the user's private key according to the PKCS#1 [PKCS1] specification. The resulting signature data is returned to the client and is used to build a Cryptographic Message Syntax [CMS] signature which is stored back into the document.

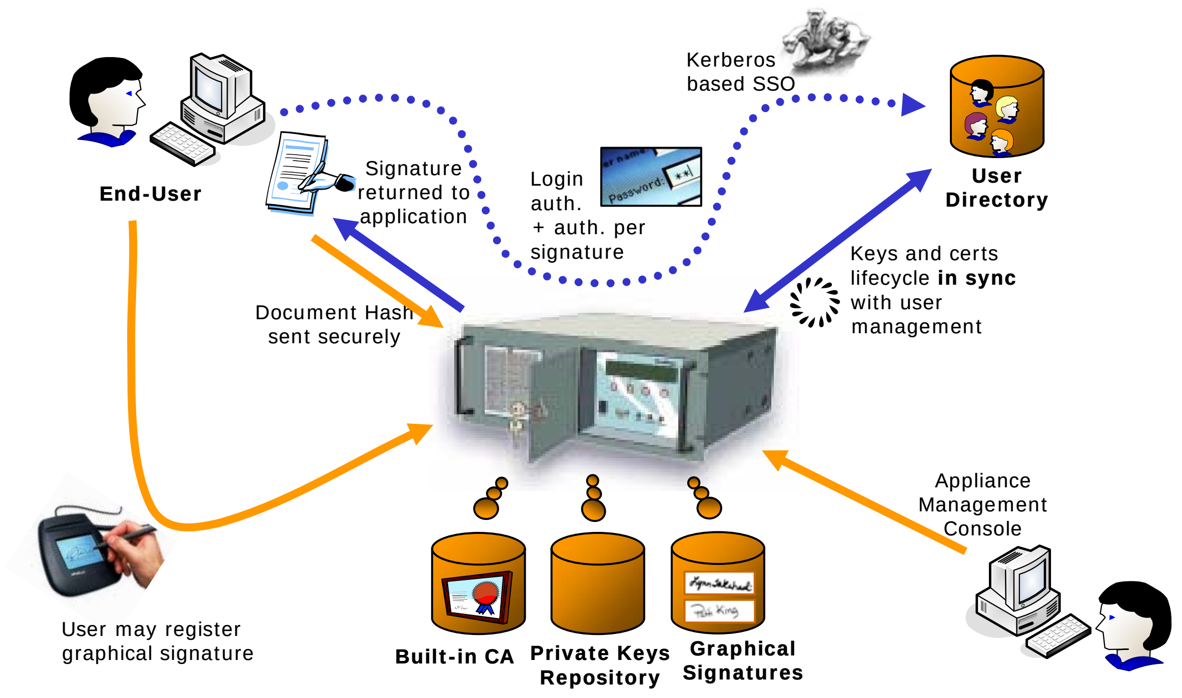

Kerberos based SSO
End-User
Signature returned to application
Login auth. + auth. per signature
User Directory
Keys and certs lifecycle **in sync** with user management
Document Hash sent securely
Appliance Management Console
User may register graphical signature
Built-in CA
Private Keys Repository
Graphical Signatures

Figure 1: Architecture

The system provides a per-signature audit log. This enables an auditor to track each and every usage of any of the private signing keys contained within the appliance including the date and time of signature, the name of the signer and the hash value signed.

Leverage existing enterprise authentication infrastructure – The pre-existing authentication system in use by the organization is used to enable users' access to their appliance stored signing keys. The user is prompted for their directory logon credentials which are then transmitted securely (see "Application Integration" below) to the appliance. The appliance verifies the credentials against the organization's existing directory servers using the LDAP [LDAP] protocol and grants access to the appropriate signing key accordingly. Secure LDAP authentication methods (such as LDAP over Transport Layer Security [TLS] and digest authentication) are used so as not to expose the user's credentials on the organization's network. In essence this approach means that the signing key is viewed as a resource on the enterprise network to which only the authorized owner has access. In cases where single-sign-on to the directory is possible (specifically when the appliance is installed within an existing Microsoft Active Directory environment which uses the Kerberos protocol [Kerberos]), there is no need to prompt the user for his credentials. Instead the Kerberos protocol is performed such that the user's identity is proven to the appliance using tickets granted by the directory server.

A policy setting, centrally enforced by the appliance, can be set to *require* users to manually re-enter their credentials each time they want to sign. These prompt-for-sign credentials are then securely transmitted to the appliance which validates them using secure LDAP authentication methods with the directory server. In addition, the appliance can be configured to use the Remote Authentication Dial in User Service [RADIUS] protocol in order to authenticate prompt-for-sign credentials with an additional external authentication server. This configuration is used to support a variety of authentication schemes such as two-factor authentication using one-time password devices.

Leverage existing enterprise directory and trust – User key generation, certificate enrollment, certificate renewal as well as revocation is automated based on data taken from and continuously synchronized with the organization's pre-existing user directory. The appliance includes a standalone built-in Certificate Authority which is initialized during appliance installation. The CA private RSA key is stored in the secure internal database. The CA root certificate is self-signed and includes subject name attributes which are defined during appliance installation. The appliance uses the LDAP protocol to periodically query the directory for users within a specific organizational unit (OU) which are members of a specific user group. These OU and group names are defined during appliance installation. Efficient query techniques are used to limit the load on the directory server. When a new user is detected the appliance retrieves the new user's information (common name and email address), generates a new RSA private-public key pair and issues an X.509 certificate using the built-in CA. The certificate is constructed using a built-in template (which defines the value of extensions such as the key usage) and includes the subject's common name and email as retrieved from the directory. Note that the administrator is not required to configure the above mentioned template, rather it has default values which are designed to produce a certificate which can be used for as many purposes as possible with the currently known PKI-aware applications. The newly issued certificate is stored, along with the private key, in the internal database. The appliance automatically refreshes soon-to-be-expired user certificates for users who are still part of the directory. When a change is made to the attributes of the user's directory object (e.g. an employee changing her surname), the appliance retrieves the updated information, revokes the existing certificate and issues a new certificate for the user, based on the updated information and the existing private key. When an existing user is removed from the directory (in most cases upon leaving the organization), the appliance revokes the user's certificate from the CA and deletes the user's records including the private signing key from the internal database. This provides immediate revocation of the key material preventing any risk of forged signatures and greatly simplifying one of the major hurdles in traditional distributed PKI namely the risk that compromised signing keys will continue to be used. All the directory-enabled certificate management operations described above are performed transparently without end-user involvement.

In essence this approach means that the directory administrator (usually working with the HR department) serves as the system's RA. We rely on the fact that organizations already have procedures and mechanisms in place to validate and control the trusted creation, modification and deletion of user objects and user attributes such as name and email in the directory.

Instead of the built-in CA, the appliance can be configured to communicate with an online CA service (using a proprietary web based protocol). In this case the same functionality is provided, with the difference being that the online CA enforces its certificate policy by validating the domain name used in the subject's email address as well as the organization name attribute in the certificate request. The online CA's policy delegates the verification of end-user identities and management of the credentials used to access the signing keys within the appliance to the customer organization's IT staff.

Publication into the directory of the CA root certificate, the CRL and users' certificates is also automatically and continuously performed. This enables smooth integration of PKI-aware directory enabled applications. This feature also enables automatic distribution of the CA root certificate into the trusted CA certificate stores of all clients in the Microsoft Active Directory domain, thus helping to automate trust in the CA root certificate within the organization.

Signing Documents in Native Format – The approach for producing signed documents follows the most popular file formats users work with everyday: Microsoft Word and Excel, Adobe PDF, TIFF and XML forms. Native file signing produces signed files which preserve the original file format and can be viewed by the native associated applications. Where possible, verification of the signature will be done using built-in support within the native PKI-aware application (for example the ubiquitous Adobe Reader, and the built-in signature validation ability of Microsoft Word XP and above). In order to enable legacy applications to produce digitally signed documents, a printer driver (i.e., a PDF distiller) is used that converts any data printed to it from any application into a signed-PDF document. Extensive use is made of the concept of Signature Fields which are visually distinct blocks added into the document which display the following data: signer's graphical signature image, signer's name (taken from the signer's certificate subject common name), signature date and time (in a variety of configurable display formats), signature reason (if entered) and the signature validation mark which indicates the validity status of the digital signature.

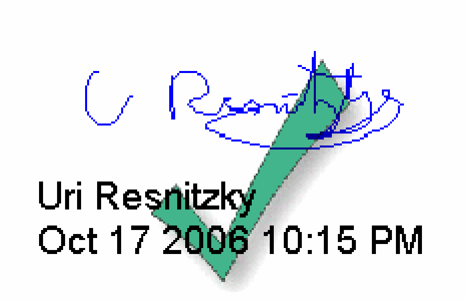


Figure 2: Signature Field Block

Streamlined GUI - Users may optionally register their graphical signature image (sometimes referred to as 'wet signature') using an electronic signature pad. The graphical signature image is securely stored within the appliance and is integrated into the signature block in native file signing. The combination of wet signature and digital signature provides a visual indication that the user is accustomed to enabling easy adaptation of the system by end-users. When using desktop applications to produce signed files, a streamlined user interface is presented to the user. In most scenarios it is sufficient to place the cursor in the desired location or drag and mark an area on the document and then click a single button to sign. After the appliance successfully generates the signature using the user's signing key, a signature field block will be inserted into the document at the specified location.

A signing ceremony is not required. However it can be configured to include the optional elements of entering prompt-for-sign credentials, and specifying a reason for signing. Signing pre-existing empty signature fields (inserted at design time into the document template or form) is similarly easy.

Users should be aware that the act of signing a file does not prevent the file from actually being modified either from within the native application (such as Word or Excel) or from the outside (using a hex editor for example). However such modifications (and even 'benign' modification such as updating a date/time field within a Word document) will result in signature validation failure. Dealing with these issues of limiting modification access rights to signed documents is in the domain of document management and archiving systems.

Streamlined Management - Management of the system requires minimal attention from administrators and is limited to system-wide tasks such as backup and restore of the appliance's encrypted database, secure loading of digitally signed firmware updates, modification of system-wide parameters and policy settings and download of audit logs. Management functions are only allowed for authenticated users that belong to a well-defined administrators group in the pre-existing organizational directory. Client software can be centrally deployed and configured by administrators. The client software detects and automatically connects to the appliance. This is achieved by searching the directory for an object created during appliance installation which contains the appliance's addressing information. Additional appliances can be added for load balancing and high availability with data replication between appliances secured using Internet Protocol Security [IPSEC]. The appliance's clock can be synchronized with the directory or an external time server using the Network Time Protocol [NTP]. It is used to provide the signature-timestamp authenticated attribute when building CMS signatures.

Application Integration – The appliance integrates with the signing applications using either client-side software, or using a web services interface.

The client software contains plug-ins and applications to support native file signing. The client software communicates with the appliance over a TLS secured channel using a proprietary protocol. It provides support for the standard cryptographic APIs (Microsoft's Cryptography API [CAPI], RSA's PKCS#11 [PKCS11] and Sun's JCA [JCA]) for seamless integration with PKI-aware applications (such as Microsoft Outlook and Adobe Acrobat). The client software also includes Signature API [SAPI]. This easy to use, high-level API is a wrapper over the lower-level cryptographic APIs and is intended to be used by applications that are not PKI-aware and do not or cannot interface with the more complex cryptographic APIs. SAPI is a native file format signature API supporting the concept of signature fields as well as graphic signature image handling. As an example, SAPI can be used by a custom workflow application to enumerate and validate all the signature fields in a document, and route or process the document according to a business logic based on the number or identity of the persons which signed the document.

The appliance can be accessed directly by applications using a web-services (WS) protocol. This protocol is a profile of the latest draft of the OASIS DSS core protocol specification and implements the full SAPI functionality direct from the appliance. When SAPI WS is used, no client component is needed, the entire document to be signed is sent to the appliance and the signed document is returned. In some cases (for example when signing PDF files) it is possible to save bandwidth by returning only the portions of the file which include the signature.

Signature Specific Features –The system provides a rich set of functionality specific to native digital-signature support for popular file formats. Within Microsoft Word and Excel it is possible to add *multiple* signature fields in order to support multiple levels of approvals. It is possible to require *dependence* of signature fields in order to make sure that clearing a signature field will 'break' the validity of dependent signatures. This supports hierarchical vs. side-by-side approvals. *Sectional* signing allows the user to specify that only the content of a specific cell-range or a document section is be signed. This has the advantage that other parts of the document can be updated or annotated within a workflow after the application of a signature without invalidation. The implementation uses these signature field attributes to decide which parts of the document content will be added to the hash value calculation.

4. Discussion

Note that due to the immediate revocation feature mentioned earlier, publication of a CRL by the built-in CA is not really needed. However a CRL is still published in order not to adversely affect existing functionality in PKI-aware client applications (for example the Adobe Reader enforces revocation checking by default). Because a CRL is published, and in order to limit its size, the built-in CA issues users certificates valid for one year only and then, as long as the user is still defined in the directory, automatically refreshes them.

It may be reasonable under the described approach to issue a single very long lived CRL once at system installation, and then to issue very long lived certificates for each user instead of renewing them each year.

The limitations and tradeoffs of our approach include the need to be online when signing documents. This has to be weighed against the resulting benefit of the immediate revocation capability.

Our system provides true user mobility which is independent on the installation of smartcard readers and does not use software keys that need to be exported and imported to new machines. In addition, the most insecure point in the system, namely the end-user's machine, does not contain any sensitive cryptographic keys, even in encrypted form, at any point during the use of the system. However the end-user's machine is still vulnerable to abuse by attacks which allow a capable intruder to generate arbitrary signatures (but not to duplicate the private signing key for offline signatures). These same vulnerabilities exist in any smartcard based PKI system.

Our centralized approach may present to a potential attacker an attractive online target which, if successfully attacked, can yield access to many users' keys. This means that the physical and logical security of the appliance and its computing environment must be sufficiently high to offset the risk. Further discussion on the subject of the security risks of centralization can be found in [Schneier] and [Perrin] which also covers some other relevant criticisms.

It should be noted that the system described here is most suitable for internal use within a single organization. This means that relying parties trust issues are limited to securely distributing the built-in CA’s root certificate following appliance installation to all machines within the organization. Existing IT management tools can easily support automating this task. In the case the online CA service is used, external trust is automatically achieved due to the fact that the online CA’s root certificate is already built into most client platforms trusted certificate stores.

In cases where relying parties exist outside the organization and the built-in CA is used, the organization needs to publish its built-in root CA certificate and relying parties must manually import it into their trusted root stores. The system is delivered with a web site template to help IT staff setup a web page for easy download and installation of their root CA certificate by outside relying parties.

5. Real World Experience

In this section we provide details of four real-world deployments of the system presented above. For each of these deployments we describe the target market, types of documents signed, and usage statistics. The statistics were collected from the appliance audit logs, in each case covering a period of at least 4 months of usage. The statistics include the number of actual signing users, the average and peak number of total signatures per working day and the average and peak signature rate per user. We assume working weeks have 5 days of 8 hours each, and that each year has 52 working weeks. Peek values are measured over a full working week.

5.1 Radiology Center

This Missouri based center performs a full range of outpatient diagnostic radiology studies for approximately 300 referring physicians. The center’s physicians use the system to digitally sign (including a graphical signature) transcribed medical reports that are then electronically sent to patients’ primary care physicians. This replaces the previously labor-intensive process of typing, printing, manually signing and faxing reports. Following an exam, a radiologist reviews a patient’s films and other images, compares them with previous studies if available, and dictates a report. An on-site medical transcriptionist types the report (typically in Microsoft Word) and stores it in a secure internal database. The radiologist can then retrieve the report to review and digitally sign as required. Regulations such as HIPAA dictate that healthcare organizations must implement a system that ensures that electronic records and signatures are trustworthy, reliable and secure. Note that in this case the PKI signature is used to maintain integrity within the radiology center’s own electronic record system. The electronically delivered reports are not verified by the receiving primary care physicians based on the digital signature, but rather by viewing the radiologist’s graphical signature image. Transition to electronic signatures enabled the center to reduce report turn-around time from two or three hours to approximately ten or fifteen minutes from dictation to electronic delivery. Labor costs decreased significantly and accounts receivable billing and reporting also improved. This deployment illustrates that the PKI system has practical value, even within very small installations. The PKI system was installed by the customer with phone support only. Note that such small organizations do not have significant in-house IT expertise. The successful installations and use of the system indicates that the goals of making a simple to manage and use PKI system were met.

Industry	Healthcare
Signing users	9

Average signatures per working day	95
Peak signatures per working day	153
Average signature rate per user	once every 45 working minutes
Peak signature rate per user	once every 5 working minutes

5.2 Bus Manufacturer

This leading North American bus manufacturing company employs about 100 engineers which handle more than 200 electronic change orders per week. The company's business is characterized by intensive customizations which require just-in-time design, engineering and manufacturing with a short lead time. Design engineers use CAD applications which render drawings in PDF format. Each PDF file is then digitally signed together with the engineer's professional stamp and graphical signature. Drawings are then electronically stored and managed by a Product Lifecycle Management (PLM) system. Previously, each drawing was plotted on paper, signed by hand and then scanned (using a manual, expensive-to-maintain scanner) back into electronic form to be stored in the PLM system. In some cases physical transportation between facilities was required to achieve the manual signing process. The biggest benefit of the installed system to the company is the process streamlining from design to document control to production, saving a lot of time, eliminating manual phases and increasing the productivity of the engineering workforce. While the investment in the new system was justified by the savings related to the scanning of drawings alone, the company is now better suited to face its main challenge of high throughput design / build-to-order. The installation of the PKI system, performed by local IT staff with phone support, was smooth and was followed by quick user acceptance reaching a usage level of over 100 signatures per day in under 2 weeks.

Industry	Engineering for Manufacturing
Signing users	98
Average signatures per working day	370
Peak signatures per working day	676
Average signature rate per user	once every 2.1 working hours
Peak signature rate per user	once every 6 working minutes

5.3 Clinical Trials Management

This company is a leading clinical technology services provider for the pharmaceutical industry, assisting drug manufacturers to setup systems to manage clinical trials. All 600 employees in three world-wide locations are using the system to sign large numbers of Word & PDF documents. These documents (such as Standard Operating Procedures and Audit Reports) are needed to demonstrate that the company's systems and operations are validated and quality controlled. The company is required to meet stringent industry standards such as GxP and the FDA Title 21 CFR Part 11 regulations for electronic records and electronic signatures. These regulations aim to ensure the accountability and data integrity of sensitive internal documents when moving from paper to electronic documents. The company undergoes between 40–50 customer audits annually to verify its adherence to those standards. The rapid deployment of the digital-signature system and its ease of use and tight integration with the existing user directory allowed it to be quickly adopted by company employees. Installation was achieved in a single day. Users were able to start signing documents right away and the signatures offered a look and feel that emulated the traditional 'wet-signatures' people were comfortable with. To meet specific requirements in the FDA's Title 21 CFR Part 11, users are required to enter both user name and password each time they want to sign and in addition add their reason for signing. In some locations the system was implemented using a thin client approach – the end users remotely login into a Citrix server located at the HQ data center and sign documents directly on the HQ network. As a result of the deployment the approval process for new SOPs was expedited from 2 weeks to less than an hour. In the past, getting signatures from 3 people in 3 different offices around the world required the use of fax and courier services which are no longer needed. Electronically signing documents also reduced the need for every document to be printed, filed, microfilmed and archived. Lost or misfiled documents are no longer a problem, saving the company considerable time and money. The physical archive was replaced with an electronic document repository. Signed documents can be viewed and validated for long periods into the future as the validation uses the time when the signature was made (and not the current time) when computing the validity of the signer's certificates. However, we recognize that this does not address long term archival and cryptographic issues related to encryption algorithm aging, new analytical attacks being discovered, or evolution of storage technologies etc, which are outside the scope of our work.

Industry	Life-Sciences
Signing users	520
Average signatures per working day	72
Peak signatures per working day	133
Average signature rate per user	once every 3.5 working days
Peak signature rate per user	once every 56 working minutes

5.4 Analytical Laboratory

This company is one of the largest privately owned analytical laboratory network in North America. Their diverse range of high-quality analytical testing and consultation expertise support numerous industries including environmental sciences (water, air, soil, waste and toxicity testing), petroleum testing and field sampling services, food safety (food chemistry and nutritional labeling, veterinary drug residues and inspection), and forensics (human drug and alcohol testing, DNA, paternity and genetic identification). Approximately 100 project managers (located in 15 different labs) are using the system to electronically sign Laboratory Information Management System (LIMS) reports and Certificates of Authority in PDF, Word and Excel formats. The labs have over 1,500,000 samples tested every year in a wide spectrum of applications. With the digital signatures system implemented, signed reports can be submitted to clients as soon as the results are available in a compliant manner and as electronic evidence in a court of law. Previously the lab employees had to print, hand sign, fax, mail and archive hard copy documents associated with the paper-based processes. The labs increased their competitive advantage by decreasing the time it takes to submit reports to clients (from 1-3 business days with a paper process to immediately available using an electronic process). The lab is using dual appliances in high availability configuration. In addition, SAPI was used to directly integrate report file signing into their LIMS system. Since signed documents need to be validated outside the organization, the lab had set up their system so that the CRL is published to an externally accessible web address (instead to the default location on their internal directory). The lab's root CA certificate was also published to an externally accessible web address along with instructions to clients on how to install this certificate in their local trusted root certificates store. It should be noted that since reports in this system are securely delivered to clients using a web portal, the lab's clients themselves are not necessarily concerned with validating the signatures. However the lab needs to protect itself from a scenario in which external parties may want to change a report to suit their needs. In this case the ability for stand-alone verification of a signed document outside of the lab's system is important for dispute resolution.

Industry	Sciences
Signing users	110
Average signatures per working day	1180
Peak signatures per working day	1400
Average signature rate per user	once every 45 working minutes
Peak signature rate per user	once every 5 working minutes

5.5 Environmental Impact

We calculate the weight of paper saved on a yearly basis in the above four deployments assuming each signature saves the printing of one standard letter-size sheet of paper. This results in an annual saving of 4480 lb (2030 kg) of paper.

Please note that the cost saving associated with the paper alone is very small compared with the other savings and benefits introduced with the digital signature system.

6. Conclusions and Further Work

In this paper we have presented a market-driven approach that enables the use of PKI technology for driving the adoption of electronically-signed documents. We have shown how this approach is successfully deployed in the field by diverse organizations. We believe that wide spread use of PKI for electronic signatures is at hand using the approach outlined here and that every effort should be made to continue to bridge the gap between PKI technology and the mass market.

7. Acknowledgements

Thanks are due to David Chadwick and the anonymous referees for their thoughtful comments and the patient support received while drafting this paper.

8. References

[CAPI] R. Coleridge, "The Cryptography API, or How to Keep a Secret", http://msdn2.microsoft.com/en-us/library/ms867086.aspx, August 1996.

[CMS] R. Housley, "Cryptographic Message Syntax", IETF RFC 3852, http://www.ietf.org/rfc/rfc3852.txt, July 2004.

[Ellison] C. Ellison, "Improvements on Conventional PKI Wisdom", Proceedings of the 1st Annual PKI Research Workshop, pp. 165-176, August 2002.

[FIPS140] National Institute of Standards and Technology (NIST), "*FIPS Publication 140-2: Security Requirements for Cryptographic Modules*", http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf, May 2001.

[Gartner] V. Wheatman et al, "Hype Cycle for Information Security", Gartner RAS Core Research Note G00139428 http://www.gartner.com/DisplayDocument?doc_cd=139428, July 2006.

[Gupta] S. Gupta, "Security Characteristics of Cryptographic Mobility Solutions", Proceedings of the 1st Annual PKI Research Workshop, pp. 117-126, August 2002.

[Gutmann] P. Gutmann, "Plug-and-Play PKI: A PKI your Mother can Use", Proceedings of the 12th USENIX Security Symposium, pp. 45-58, August 2003.

[IPSEC] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", IETF RFC 2401, http://www.ietf.org/rfc/rfc2401.txt, November 1998.

[JCA] Sun Microsystems, Inc., "Java Cryptography Architecture, API Specification & Reference", http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html, August 2002.

[Kerberos] Microsoft Corp., "Windows 2000 Kerberos Authentication", http://www.microsoft.com/technet/prodtechnol/windows2000serv/deploy/confeat/kerberos.mspx

[Landwehr] J. Landwehr, "Making digital signatures easier to use and deploy with roaming credentials", Adobe Security Matters Blog, http://blogs.adobe.com/security/2006/09/making_digital_signatures_easi.html, September 2006.

[LDAP] K. Zeilenga, "Lightweight Directory Access Protocol", IETF RFC 4510, http://www.ietf.org/rfc/rfc4510.txt, June 2006.

[Lorch] M. Lorch, J. Basney and D. Kafura, "A Hardware-secured Credential Repository for Grid PKIs", 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 640-647, April 2004.

[Marchesini] J. Marchesini, S.W. Smith, M. Zhao, "Keyjacking: Risks of the Current Client-side Infrastructure", Proceedings of the 2nd Annual PKI Research Workshop, pp. 128-144, April 2003.

[Nielsen] R. Nielsen, "Observations from the Deployment of a Large Scale PKI", Proceedings of the 4th Annual PKI Research Workshop, pp. 159-165, August 2005.

[NTP] D. L. Mills, "Network Time Protocol", IETF RFC 1305, http://www.ietf.org/rfc/rfc1305.txt, March 1992.

[OASIS] S. Drees et al, "Digital Signature Service Core Protocols, Elements, and Bindings", OASIS Digital Signature Services Technical Committee Draft, http://docs.oasis-open.org/dss/v1.0/oasis-dss-1.0-core-spec-cd-r5.pdf, August 2006.

[PKCS1] RSA Laboratories, "PKCS #1 v.21: RSA Cryptography Standard", ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf, June 2002.

[PKCS11] RSA Laboratories, "PKCS #11 v2.20: Cryptographic Token Interface Standard", ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf, June 2004.

[Perrin] T. Perrin, L. Bruns, J. Moreh and T. Olkin, "Delegated Cryptography, Online Trusted Third Parties, and PKI", Proceedings of the 1st Annual PKI Research Workshop, pp. 97-116, August 2002.

[Pope] N. Pope, J. C. Cruellas, "Oasis Digital Signature Services: Digital

Signing without the Headaches," IEEE Internet Computing, vol. 10, no. 5, pp. 81-84, September/October 2006.

[RADIUS] C. Rigney et al, “Remote Authentication Dial In User Service”, IETF RFC 2865, http://www.ietf.org/rfc/rfc2865.txt, June 2000.

[RSA] R.L. Rivest, A. Shamir and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, Communications of the ACM, vol. 21, no. 2, pp. 120-126, February 1978.

[SAPI] ARX, “SAPI Signature API Programmer’s Guide Version 4.1”, Pub. No. CSN.SAPI.V32.1206, Available on request from info@arx.com, December 2006.

[Sandhu] R. Sandhu, M Bellare and R. Ganesan, “Password-Enabled PKI: Virtual Smartcards versus Virtual Soft Tokens”, Proceedings of the 1st Annual PKI Research Workshop, pp. 89-96, August 2002.

[Schneier] B. Schneier, “Security Risks of Centralization”, Crypto-Gram, http://www.schneier.com/crypto-gram-0403.html#11, March 2004.

[SHA1] National Institute of Standards and Technology (NIST), “*FIPS Publication 180-1: Secure Hash Standard*”, http://www.itl.nist.gov/fipspubs/fip180-1.htm, April 1995.

[TLS] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol”, IETF RFC 4346, http://www.ietf.org/rfc/rfc4346.txt, April 2006.

[Whitten] A. Whitten and J.D. Tygar, “Why Johnny Can’t Encrypt: A Usability Evaluation of PGP 5.0”, Proceedings of the 8th USENIX Security Symposium, pp. 169-184, August 1999.

A New Paradigm in PKI Architecture: OTPK Technology For Online Digital Signature

Data Security Systems Solutions Pte Ltd
teikguan@dsssasia.com
efroni@datasecurity3.com
http://www.dsssasia.com

In this paper, we present a paradigm shift in PKI architectures. The OTPK concept is alarmingly simple to understand. Whenever a digital signature is required, the private key is generated, certified, used to compute the digital signature and immediately deleted. All that remains is the digital signature and the public key certificate from the Certification Authority (CA) that is used to verify the digital signature. There is no possible compromise on the private key, no need for user smart cards/USB tokens, no need for CRLs, no need for LDAP directories, no need for OCSP. It is compliant to international digital signature laws. The OTPK technology should be evaluated as a new and cost effective solution for on-line digital signature providing full mobility for mass usage of the public in different industries. It should be evaluated for this perspective, not from a CA perspective.

1 Introduction

The growth of on-line activities strongly depends on the reliability and availability of the service. The recent attacks on the on-line services such as phishing, man-in-the-middle and others have forced us to react. To meet the needs, a strong two-factor authentication has been introduced and become increasingly popular. In Singapore and Hong Kong, it is already mandatory to use two-factor authentication in finance sectors. In USA, two-factor authentication solutions were recommended last year. Subsequently, many organizations have been moving to a strong two-factor authentication using different solutions, for example, tokens, Vasco, RSA, Verisign, Active Card and many others. In addition, software tokens such as the OATH tokens, a Java midlet on mobile devices, SMS, scratch card, biometrics have been deployed to protect online applications against the hackers who try to steal users' identities and make use of them.

It is well known that PKI authentication and PKI digital signatures provide the best security for on-line activities for authentication and data integrity. But as of today, a full PKI solution for applications such as retail internet banking for banks that have a few millions of users have not been successfully deployed. This is because the cost of deployment is huge, the key management is troublesome, annual renewal of certificates and the operation of CA is a 'big headache'. But, can we simplify the process to take the advantages of the non-repudiation by means of digital signature? DSSS introduces a new concept with the OTPK.

The objective of this document is not to introduce a new CA technology. The aim of the paper is to introduce a new flow to implement digital signature, based on the recent endorsement of strong user authentication, to deliver an On-Line Digital Signature with full mobility that will make the digital signature affordable to all internet and intranet users.

2 About *OTPK*

2.1 Background of OTPK

The OTPK technology relies upon using a strong authentication infrastructure (e.g. OTP token authentication) to provide the functionality of on-line digital signature.

As for today, the use of asymmetric cryptographic keys to carry out data security functions such as digital signatures is becoming prevalent. Many countries including USA, European countries, Australia, Japan, Hong Kong and Singapore have passed some forms of legislation to recognize PKI, thus, to legalize the use of digital signatures as equivalent to hand-written signatures in contracts, transactions, etc. The applications that can use digital signatures range from a paperless-office to high-value B-to-B transactions over the Internet to high-security health-care information systems.

In a PKI, all communicating entities[1] or users rely on a trusted body, typically a trusted-third-party, to perform the necessary validations on the identity. This trusted-third-party, known as the Certification Authority (CA), will issue (directly or indirectly via a Registration Authority) to each of the participating entities, a digital certificate containing information about the entity such as the name, organization, country, the policies governing the use of the certificate and, most importantly, the entity's public key. The certificate asserts that the entity is the rightful and sole owner (and user) of a secret private key with which the public key is associated. If there are asymmetric cryptographic operations, such as a digital signature which is carried out using the particular private key, it can only be carried out by this certified entity and easily verified using the entity's public key published in the digital certificate and up the certificate chain to the trusted key by CA.

Naturally, we can equate the integrity and non-repudiation of the transaction to the security accorded to the protection of the entities' private keys. Each entity's private keys need sufficient protection to ensure that the keys are always in the possession and control of the rightful owners and cannot be stolen or duplicated. Smart cards (or USB tokens) are commonly used, as the medium, to protect private keys. But smartcards or USB tokens introduce additional problems of costs and logistics. The cost of a large PKI rollout using smart cards (taking into account of the cards, personalization, certification, etc) has been estimated at about US$100 per user. This estimation is extremely prohibitive for a regular day-to-day PKI usage and probably has been one of the most widely quoted reasons for the apparent slow adoption rate of PKI in most of the countries worldwide. In contrast, the total cost for a recent Internet Banking Security 2-factor authentication implementation using Vasco OTP tokens that we deployed for 1 million users was less than US$18 per user.

This paper, presents a revolutionary method to implement and deploy PKI, ensuring the same, if not higher, level of integrity and non-repudiation of the transactions, and yet not needing to incur the costs and logistics involved in deploying smart card solutions to apply to the digital signature law. We predict that this is the catalyst that the PKI boom has been waiting for.

2.2 The Concept of OTPK

The main concept behind OTPK is that the private key is a "One-Time Private Key" that works in connection with a short time certificate and is used for digital signature only to secure on-line transactions. As it is, OTPK cannot be used effectively for data encryption and user authentication.

There are essentially four steps that are carried out for each OTPK digital signature:

[1] An entity here is used loosely in this paper to represent a machine, a user, a group of users, an organization, a country, etc.

i. Generate the asymmetric key
ii. Send the public key for certification with the CA. At this step, OTPK relies on some form of authentication (strong 2-factor authentication is recommended) with the CA
iii. Receive the certificate and sign the transaction
iv. Delete the private key.

The validity of PKI certificate in this case need only be an extremely short term (in the order of minutes or seconds) to remove any chances of compromise. Since OTPK would result in a one-to-one mapping between the certificate and the transaction to be signed, details of the transaction can even be embedded in the certificate request for time stamping purposes.

In a typical PKI system, the user does a one-time generation and registration, and stores the certified key in a smartcard (or USB token) for a longer period of use. In contrast, the private key in the OTPK system is for one-time use only. A user always generates a new private key and authenticates securely with the CA in order to get a digital certificate for every transaction. Once the private key is used, it is expired and erased. There is no need to permanently store the private key in any media. Such a process sounds cumbersome; however, the overheads are actually not much more than any mobile credential solution.

The setup of OTPK requires the CA to have an online authentication and certification facility to fulfill all certification requests at a much higher throughput than existing setups of PKI. The entity could require a plug-in, implemented entirely in software to generate the private key, send the public key for certification, perform the digital signature operation, and delete the private key securely. The plug-in can be implemented as a PKCS#11, CAPI DLL or even as a zero-install Java applet embedded within the web browser.

3 Related Work

There have been many attempts to address the cost and logistics problems, each with varying degrees of success. Among all the attempts, perhaps the most widely deployed solution is the Microsoft CSP (Cryptographic Service Provider) [1] that is installed with the Windows Operating System. The Microsoft CSP is implemented as a software-token that operates as if it is a smart card and would perform the cryptographic functions of digital signing, encryption, key storage, etc. Access to the CSP can be protected by a password. The obvious problem behind the Microsoft CSP and all software tokens *per se* is that the private keys are typically stored in local hard disk storage which opens the chance for hackers to make duplicated keys.

A HSM (Hardware Security Module) is a widely deployed solution, too. While HSMs have traditionally been used as a host-attached appliance to carry out cryptographic operations for the host, many commercial HSM vendors, such as Eracom [2], nCipher [3], SafeNet [4] and Thales [5] have implemented HSM devices that communicate via network and, hence, can support multiple client/user connectivity. The network-attached HSM could, thus, function as a pseudo smart card for each of the entities connected on the network and would be responsible for the cryptographic storage and operations. However, the legal definitions of PKI may be violated, as the private key would not be technically in the possession of the entity and the digital signature is carried out on behalf of the entity.

Another alternative that can be seen to address the problem is the Keon Web Passport solution [6] by RSA Security, Inc. Keon Web is a “virtual” smart card implementation which relies on a backend server to secure and store private keys. When the entity requires the private keys, the keys can be downloaded securely to the entity's machine for usage, but it is still not foolproof. Backups in the backend server mean that multiple copies of the private keys exist. Moreover, the private keys are not always in the physical possession of the entity. This is a point of contention with some of the legal definitions [7, 8, 9] of a trusted and reliable PKI.

The Cosign [11] by Algorithmic Research Ltd is a very good example of a full digital signature solution. Cosign is mainly designed to support the enterprise that have a central user management like Microsoft Active Directory and strip most of the management issues. But again, the signing keys are stored in the server and the server signs on behalf of the user.

In the market, there are solutions that use on-line remote registration to acquire PKI credential including MyProxi, Kx509, Kerberized-CA and MyCA, to name a few. For authentication purposes, these solutions acquire long term or short term certificates and store them in either the server or client machine or other external storage device like smartcards. But they do not provide support for a short time certificate for digital signature.

4 OTPK vs PKI

The advantages of OTPK over the existing PKI systems include:

4.1 No Need for Smart Cards for Entities

In the OTPK system, since the entities' private keys are generated prior to a transaction and discarded after use, there is no need for traditional smart cards (or USB tokens) to store and protect the private keys. This represents very significant savings in terms of costs, resources and time overheads in implementing and maintaining a PKI system.

4.2 Much Smaller Window of Compromise

In the OTPK system, the duration of validity of the private key and certificate is extremely short. Also, by tying the certificate request to the content of the transaction and by adding time stamp we reduce misuse of the signing key. Typically the private key is used to generate only one or a few digital signatures for its lifetime. Moreover, the private key is erased after use. The combination of short duration, the lack of substantial signature data and absence of any key storage makes the OTPK system more difficult to compromise.

4.3 No Need for Large LDAP Systems

In a typical PKI system, the CA, after issuing the user's certificate, would publish the certificate with a LDAP system. This is to allow other participating entities to retrieve the certificate for verification purposes. Such LDAP systems have to be able to handle large amounts of load in order to support the verification process. In the OTPK system, since each certificate has small and limited time validity, the use of the LDAP for storing and publishing the entities' certificates is not feasible. Instead, the OTPK protocol would require that the certificate be attached with the digital signature in the transaction, for validation purposes only.

4.4 No Need to Maintain CRL or OCSP for User Certificates

In a typical PKI system, a CRL (Certificate Revocation List) and/or OCSP (Online Certificate Status Protocol) mechanism has to be in place to maintain the up-to-date status of the certificates. If a user has lost the private key, the corresponding certificate should be revoked and listed in the CRL. By doing so, the corresponding entities do not rely on the certificate from that point onwards. However, the CRL and OCSP mechanisms add significant overheads to the entire PKI process. In the OTPK system, the CRL and OCSP are no longer relevant because the private keys and certificates have limited time exposure and would not be compromised.

4.5 Lower Learning Curve

One of the problems with existing PKI implementations is the need to educate and re-educate the users. Most users find PKI rather confusing with the need to understand how to use smart cards including installing smart card readers, entering pins, changing the pins on a regular basis, how to use certificates, and what to do when the certificates expire, etc. Educating users takes up significant time, costs and resources. For the OTPK system, all the confusing cryptographic technology and PKI protocols are abstracted from the users. Instead, the users will need to use a more familiar 2-factor OTP authentication to approve the transaction. The complexity of the certificates and digital signature is either made redundant by the OTPK design or handled easily by the client plugin's interaction with the CA.

4.6 Easy Interface into 2-Factor/Biometric and Other Authentication Solutions

In a typical PKI system, there exists two points of authentication. One is with the CA for issuing an initial certificate which is carried out once in a long time. The other is with the media such as a smartcard that contains the private key. Authentication to the media is usually static PIN-based, as it is the media that enforces the authentication. If the protection of the media requires more complicated or stronger authentication, a lot of more complexity will have to be built into the media, resulting in higher costs. Moreover, not all media can support all forms of strong authentication.

In the OTPK system, only one point of authentication (with the CA) is needed. It is carried out when a private key needs to be used. Since the authentication can be centralized to a CA or a collection of CAs, there is economy of scale in implementing a strong authentication (such as 2-factor, biometric etc) to the CA and the cost can be shared across a large pool of entities. There is also no constraint on the media. This makes it easier to integrate a strong authentication mechanism into the OTPK system.

However, we do recognize that there are limitations to the implementation of the 2-factor or biometric authentication to the CA. For example, while OTP tokens are suitable for Internet-based transactions, remote authentication over Internet using biometrics is inherently insecure, and subject to replay attacks. On the other hand, using biometrics within a controlled office or a Kiosk environment for paperless e-Document systems is more convenient as compared to the OTP tokens. These considerations will have to be taken into account when designing the OTPK deployment.

We envision several scenarios where OTPK can be deployed:

- <u>Internet Transactions.</u>
  A merchant operates an Internet trading portal which requires the user to digitally sign transactions to signify approval. Users will login to the portal using a browser. In such case, the

OTPK client is a Java-Applet that is dynamically downloaded within the browser and users can be issued an OTP hardware token (e.g. Vasco or RSA SecurID) to authenticate with an Internet-based online CA for OTPK certificates.

- Enterprise eDocument
  A large enterprise operates an electronic document system to digitize the entire business workflow for processing efficiency and regulatory compliance. The application requires Microsoft Office and Adobe Acrobat documents to be digitally signed during the creation and approval process. For this scenario, the OTPK client can be in the form of a pre-installed CSP (Cryptographic Service Provider) DLL, and users can use UserID-Password, or Active-Directory authentication to authenticate to the enterprise OTPK CA for OTPK certificates. Biometrics, in the form of hand-written signatures, can also be used as the authentication means to the OTPK CA.

- Banking Kiosk
  A bank operates an ATM (auto-teller machine) network and requires the use of digital signatures for high-value transfers. The ATM can be deployed with finger-vein or palm-vein biometrics to serve as a stronger form of authentication. In this case, the end-user can use biometrics to authenticate to the OTPK CA (via the bank's internal ATM network) to get the certificate.

- Mobile phone
  A healthcare provider operates an e-prescription system that allows doctors to issue patient prescriptions electronically. All prescriptions need to be digitally signed. In this scenario, the doctor's mobile phone can be installed with an e-prescription application with OTPK capability. Doctors can be issued with a hardware OTP token. When issuing a prescription, the doctor will enter the OTP from the token to the e-prescription application which will generate the one-time-use key and authenticate to the OTPK CA for the certificate before submitting the prescription and signature to the health-care e-prescription system.

4.7 Private Key Always in the Possession of Users

Many of the legislation regarding Digital Signatures and PKI explicitly require that the user's private keys be always in the possession and control of the user [7, 8, and 9]. Such requirements imply that some of the mobile credential solutions would not be recognized as compliant to the Act. The OTPK system relies on a client plug-in to generate and temporarily store the private key for the short duration that the Private Key is used. In the entire process, the private key remains in the possession and control of the user.

4.8 Protocol Is Interchangeable for All Asymmetric Algorithms

The OTPK system does not differentiate between different asymmetric algorithms and allows for entities using different asymmetric algorithms (e.g. RSA, DSA, ECDSA, etc) to participate within the same PKI. This means that one user can use RSA to perform digital signatures while another user can use ECDSA. Since the CA handles the certification collectively at the point of performing the digital signature, the OTPK solution is flexible enough to allow different entities using different algorithms to participate together. For example, the same user may use RSA in one country and ECDSA in another country depending on the electronic regulations and laws governing the countries.

Also, in the event that an algorithm is deemed undesirable, due to whatsoever reason such as cryptographically broken, insufficient key length, licensing, poor performance, platform constraints, etc, the user can easily use a different algorithm or key length without affecting all the other participating entities. The CA may also seamlessly migrate entities from using one algorithm to another without affecting the PKI or the PKI operations.

This flexibility allows different entities to use different applications. It also allows entities with certain platforms and restricted type of algorithms to participate in the system. Finally, it allows entities that are unable (or not allowed) to use certain types of algorithms or certain key lengths to participate in the PKI. Such flexibility is currently not practical within the existing PKI system.

4.9 Solution Is Very Scalable

Since most of the cryptographic load (i.e. Key generation, etc) is actually carried out at the user end, the load on the CA is only the cost of 1 asymmetric key signing operation per transaction. Each signing operation is also stateless, meaning that multiple CAs performing the OTPK certification need not synchronize the keys or certificates with each other.

From an operational perspective, the OTPK solution can be easily scaled up to handle larger volumes by adding more points of presence of the certification and authentication servers. The implementation can rely on a certification chain, leading up to the root CA, where sub-CAs that operate the certification and authentication servers can perform the user certification on behalf of the root CA. These sub CAs spread out the certification load and do not compromise the overall security of the OTPK solution.

4.10 Efficient and Effective Business and Pricing Model for CA

In the typical PKI, the CA charges on per-certificate basis. However, since the private key to the certificate can be used to sign many transactions, the CA charges a significant amount of money per certificate. Such a pricing model does not efficiently charge according to the actual usage since a user that uses the private key regularly versus another user that uses the private key rarely are charged the same amount.

In the OTPK system, since the certificates are issued each time a private key is used, the CA can charge a much smaller amount for each certification. Such a pricing model will mean that entities that use the private key more often will incur more charges, and vice versa. This results in a fairer and more acceptable pricing model. It also allows CAs to price the certificates and services differently for different applications such as (but not limited to) the following:

- Mode - online certification versus batch certification are priced differently
- Timing - certification requests during peak hours will incur higher charges
- Loyalty - the more certificates are requested, the cheaper the cost of each certificate
- Branding - different classes of certificate with different certification policy are priced differently
- Algorithm - certificates for different algorithms are priced differently.
- Insurance - price of certificate includes insurance on the transaction that is tied to the certificate
- Duration - one-time use versus per-session use certificates cost differently

A further advantage is that the OTPK certificates can integrate transparently with current PKIs. Relying party software that can process traditional PKI certificates can also process OTPK certificates, barring a possible X.509 extension indicating that status information is not published. An existing CA can choose to issue both traditional PKI as well as OTPK certificates and allow both systems to interoperate, ensuring the maximum flexibility for the CA to adjust the business model.

5 Addressing OTPK Issues

While OTPK is able to solve some of the very key issues (e.g. logistics, costs, compliance to laws) plaguing traditional PKI setups, OTPK introduces a number of new issues that have to be addressed in order to make OTPK a viable digital signature solution.

5.1 Online CA key

The most distinct difference in the backend setup of a traditional PKI CA versus an OTPK CA is the use of the CA Key, or the key that is used to certify the certificates.

For the OTPK CA, the CA Key has to be accessible online as the certification requests are expected to be fulfilled in real-time. The entire certification process is expected to be carried out within a couple of seconds to ensure that the transaction approval process is not delayed. In contrast, the CA Key in traditional PKIs need not be online as the certification process may take up to 48 hours to allow for manual processes to be carried out. The concern here is if the security of the CA Key is compromised in any way by making the key online, versus using some physical means to ensure that the key is not accessible on the Internet.

We argue here that while the concerns, on the surface, seem to point to a more vulnerable CA, having an online CA Key does not lower the security of the PKI setup. This is because:

- Stolen CA Key
  The use of a high-level FIPS-certified HSM (at least Level 3) will mitigate this risk by making it impractical to extract the private key.

- Fake certificate requests
  All OTPK certificate requests come embedded with the authentication credentials of the user. The authentication credentials can be in the form of a one-time-password from the user's token. This allows the CA to verify the user before issuing the certificate.

  The process of verifying the authentication credentials + certifying the key should be done in one atomic step within the HSM to ensure that a compromised system is unable to illegally send certification requests to the HSM. By insisting on the use of strong authentication + HSM with OTPK, we are able to mitigate this exposure.

5.2 User Registration

Another difference is in the user registration process. In the traditional PKI CA, the user registers with the CA once to generate the user's private key, and get the public key certified by the CA. During the registration process, one key step is that the CA would verify the credentials of the user. Once done, the user is free to use the private key without needing to contact the CA.

For OTPK, this registration process seems to be missing while the certification process is repeated each time the user needs to sign a document or transaction. However, we need to clarify here that the registration process did happen. If we are to extrapolate backwards to the point in time when the user was first assigned the authentication token (assuming a one-time-password token), this was when the registration process actually took place. As for the repeated certification processes, it is simply equivalent to the user authenticating to the CA and obtaining services from the CA.

5.3 Secure Time-stamping

Current time-stamping (or electronic notary) solutions rely on a central time-stamping server that essentially signs on the hash of the transaction and include a time-stamp with the transaction [15]. This is an issue that is relevant even for traditional PKI digital signature implementations which typically rely on the user's PC date/time for the time stamp. How can we prove that the user signed the transaction at a particular time?

For OTPK, the solution is rather apparent. This can be done by simply allowing the CA to also function as the secure time-stamping service. When the user generates the certificate request, the hash of the transaction to be signed is also embedded as part of the certificate request, along with the authentication credentials. This allows the CA to issue the short-lived OTPK certificate of the user key, and with a reliable time-stamp on the hash value (e.g. as one of the X.509 extensions) back to the user. This certificate can thus be used as the proof for the time-stamp.

5.4 Secure Private Key deletion

The issue for private key deletion comes in when the certificate has expired, and we do not want the private key to be used for any other purposes (since it is no longer valid). For traditional PKI setups where the private key is securely stored in a smartcard or USB token, the destruction of the private key is more visible, and since certificate expiries do not happen as often (typically only once a year or once in 3 years) as OTPK certificates, the requirements for key deletion is not as pronounced.

For OTPK certificates, a new key is used for each digital signature which may result in hundreds (or even thousands) of keys used by a user in a year. This gives a hacker potentially more chances to obtain a user's private key, albeit with an expired certificate.

For OTPK, we are able to address the problem in two ways: directly and indirectly. In the direct approach, we have to ensure that the private key is deleted as designed. This can be achieved by implementing the key deletion process in the OTPK client as part of the atomic function of the signing process (i.e. Generate Key-Get certificate-Sign-Delete Key), and ensuring that this process cannot be modified through secure programming techniques as well as sending the OTPK client for FIPS-140 certification. We recognize that this method is not foolproof standalone and is vulnerable to a crafty signer who fully intends to cheat the system.

In the indirect approach, we have to make sure that the private key cannot be used for any other transaction. This can be implemented similarly to the secure time-stamping in Section 5.3. Since the certificate and key is directly tagged with the transaction, using the private key to sign a different transaction would result in a signature validation failure.

6 Conclusion

The OTPK technology is bringing up a new concept in which a user will generate a signing key with an extremely short lived certificate to perform the digital signature. The PCT (Patent Cooperation Treaty) has defined the OTPK as *'novel and innovative'* [14]. The key of the innovativeness is that the OTPK technology allows an implemention of on-line digital signature system that complies to the digital signature law with full mobility and low cost of ownership. The entity is generating the signing key and owns it during the whole process of "Key Generation" "Certification" "Signing" and after signing deleting the Signing key. It could be regarded as a new paradigm in the "PKI" technology that allows the population of the digital signature to many vertical markets.

To put things in perspective, we have benchmarked a Java applet OTPK implementation which uses RSA-1024 keys on an IE browser. The time taken for the key generation + certificate request + digital signing takes less than 7 seconds on a Pentium 3 machine and less than 2 seconds on a Pentium 4 machine. For the mobile phone, a J2ME OTPK application using ECDSA-P192 averages between 3 to 10 seconds on various mobile phones.

DSSS is currently implementing OTPK protocol and proof of concept into the DSSS Authentication Server for demo purpose only. It is planned to be further enhanced with XKMS and WS-Security. (The OTPK is patent-pending USPTO 60/590,348)

References

[1] Microsoft Platform SDK: Cryptography. http://msdn.microsoft.com/library/en-us/seccrypto/security/microsoft_cryptographic_service_providers.asp

[2] Eracom Technologies. http://www.eracom-tech.com

[3] nCipher Corporation Ltd. http://www.ncipher.com

[4] SafeNet Inc. http://www.safenet-inc.com

[5] Thales e-Security. http://www.thales-esecurity.com

[6] RSA Keon Web PassPort. http://www.rsasecurity.com/node.asp?id=1230

[7] Electronic Transaction Act 1998, Singapore, Part IX – Duties of Subscribers, Clause 39. http://www.ida.gov.sg/idaweb/pnr/infopage.jsp?infopagecategory=regulation:pnr&infopageid=I1944&versionid=1

[8] Utah Digital Signature Act, Part 3. Duties of certification authority and subscriber, Utah Code 46-3-303. http://www.jus.unitn.it/users/pascuzzi/privcomp97-98/documento/ firma/utah/udsa-3.html

[9] Georgia Digital Signature Act. Section 1, Article 3, Clause 10-12-33, Subscriber's Warranties

[10] Peter Gutmann, "Plug-and-Play PKI: A PKI your Mother can Use", Usenix Security Symposium, 2003 http://www.cs.auckland.ac.nz/~pgut001/pubs/usenix03.pdf

[11] Algorithmic Research Cosign – www.arx.com

[12] AlphaTrust PRONTO™ Server-http://www.alphatrust.com/

[13] Deepnet Technology Smart ID VSC http://www.deepnettechnologies.com/

[14] PCT Review, PCT/SG2005/000226

[15] RSA Security, "What is digital timestamping ?". http://www.rsasecurity.com/rsalabs/node.asp?id=2347

Universal Certificate Authentication to Key Applications at Argonne National Laboratory

Doug Engert
Rich Raffenetti
David Salbego
John Volmer

Argonne National Laboratory
9700 South Cass Avenue, Argonne IL 60439

February 26, 2007

Argonne National Laboratory has implemented a laboratory-wide portal that provides centralized access to key administrative applications and employs certificates for authentication. This portal relies on an infrastructure comprising Microsoft Active Directory, Microsoft Certificate Services, Sun Microsystems Java Enterprise Suite, and open-source software. The capabilities of the Microsoft, Sun, and open-source products have enabled Argonne to readily deploy certificates for partial, as well as for end-to-end, authentication from all Argonne client operating systems. The Argonne experience demonstrates that certificate authentication to corporate applications is readily doable today. Further, the adoption of these technologies positions Argonne to exploit widespread certificate deployments, as intended by Homeland Security Presidential Directive-12.

Introduction

Challenge

In enabling certificate access to applications, organizations must address two issues:

- Providing users with certificates, and
- Enabling applications to accept certificates.

Users cannot be easily provided with certificates because certificates (and their corresponding private keys) are difficult — if not impossible — for users to simply manage. The certificate and private key comprise a few thousand bytes of binary data. Further, the private key must be safeguarded because control of the key is the basis for assuring identity.

Certificate authentication also cannot be easily incorporated into applications because enabling certificate authentication requires considerable technical knowledge of Public Key Infrastructure (PKI) concepts and public/private key algorithms.

Solution

Argonne National Laboratory has overcome both of these issues and enabled certificate authentication to corporate applications for most users by using commercially available and open-source technology. Argonne addressed the issue of providing users with certificates by adding Microsoft Certificate Services and the University of Michigan’s KX.509 package to its authentication infrastructure. Microsoft Certificate Services enable organizations to issue either short-term or long-term certificates to hundreds of users. Simultaneously, Argonne adopted Sun Microsystems Java Enterprise Suite to incorporate certificate authentication into web-based applications. The Sun Microsystems suite includes simple mechanisms for adding certificate processing to applications.

The combination of these two commercial technologies and open-source software provided immediate and unexpected benefits. Not only do users have certificate-based application access, but in many cases, the approach that we used enables single sign-on. Further, with the addition of smart cards, we were able to provide end-to-end authentication based on certificates.

An organization that is readily able to accept certificates for authentication is ideally positioned for the implementation of Homeland Security Presidential Directive-12 (HSPD-12). The intent of HSPD-12 is for smart cards containing certificates to be issued to all federal employees and contractors beginning in October 2006. An organization able to capitalize on the widespread availability of certificates can greatly simplify the user's password management burden.

This paper presents a detailed discussion of how Argonne National Laboratory addressed the two challenges associated with certificate access: providing user certificates and enabling applications.

Background

The Argonne National Laboratory authentication infrastructure has developed over the years from standalone Kerberos servers to a Distributed Computing Environment (DCE) and, recently, to Microsoft Active Directory. Active Directory has become Argonne's institutional authentication mechanism.

Upon date of hire, employees are provided with an identity in Active Directory, which is a combination data store and service provider. Domain controllers are computers that manage the data store and offer, for example, the following services to clients:

- Kerberos ticket services, and
- Lightweight Directory Access Protocol (LDAP) access to Active Directory's contents.

A *domain* is the collection of computers and users managed by an active directory instance.

Active Directory is a powerful tool in the management of a Microsoft Windows domain. It permits distribution of information and policies to all members of the domain, both client computers and users. One use of Active Directory is to define trusted root certificate authorities. Certificate authorities defined in Active Directory are automatically trusted by all clients and domain controllers.

Approximately 60% of Argonne's desktop workstations have Microsoft Windows 2000/XP operating systems, and nearly all of these workstations are members of the ANL.GOV domain managed by Active Directory. The other 40% of Argonne's workstations are Macintosh (20%) and Unix (20%). Argonne's Mac and Unix workstations are not managed by Active Directory. Argonne's ANL.GOV domain processes 1,500 unique logins per day.

Because all employees are provided with an identity in the ANL.GOV domain (Active Directory), they all have Kerberos principals. All employees, regardless of their desktop platform, are able to acquire Kerberos authentication credentials from the authentication infrastructure.

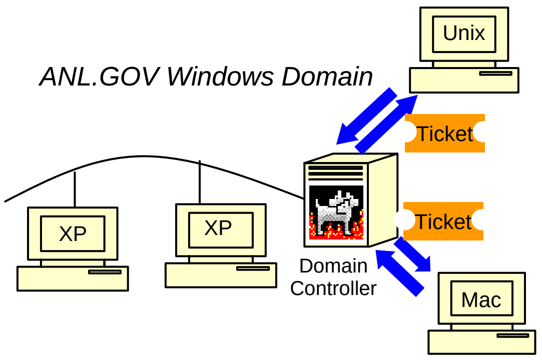


Fig 1: All Platforms Can Authenticate to Active Directory

As shown in Figure 1, Argonne's Unix and Mac computers can take advantage of the Kerberos services provided by Active Directory if they are configured to do so. Unix workstations implementing pam_krb5 and Mac workstations configured to use Active Directory obtain Kerberos tickets automatically during login processing. If the computer is not configured to perform Kerberos logins, the Kerberos kinit command ("acquire a Kerberos ticket") can be run on the computer to acquire Kerberos credentials after logon.

Many of Argonne's administrative systems rely on the ANL.GOV domain to authenticate users, particularly systems that are used directly by all employees. These systems include high-profile applications such as Human Resources' Performance Appraisal and Open Enrollment systems.

Certificate Authentication Architecture

Certificate Issuance

In the spring of 2002, Argonne began experimenting with the University of Michigan's KX.509 suite to enable testing of certificates with real-world applications, particularly for the Globus project. Globus relies on certificates to perform user authentication, and KX.509 permits organizations with a Kerberos infrastructure to easily issue short-term user certificates. KX.509 constructs short-term certificates from existing Kerberos credentials.

Subsequently in 2004, Argonne began investigating two-factor authentication for its Microsoft Windows-based administrative users. Microsoft Windows naturally supports smart cards; the most straightforward path for enabling a smart card pilot was to install Microsoft Certificate Services.

Microsoft Certificate Services are primarily a certificate authority coupled with a web application. Smart cards require the deployment of Microsoft Enterprise Certificate Services[1].

Microsoft's Active Directory provides the framework for Enterprise Certificate Services. For example Certificate Services uses Active Directory to identify users for smart card issuance and to publish Certificate Revocation Lists (CRL). Additionally, Active Directory policies can draw on Enterprise Certificate Services. One of these policies (which will be discussed later) is the ability to automatically issue certificates to users — in effect, to auto-enroll users.

As an example, within 24 hours of our installation of Enterprise Certificate Services, all of Argonne's 38 domain controllers detected its presence and requested domain controller certificates from Certificate Services. In response, Certificate Services automatically issued domain controller certificates to each of the domain controllers, as shown in Figure 2.

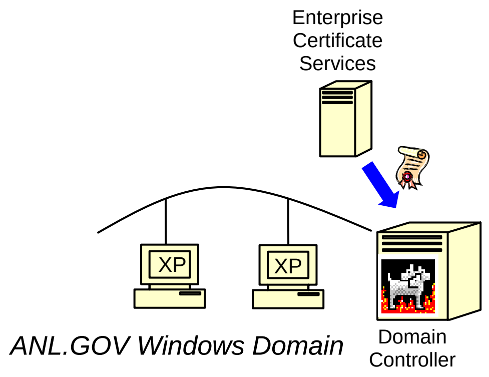


Fig. 2: Issuance of a Certificate to a Domain Controller

A web application associated with Enterprise Certificate Services provides the means to manually submit requests and obtain certificates. The web site acts as an enrollment station for agents to provide smart cards on behalf of clients.

Auto-Enroll Certificates

Microsoft Enterprise Certificate Services provide a capability known as Auto-Enroll Certificates. Auto-enrollment allows organizations to avoid the high effort costs associated with traditional certificate issuance by using domain policy to automatically issue certificates. No new services need to be installed to enable auto-enrollment.

Users who log in to Windows XP work stations and who are members of the domain are selected by group policy to trigger the auto-enrollment process. A certificate request is issued, and Certificate Services immediately responds with a certificate for the user. The certificate and private key are stored in the user's profile, and the certificate is propagated to the user's certificate store. Figure 3 depicts this process.

At Argonne, approximately 2,000 users are presently selected to receive auto-enroll login certificates. Each week, 600 Auto-Enroll

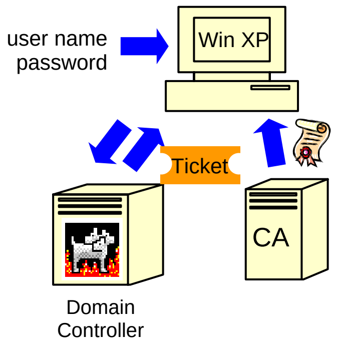


Fig. 3: Issuance of an Auto-Enroll Certificate to a User

Certificates are issued to users. These certificates have a lifetime of 30 days.

KX.509 Certificates
The University of Michigan's Kerberized Certificate Authority (KCA) and kx509 (an element of the KX.509 suite) programs are used to provide short-term certificates to users of workstations that are not members of the ANL.GOV domain managed by Active Directory. These tools provide the same service as the Auto-Enroll Certificate, i.e., short-term login certificates derived from login credentials. The KX.509 tools are available on workstations that do not run Windows, such as Unix and Macintosh, and to non-domain Microsoft Windows clients as well.

Two KCA servers issue certificates to users. KCA certificates are derived from the Kerberos tickets of the users who make kx509 requests. The certificate subject name is derived from the Kerberos principal name, and the certificate lifetime is the remaining lifetime of the Kerberos ticket used in the request. The subject name is always the same for a given user. Figure 4 shows the KX.509 certificate issuance process.

When kx509 is run on a Windows machine, the certificate and private key are stored in the user's certificate store, and are thus accessible — like any other certificate. When kx509 is run on a non-Windows machine, the certificate and private key are stored in the Kerberos ticket cache. Both are made available to applications via the KX.509 kpkcs11 executable.

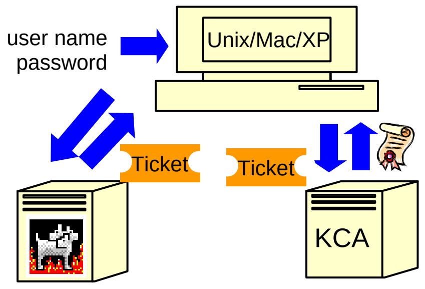


Fig. 4: Issuance of a KX.509 Certificate to a User

Because the certificate lifetime is usually less than a day, CRLs are not issued or checked. A user can also discard the certificate and the private key by using the kx509 program or by destroying the Kerberos ticket cache.

KX.509 certificates are rarely requested. The two KCA servers issue fewer than two certificates per day.

Smart Card Issuance
Smart card issuance requires the following two additional components beyond the installation of Microsoft Enterprise Certificate Services:

- The physical equipment of smart cards and readers, and
- Smart card middleware — specifically a Cryptographic Service Provider (CSP) that provides an interface between Microsoft Windows and the smart card.

Argonne chose Gemalto GemSAFE smart cards and Gemalto GemLIB v4.2 middleware for its smart card pilot[2]. The middleware includes both a CSP for accessing the card, as well as a tool for managing the card.

Microsoft Enterprise Certificate Services provide a web interface for smart card issuance. The default interface assumes that smart cards will be issued in person by an authorized official, such as an enrollment agent. At Argonne, the Laboratory's Account Services personnel issue smart cards. Account Services personnel select the user who is being issued a smart card from Active Directory.

Microsoft Enterprise Certificate Services interact with the smart card to generate a public/private key pair, construct a certificate request, issue a certificate, and place the certificate on the smart card. Smart card issuance requires 5 minutes, and the certificate is valid for 2 years.

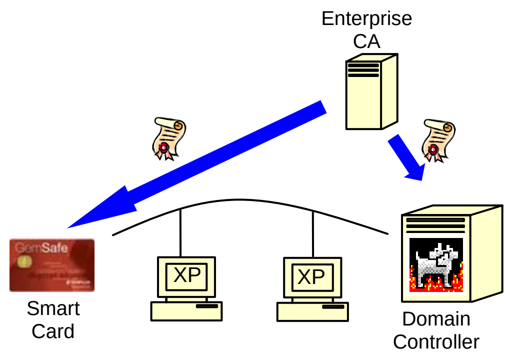


Fig. 5: Issuance of a Smart Card Certificate to a User

As shown in Figure 2 earlier, Microsoft Enterprise Certificate Services automatically issued certificates to domain controllers. With the issuance of a certificate to the user, as shown in Figure 5, a third-party trust model is created within the Windows domain.

Once issued, the smart card instantly enables login to Microsoft Windows workstations equipped with smart card readers and the smart card middleware. No additional configuration action is required by computer administrators. At login or when the smart card is inserted in the reader, the certificate is propagated to the user's certificate store.

Today, Microsoft smart card login certificates contain the *User Principal Name (UPN)* in the *Subject Alternate Name* field of the certificate. The UPN form is *username@domain*, which is the Active Directory identity of the user. This UPN is used by the domain controllers to select the user account for login when presented with a smart card. Thus, the mechanism for issuing the smart card requires smart card users to be members of Argonne's Microsoft Windows domain (ANL.GOV). Approximately 60 users at Argonne have smart cards.

Portal Architecture

In 2004, Argonne National Laboratory undertook a strategic business initiative to implement a web portal for its business systems. The developers envisioned a single framework that would serve as the official repository for all administrative applications and information — enabling Argonne to manage identities, roles, and responsibilities and providing employees with customized access to information. Employees would benefit from a single sign-on interface that would speed entry to the administrative applications.

Initially, the portal was designed to host in-house developed applications for Human Resources and Payroll transactions. The goal was to automate these tasks in order to significantly increase employee productivity.

Overview

The Argonne Administrative Systems Portal architecture is based on the 2005Q4 release of the Sun Java Enterprise System (JES). The JES suite consists of a number of related products, including a directory (LDAP) server, a web server, a Java application server, a portal server, and an access manager. These products represent the core of the product suite, and they are all in use at Argonne in a redundant, load-split architecture.

The Sun JES was chosen by Argonne for two primary reasons: past positive experience and attractive cost. Several of Sun's software products have been in use at the laboratory for many years, including the web server and directory server. Argonne had an established group of administrators who were familiar with Sun products and whose positive experiences with these products allowed us to experiment with additional Sun software. Sun software is also relatively inexpensive compared with other commercial software; the Sun JES software itself is free, although support is not. A comparison of the cost of the Sun software with that of competing commercial single sign-on and identity management products and our past positive experiences with Sun products made the selection of Sun JES straightforward. The Sun JES suite is licensed across the Argonne campus.

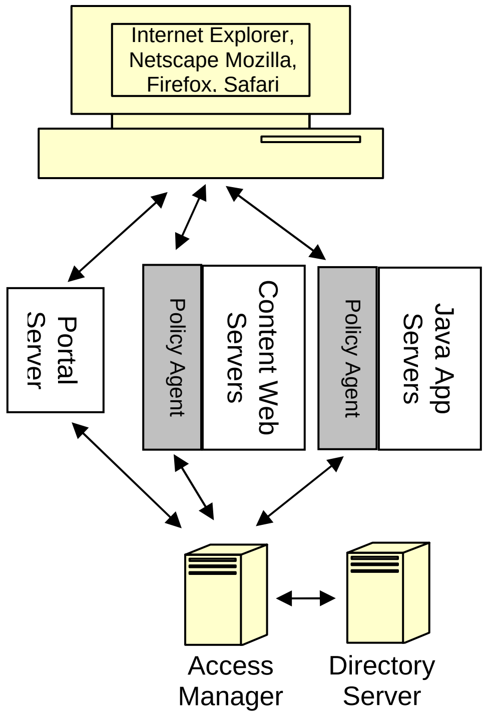


Fig. 6: Authentication Communication of the Sun Access Manager

Access Manager
The Access Manager is the central authentication and authorization mechanism used by other web services and resources. All requests for authentication, authorization, and session state flow through this service. As shown in Figure 6, the Access Manager is the key component — bringing together disparate web services and resources.

Authentication
A number of different authentication protocols (including LDAP, Unix, SecureID, RADIUS, and X.509) are accepted by the Access Manager. At Argonne, the LDAP and X.509 modules are used in production. The LDAP module is configured to work with Argonne's Active Directory infrastructure for user name and password authentication. The X.509 module is configured to accept several types of X.509 user certificates for authentication.

These authentication modules may be *chained* together. Chaining allows multiple authentication modules to be used in succession. Rules define the requirements for each module. For example, users may be required to authenticate against module "A," with authentication against module "B" being optional. Another scenario may require authentication against both module "A" and module "B."

In Argonne's scenario, two modules are chained together for public use: a certificate module and a user name and password module. The certificate module is invoked first. If a user has established a Transport Layer Security (TLS) connection to the Access Manager by using a client certificate, the certificate module attempts to use that certificate to authenticate the user. If no client certificate is available, a username and password prompt appears.

When an X.509 certificate is presented by an end user, it must be signed by a trusted certificate authority[3], and there must be a map between a distinguished name component and the user profile stored in the LDAP server of the portal. The map defines which components of the presented certificate are to be used to locate the correct user profile in the LDAP server. The user profile specifies the Active Directory identity (i.e., Kerberos principal) of the user.

Authorization
The Access Manager employs an LDAP service to store configuration data, user session information, user portal profiles, and additionally authorization data such as group and role information. Applications use these authorization data to determine user privileges.

Argonne has developed an in-house centralized "role" (or group) management system called Information Services Authentication and Access Control (ISAAC). Users of ISAAC may create new roles, modify the membership of roles, and produce reports based on given criteria. Roles are defined and managed within ISAAC and distributed to multiple systems, including Oracle, Active Directory, and LDAP (for Access Manager).

Portal User Interface
The Argonne Administrative Systems Portal represents the gateway to other web applications and resources throughout the organization. This web site provides access to related applications. In Argonne's case, the portal is used to co-locate the entry points for many administrative systems required by individual employees. Users access the portal applications via their web browsers. One feature of portals is that users can customize their portal experience; their preferences are saved as part of their user profiles.

The Administrative Systems Portal relies on the Access Manager to provide authentication and authorization services. The portal also heavily relies on the directory service to store user profiles and related information. The Portal is the largest user of roles stored in Access Manager.

By utilizing roles, portal developers create a dynamic, customized end-user experience. An example role is "supervisor." After logging in to the portal through the Access Manager, an employee with a "supervisor" role will have additional application links visible to them. Applications such as "Performance Appraisal" will behave differently when used by a supervisor, as opposed to an employee.

Content Web and Java Application Servers
A number of web and application servers are used at Argonne, and many of them provide laboratory applications. The most well known is the Human Resources Performance Appraisal application. Frequently, these applications require authentication before they can be used. All portal-based applications use Access Manager Policy Agents to conduct authentication on their behalf. Different Policy Agents are available for a wide variety of web and application servers. For cases in which authentication is required, Argonne's web servers use TLS. The server certificates that enable TLS are signed by widely trusted external commercial certificate authorities. This approach allows all browsers, specifically the non-Active Directory browsers, to automatically trust Argonne's administrative web servers.
The key to a seamless portal experience is single sign-on (SSO). Although not required, single sign-on allows users to jump from resource to resource and application to application within the portal without having to log in to each component individually. If a user had to provide credentials to each application he accessed, even if those credentials were identical, the experience would be severely diminished.

Policy Agents
The Access Manager provides for SSO capabilities within the portal through the Policy Agents, i.e., the security layer between the user and the resource. When a protected resource is accessed, the Policy Agent determines whether a user is authenticated, whether a resource is protected, and whether an authenticated user has access to a protected resource (authorization). These aspects are configured through the use of a local configuration file and a central policy repository located on the Access Manager.

Traditionally, simple web applications request a user name and password by using Hypertext Transfer Protocol (HTTP) *basic* authentication. For this type of authentication request, the web server instructs the client browser to bring up a pop-up window that asks for a user name and password. The provided user name and password are returned to the web server, which validates the response against a local data store (can be a simple text file or perhaps an LDAP server). If the proper information was provided by the client, the environmental variable *REMOTE_USER* is set by the web server. The web application can then use the *REMOTE_USER* variable in any way it wishes. The authentication layer is therefore separated from the application layer.

The Policy Agent works in a similar fashion. Accesses to protected URLs are intercepted by the Policy Agent, which asks the Access Manager to validate the end-user's credentials. If the end-user has not previously authenticated (does not have a proper SSOToken browser cookie), he is redirected to the Access Manager for authentication. After credentials are

provided, the Access Manager redirects the end-user back to the Policy Agent.
When the Policy Agent returns control to the application, it also returns several standardized data objects, including *REMOTE_USER*. The Policy Agent sets the same environmental variables that are set by a web server using basic authentication, so for the application being protected, it does not generally matter whether HTTP *basic* authentication or Policy Agent authentication is used. The underlying application can then use the environment provided by the Policy Agent for further authentication and authorization.

A simple application that previously used *basic* authentication can easily be configured to use the Policy Agent, which is installed as a separate component onto a web server or application server. In the case of Sun Web Servers, the name of the shared library containing the Policy Agent executable is added to the *magnus.conf* file. A configuration file is simultaneously created on the web server that simply defines *which* URLs are to be protected by the Policy Agent. The native web server access control list is modified to disable protection of the resource.

The specific access control list for the URL is maintained by the Access Manager. The Sun JES includes a graphical user interface to manage access control lists.

Complex web applications — typically larger open-source projects and commercial products — require code modification and customization to integrate into a Policy Agent environment. A direct API is available for applications needing to forego the Policy Agent and communicate directly with the Access Manager. The amount of effort required to integrate a product into the Access Manager SSO environment depends heavily on the complexity and implementation of the application.

A significant advantage of using the Access Manager to provide authentication to a large number of applications is consolidation of authentication. Assuming a modest application inventory, it would be challenging to upgrade each application to accept a new form of authentication credentials. For example, if a site were to move from username and password to certificates as its primary authentication mechanism, each application must be modified to accept this new credential. By using the Access Manager, credential changes only need to be made in one location. Applications using Policy Agents or the Access Manager API do not need to be altered. Once an application is integrated into the Access Manager environment, little else needs to be done to accommodate new authentication mechanisms.

Policy Agents are available from Sun Microsystems and third-party vendors for a wide variety of web resource environments, such as the commercial Java application servers (IBM WebSphere, BEA WebLogic, Oracle Application server, and Redhat JBoss), Microsoft Internet Information Server (IIS), Apache, and Tomcat, among others. The Agents allow third-party web providers to be integrated into a centralized authentication infrastructure. The Access Manager API can be used directly to integrate almost any application, provided source code is available. Commercial software vendors have been willing to modify their products to integrate into SSO solutions.

Browsers
Browser compatibility is a significant issue in enabling a successful portal. Argonne’s portal components have been tested with Microsoft’s Internet Explorer, Firefox, and Mozilla.

Multiple browsers (including Internet Explorer, Mozilla, and Firefox) can use the University of Michigan’s KX.509 certificates. Mozilla and Firefox require the kpkcs11 component of the KX.509 suite to be installed on the client workstation.

The kpkcs11 component implements the RSA PKCS#11 standard to enable applications such as web browsers to access certificates stored by kx509. The Windows version is a dynamic link library (dll), and the Unix version is a shared library. These executables emulate security devices (e.g., smart cards) to Mozilla or Firefox.

Authentication Process

Workstation Login

Workstation login that results in users having a personal certificate can be conducted in three ways:

- With a user name and password using auto-enroll certificates,
- With a user name and password using KX.509 certificates (manual), or
- With a smart card.

The highlight of all logins is that, at the end of login, the user has *both* a Kerberos credential issued by the domain controller *and* a certificate issued by either Microsoft Enterprise Certificate Services or the KX.509 package. The user can immediately access resources that request *either* form of authentication.

Figure 7 summarizes authentication credential flow from user workstation logon to application admission.

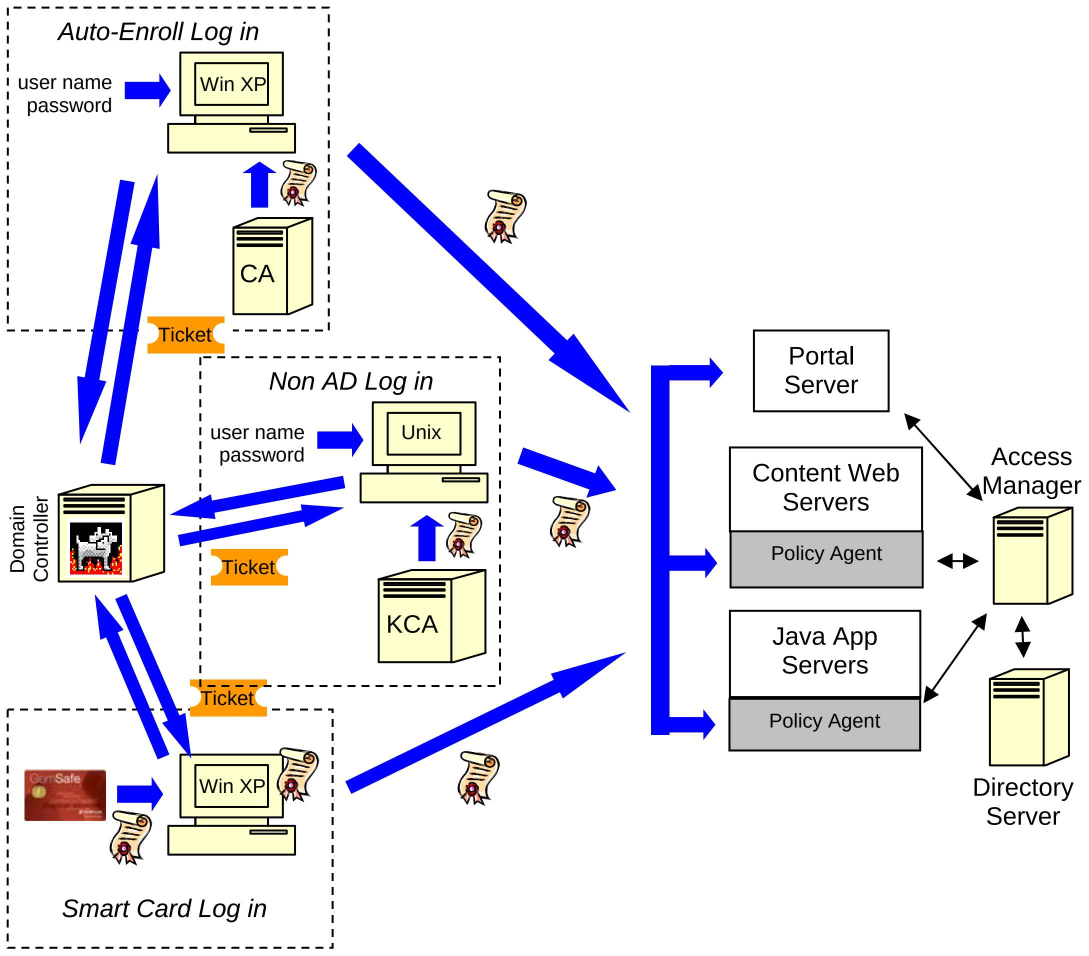


Fig.7: Authentication Communication From Logon to Application

User Name and Password

In this mode, the users log in to their workstations by using their user name and password and then acquire a certificate.

Certificate acquisition may be automatic and invisible when using auto-enroll or manual when using the KX.509 process.

• *Auto-Enrollment*
Users initiate a standard Microsoft Windows domain login by providing their user name and password. At login, the user obtains Kerberos credentials. As part of the login process, Group Policy settings are evaluated (including the policies for auto-enrolled certificates). If there is no certificate or if it has expired, an auto-enroll certificate is obtained. The process is completely transparent; users are unaware of the auto-enroll certificate process, and no action is required on their part to obtain or renew a certificate.

Auto-enroll certificates are usable only as long as the associated Windows domain account is enabled. In a non-Roaming Profile environment, a user logging onto another Microsoft workstation obtains a new auto-enroll certificate. In a Roaming Profile environment, the certificate is transmitted with the user profile.

• *Dynamic KX.509 Certificates*
Kerberos Login. If the desktop computer conducts a Kerberos login, the user receives a Kerberos ticket as part of the login process. The user then manually requests a short-term certificate by running the kx509 client program on his/her desktop computer.

The kx509 program uses Kerberos to authenticate to one of the KCA servers. The program generates a public/private key-pair and sends the public key to the KCA server. The KCA server returns a certificate good for the lifetime of the Kerberos ticket used in the request — typically 12 hours or less. The certificate and private key are stored on the local computer.

Non-Kerberos Login. If the desktop computer does not conduct a Kerberos login, users must manually obtain a Kerberos ticket using kinit. Users request a short-term certificate by running the kx509 client program on their desktop computers, as they would if they had performed a Kerberos login.

• *Smart Card*
The insertion of a smart card is automatically recognized by Microsoft Windows Graphical Identification and Authentication (GINA). Windows immediately prompts the user for the Personal Identification Number (PIN) that permits use of the private key functions of the card.

The client workstation uses the PKINIT component of the Kerberos protocol to obtain Kerberos credentials. The *User Principal Name* contained in the *Subject Alternate Name* field of the certificate enables the domain controller to select the user for which a session should be initiated.

Validation of the user's certificate by the domain controller is included in the login process. The controller validates the certificate chain and inspects the CRL of Microsoft Enterprise Certificate Services.

Smart card login is quick and easy for users. In the Argonne smart card pilot program, several non-technical users have been issued smart cards. They use the cards routinely with no complaint (even though they are optional).

Access Manager Authentication
The end-user portal authentication experience is straightforward. All web and application resources that are protected by a Policy Agent rely on the Access Manager to provide authentication and authorization. Therefore, accessing any protected resource results in the same experience.

The only variation is whether the client has previously authenticated to the Access Manager and still owns a valid session.

The general end-user experience can be described as follows:

1. The user starts a new browser and points it at https://www.anl.gov/protected/.

2. The Policy Agent uses information from a requested cookie and the Access Manager to determine whether access can be granted.
3. If access can be granted, the user is granted access to the resource.
4. If access cannot be granted (no session), the user is redirected to the Access Manager to provide credentials.
5. The user provides a certificate to the Access Manager.
6. The user is redirected to the protected resource, and the process resumes at Step 2

By default, Internet Explorer will automatically present a certificate in the certificate store to a web site that requests one, assuming that only one certificate is present. Most Argonne users have only the auto-enroll certificate available in their certificate store. Therefore, when such users contact the Access Manager for authentication, the authentication process begins immediately; no user action is required. The Access Manager accepts the certificate and creates a user session.

Authentication is virtually the same if the certificate is contained on a smart card. The only difference is that the user is prompted for the PIN so that the private key functions of the card may be used.

During an average business day at Argonne, roughly 1,000 users will authenticate to the Access Manager. Half of these users authenticate by using a certificate.

Conclusions

Argonne National Laboratory's deployment of a certificate-enabled infrastructure and portal technology addresses the two vexing challenges associated with enabling certificates for authentication:

- Providing certificates to users, and
- Enabling applications to accept certificates for authentication.

The result is that Argonne employees routinely and transparently use certificates to access Laboratory applications.

Argonne succeeded in this endeavor by successfully leveraging and integrating a number of authentication and related technologies to use certificates in a real-world, end-user environment. Certificate deployment was accomplished by using two types of technology:

- Short-term user certificates issued via Microsoft's auto-enroll technology or the University of Michigan's KX.509 suite, and
- Long-term user certificates contained on smart cards.

We enabled the applications to accept certificates by adopting the Sun Java Enterprise System. The Microsoft, Sun Microsystems, and University of Michigan products demonstrate daily that certificate authentication — even end-to-end certificate authentication — is doable today.

Argonne has accrued several specific benefits through its certificate-enabled infrastructure, as described below.

Benefits

Approach Enables Single Sign-On for Users

In Argonne's authentication infrastructure, one authentication credential provides access to many applications. Successfully obtaining a Kerberos ticket permits the user to obtain a certificate (auto-enroll, KX.509). On the other hand, possessing a valid certificate allows the user to obtain a Kerberos ticket (smart card). At the completion of login, the user possesses both types of credentials and can immediately interact with downstream applications requiring either authentication technology.

Short-Term Certificates Eliminate User Certificate Management Burdens

The Microsoft auto-enroll and the University of Michigan KX.509

technologies provide a quick and simple way to issue certificates to users and thereby accelerate certificate deployment. These tools eliminate many of the burdens of certificate management for the user — such as issuance, private key management, and renewal. Overnight, users can be certificate-enabled.

Short-Term Certificates Jump Start Application Certificate Enablement
Organizations should consider using Microsoft's auto-enroll technology to allow rapid deployment of applications that use certificate authentication. Auto-enroll technology allows certificate deployment to be decoupled from application enablement. Organizations can benefit from certificate-enabled application without having to address the burdensome aspects of certificate deployment.

With auto-enroll technology, organizations can prepare their applications now for HSPD-12 certificate availability.

Approach Provides Universal Application Access
Argonne has made use of proprietary, open-source, and standard protocols and technologies to enable employees using various desktop operating systems to access Laboratory applications by using certificates. Whether the end-user is running Windows, Linux, Solaris, or Macintosh, there is a method to acquire a certificate and import it into a browser for use in portal applications.

Applications Are Authentication-Neutral
The applications provided in the portal are authentication neutral. The application authentication process is centralized through the Sun Access Manager, which provides the flexibility to support future authentication mechanisms without making changes to the applications that depend on authentication. A certificate can be used for authentication just as easily as a user name and password. The application only knows that the user has authenticated. The manner in which authentication occurred is not critical.

Approach Allows End-to-End Certificate Authentication
Via smart cards, Argonne is providing end-to-end SSO based on certificates. That is, users rely totally on certificates for authentication; they never present a user name and password. Indeed, in the future, users will not have passwords.

Smart Cards Allow Functional Certificate Portability
Adding smart cards to a certificate instantly enables certificate portability. Microsoft Windows manages smart-card-stored certificates as seamlessly as it manages internally stored certificates. Users find that smart cards have a negligible impact on workplace efficiency and permit certificate authentication from any workstation. In a properly equipped environment, the certificate is as portable as the user name and password.

Approach Increases HSPD-12 Readiness
An outcome of smart card deployment, as required by HSPD-12, is that users possess portable long-lived certificates. The deployment of portal technology has positioned Argonne for the availability HSPD-12-compliant smart cards and certificates. The versatility of X.509 authentication included in the Sun Access Manager enables Argonne to readily accept an HSPD-12 certificate for application authentication.

Lessons Learned

Two-Factor Authentication Can Enable SSO
Two-factor authentication requirements that require smart cards (e.g., HSPD-12) can be a vehicle for user certificate deployment. Argonne's smart card deployment has shown that users can readily employ smart cards for client authentication and subsequently use the same smart card for application access. Smart cards and their supporting software readily interact with

browsers. The user impact of performing a smart card login is negligible.

Auto-Enroll Certificates Complement Smart Cards

Argonne has realized the benefits of issuing auto-enroll certificates to users who log in with smart cards, especially users who must frequently authenticate to applications. Application authentication via certificates requires access to the private key. Repeated presentation of the smart card becomes burdensome to the user. Issuing an auto-enroll certificate to smart card users permits two-factor authentication for the initial login without requiring two-factor authentication for each successive application.

Automation Is Key to Certificate Usage

As noted above, Argonne's Administrative Systems Portal processes 500 certificate authentications per day. Argonne's KX.509 KCA servers issue fewer than two short-term certificates per day. The vast majority of application certificate authentications rely on auto-enroll certificates.

So, although only two commands (kinit and kx509) are required by users to enable certificate authentication — and thus SSO — from Linux and Mac desktops to Argonne's Administrative Portal, these users continue to rely on user name and password. It is clear that certificate acceptance is achieved through automation of certificate issuance and management.

Issues

Undesirable SSO

The concept of using a certificate instead of a user name and password for authentication is new to many users. Confusion can arise when a user wishes to access an application as another user. For example, a Human Resources representative may wish to have an employee who is sitting in his or her office log in to an application. When the application is accessed, the Human Resources representative is automatically logged in because the certificate contained in the user's profile is automatically presented by Microsoft's Internet Explorer to the Sun Access Manager. As a result, Argonne Human Resources representatives do not receive auto-enroll certificates.

User education is therefore an important part of certificate rollout. Confusion may arise unless an employee understands how authentication is occurring and how to change default browser behavior.

System Administrator Skills

Similarly, system administrators are generally unfamiliar with certificates and public key infrastructure concepts. Technical staff charged with maintaining the desktop environment and supporting users may not understand the roles of certificate authorities, certificates, and smart cards. Too often, system administrators equate the smart card PIN to the user's password.

The challenges that system administrators face will increase with the adoption of HSPD-12 smart cards for authentication. Today at Argonne, the certificate authority is local, and certificate issues can be addressed by on-site staff. As Argonne accepts externally signed certificates for authentication, its staff must be prepared to work with external service providers to address real-time authentication issues. During Argonne's smart card pilot program, staff experienced the absence of a valid CRL, which disables Microsoft smart card login. Under HSPD-12, organizations will have to depend on external information sources for authentication.

Smart Card Certificate Requirements

The current requirement that the *Subject Alternate Name* field of the smart card certificate contain the *User Principal Name* of the user prevents the card from being used for desktop logins in other Windows domains. As described earlier, the UPN is username@domain, and a domain controller cannot normally establish a session for a user of another domain. Microsoft is expected to remove the requirement for the

UPN in the next version of Microsoft Windows.

Related Work

PIV Smart Card Support

Argonne National Laboratory sees great value in having a broadly trusted and interoperable identity credential as envisioned by HSPD-12. The Laboratory has obtained NIST SP 800-73-1-compliant smart cards from vendors and has developed enhancements to the Open Source Smart Card Project (OpenSC) to enable Linux and Mac platforms to use the certificates contained on these cards. These PIV enhancements have been donated to OpenSC.

OpenSC provides a PKCS#11 interface, thus making the smart card available for Kerberos login via PKINIT. For example, the Heimdal Kerberos with PKINIT enables smart card login for open systems such as Linux.

Acknowledgements

This work was produced with funding from the U.S. Department of Energy's Office of Science under Contract Number DE-AC02-06CH11357.

The authors would like to thank Gemalto, MobileMind, and Oberthur for providing sample PIV cards.

References

Butler, R., D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, "A National-Scale Authentication Infrastructure," *IEEE Computer*, 33(12):60–66, December 2000.

Doster, W., M. Watts, and D. Hyde, "The KX.509 Protocol," University of Michigan, Ann Arbor, MI, 2001 (http://www.citi.umich.edu/techreports/reports/citi-tr-01-2.pdf).

Foster, I., and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit." *Intl J. Supercomputer Applications*, 11(2):115–128, 1997.

Heimdal Kerberos Implementation (http://www.pdc.kth.se/heimdal/), October 5, 2006.

Komar, B., "Microsoft Windows Server 2003 PKI and Certificate Security," Microsoft PKI Team, Microsoft Press, Redmond, WA, 2004.

Microsoft Corporation, "2821A: Designing and Managing a Microsoft Windows Public Key Infrastructure," Microsoft Official Curriculum 2821A, Redmond, WA, 2003.

Neuman, B.C., and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications*, 32(9):33–38, September 1994.

Open Smart Card Project (http://www.opensc-project.org/), October 5, 2006.

RSA Laboratories, *PKCS #11 v2.20: Cryptographic Token Interface Standard*, Bedford, MA, June 2004.

Sun Documentation (http://docs.sun.com/, http://docs.sun.com/app/docs/prod/entsys.05q4#hic).

Sun Java Enterprise Server 2005Q4, (http://www.sun.com/software/javaenterprisesystem/index.xml).

U.S. Department of Homeland Security, *Homeland Security Presidential Directive (HSPD-12): Policy for a Common Identification Standard for Federal Employees and Contractors* (http://www.whitehouse.gov/news/releases/2004/08/20040827-8.html), August 27, 2004.

Zhu, L., and B. Tung, *RFC 4556 Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)*, The Internet Society, June 2006.

[1] Microsoft offers both Standard Certificate Services and Enterprise Certificate Services. All of the Microsoft functionality discussed in this paper is achieved via the Enterprise Certificate Services.

[2] The selection of smart cards and corresponding middleware is in itself a research undertaking and so outside the scope of this paper. The smart card market is fragmented and proprietary. Homeland Security Presidential Directive 12 (HSPD-12) and Federal Information Processing Standard 201 (FIPS-201) will enable significant improvement in the standardization and interoperability of smart cards.

[3] The Access Manager uses an internal list of trusted root certificates to validate user certificates; it does not rely on root certificates contained in Active Directory.

Temporal Key Release Infrastructure

Ricardo Felipe Custódio * Júlio da Silva Dias † Fernando Carlos Pereira ‡
Adriana Elissa Notoya §

Abstract

Timed release of confidential information, where information is revealed at the date and time established by the author, is a security requirement in applications such as auctions, wills, and government buying processes. We have found that this security requirement is achieved through the fulfillment of a group of requirements that are not completely understood. We propose the development of an infrastructure that enables the fulfillment of the studied security requirements. The infrastructure, Temporal Key Release Infrastructure (TKRI), was developed as a proof of concept. We also discuss the advantages of our approach against other proposals.

1 Introduction

Electronic documents have been used to substitute paper documents in many computer applications and information systems. The ease of their use, transmission, and storage has motivated this substitution. However, in some practical situations, the use of electronic documents is possible only with some security assurances[1]. These security assurances are authenticity, integrity, non-repudiation, and confidentiality. Additionally, the use of time-related evidence is also necessary. Authentication and integrity are assured through cryptographic methods such as hash functions in conjunction with digital signatures schemes [15, pg. 425]. The non-repudiation assurance is provided through non-retractability and non-refutability, and is claimed to be technologically fulfilled through authentication [14, 2]. Confidentiality, is achieved using symmetric cryptographic algorithms [24, pg. 44]. The time-related evidence is the precise time at which the existence of an electronic document can be proved, along with the fact that it has not been modified since that time. This property is provided by a trusted third party, an entity called Time Stamp Authority (TSA), that issues a time stamp [11, 4, 7]. This work deals with timed release of the document content. It is a special case of the confidentiality security requirement where the information is to be revealed only at the moment established by the author.

There are three different aspects to consider when dealing with paper documents confidentiality: transportation in order to guarantee confidentiality during the communication process, the sealed envelope as a way to guarantee document content confidentiality, and timed release of the document content to address the act of opening the sealed document at the time and date established by the author.

Exactly the above aspects also arise when dealing with electronic documents. First, the communication protocols usually employed in data communications use cryptographic techniques in the transport or network layers, which are similar to those of the ISO/OSI model such as TCP/IP. The best known protocols are IPSec in the network layer and SSL/TLS [8] in the transport layer.

Second, secure electronic document storage can be obtained using a trusted third party or symmetric and asymmetric cryptographic techniques. A trusted third party receives the document, stores it, and only reveals its contents after the required authentication is provided. This approach presents high cost and risk because it is necessary to trust that the third party will be honest, will maintain the document's integrity, and will allow access to authorized entities only. The use of cryptographic techniques allows those interested in providing confidentiality to a document to encrypt and store it securely. However, care must be taken because the loss of the encryption key renders

*R. F. Custódio is with Departamento de Informática e de Estatística, Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brazil. E-mail: custodio@inf.ufsc.br.

†J. S. Dias is with Universidade do Estado de Santa Catarina (UDESC), Florianópolis, Brazil, E-mail: jdias@inf.ufsc.br

‡F. C. Pereira is with PGEEL, Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brazil. E-mail: fcarlos@inf.ufsc.br

§A. E. Notoya is with PGEEL, Universidade Federal de Santa Catarina (UFSC), Florianópolis, Brazil. E-mail: elissa@inf.ufsc.br

[1]In Brazil, the legal validity of the electronic documents is based on the fulfilment of security assurances. However, in some applications or countries, the observance of these assurances cannot be obligatory for the use of electronic document.

the electronic document illegible. Therefore, the encryption key must be securely stored for the entire period of time over which confidentiality is required. We can use various approaches to solve this problem. One strategy consists of encrypting the document with the public key from a trusted third party. In this situation, the RP may lose the private key, and the trusted third party can be contacted in order to decrypt the document. Another approach consists of an encryption of the document with a public key whose private key is distributed to an authorized group of people, in parts, using of secret sharing techniques. Using these techniques, the RP could lose the private key and still submit a request to an authorized sub-group of the complete group that has received parts of the key, for the reconstruction of the original decryption key.

The third aspect of confidentiality in paper documents that needs to be considered in electronic ones is timed release of documents. As with storage, this requirement can be achieved with the use of cryptographic techniques or trusted third parties. Trusted third parties are trusted to only release the document content at the specified time. This solution presents high risk and cost. This is because in order for the third party to be trusted, they are required to have both the ability and the robustness necessary to maintain the document's integrity and confidentiality until the time of release. When using cryptographic techniques, the problem is reduced to keeping secret the decryption key until the time of release. After the key has been released, the document can be read.

Applications which require document confidentiality, such as public and private buying processes, electronic auctions, e-voting schemes and wills, require secure storage and later release of electronic document content. In all this applications the unauthorized release of the information before the specified time can invalidate the whole process. In order to make the document confidential it is necessary to encrypt the document and keep the decryption key secret. The existing solutions used to perform this task employ complex and expensive systems such as trustworthy environments and specialized computer platforms to keep the decryption key secret.

Despite the importance of the fulfillment of the confidentiality security requirement and timed release of documents, little effort has been made to provide a simple and inexpensive infrastructure. It is not possible in present applications identify document contents or modify document usage policies during the document life cilcle. To achieve this aim it is necessary to carry out a rigorous study of the security requirements, as well as the services which a proposed infrastructure would need to offer. The aim of this paper is to present the security requirements and propose an infrastructure for electronic document temporal confidentiality.

The rest of this paper is organized as follows. Section 2 states the problem. In the subsection 2.1 is showed the security requirements necessary to achieve temporal confidentiality, obtained through the study of applications in which the controlled storage, transport and release of the document content is essential. In the subsection 2.2 is presented a review of related studies that apply cryptographic techniques to achieve timed release of electronic documents. Confidentiality is achieved in this proposal through the encryption of the electronic document. An encryption module is proposed and presented in section 3 to allow fast and secure encryption following policies established by the document author. The infrastructure for the temporal confidentiality of electronic documents is presented in section 4. In section 5, an implementation of the infrastructure that was developed is presented. In section 6, the results of the study are discussed and improvement possibilities are presented. The appendix A summarizes the notation and symbols used throughout this paper.

2 Problem Statement

To date there is no a definitive solution for temporal electronic documents confidentiality as we have with paper documents. We show in this paper, a new way to analyze and propose a different solution to this problem. Our approach consists in emulate in the electronic world all features we have with a paper envelope. Then our problem is to present the security requirements and propose an infrastructure for electronic document temporal confidentiality with the same requirements.

2.1 Security Requirements

The establishment of an infrastructure for the temporal confidentiality of electronic documents begins with the definition of the security requirements that must be fulfilled by this infrastructure. This is an essential step to guarantee the correct utilization of the services provided. In order to determine these security requirements, we have studied applications in which electronic documents must be kept confidential for specified periods of time, such as secret price proposals in public bidding processes [20] and wills in notary services [3]. Both applications can be compared to a paper document in a sealed envelope. After sealing the envelope, the document content is confidential and will only be released when the seal is broken and the envelope opened at the specified time.

This study leads us to the following general requirements for electronic or paper documents. Additionally, where appropriate, we add what a specific requirement if the means of implementing the temporal confidentiality on an electronic document is through the use of encryption.

r1. After the document was sealed, it must not be possible to determine its content before the specified time of release;

    (a) The decryption key that allows access to the document content cannot be known before the specified time of release;
    (b) It must be possible to control access to the document content;
    (c) The decryption key must be given only to the authorized entities;
    (d) It is necessary a mechanism to show the public part of the electronic document;

r2. Once the document is released, the entity having the document cannot deny knowledge of the document content;

r3. It must be possible to prove, after the decryption key was published, that the document content has been revealed;

r4. It must be possible to destroy the document without accessing its content;

r5. It must be possible to determine the group of users that witnessed the opening of the document;

r6. It must be possible to verify in a non-repudiable form the authenticity and integrity of the document. After being revealed, the document must be authentic and its content must be related and equal to that provided by the author;

r7. It must be possible to audit the activities performed by the entities involved as well as to audit the resources used.

These security requirements are general and some of them can be waived or may not be necessary for some applications.

2.2 Timed-release Cryptography

Timothy May [13] was the first author to use cryptography to address timed release of electronic documents, using the term *timed-release cryptography* to discuss this problem. May has presented as a solution the use of trusted third parties (TTP) to store and release the document at a specified time. In addition, he proposed to encrypt the document maintaining secret the cryptographic decryption key. For economy of performance, scale and resource, the second approach is the most common choice. In this approach, the decryption key must be kept secret by the TTP until the time specified by the author. The TTP receives the decryption key and the time of release of the document and agrees to keep the key secret until the date and time specified. Thus, the TTP is most sensitive and most vulnerable to threats.

There are two main approaches to increasing trust in the TTP with respect to key encryption and timed release. In the first approach, the decryption key is produced as the solution of a problem whose resolution time is known [21]. The challenge in this approach is to construct an algorithm whose resolution is as independent as possible of the computer processing power and memory available and takes exactly the time specified to reveal the document content. Following this approach, Ronald Rivest [21] has proposed to model the problem as a time-lock puzzle. The puzzle is designed so as to perform only sequential tasks not allowing parallel operations. The puzzle can only be solved after the execution of a series of sequential steps, where each step takes a constant time t to be executed. The total time $T = nt$ will be the time specified to release the decryption key necessary to reveal the document content. However, the puzzle proposed in the literature depends on the computer processing power and memory available to the computer. An alternative solution is the insertion of the puzzle into a sealed computer system. The process must be carried out in such a way that it is not possible to have access to intermediate results without destroying the puzzle. The computer must be kept in a secure and monitored environment. After the time specified the computer will reveal the decryption key to allow the release of the document content. The most well-known implementation of this approach is the time capsule *LCS35* built by Rivest [22]. Wenbo Mao [12] has improved Rivests work, proposing a technique to control execution times of the puzzle. In this way, the puzzle is parameterized in terms of floating point operations that have almost constant execution times. The control of floating point operations allows better control of the total execution time of the puzzle.

The second approach to increase trust in the process is to share the key using secret sharing techniques [25, 26]. In this technique, the decryption key will be divided into pieces, each called a share. Each share is given to a different TTP, which agrees to release it at a specified date and time. The reconstruction of the decryption key is permitted by an authorized subgroup of TTP, which must agree on releasing its shares to reconstruct the key. If we have honest TTP subgroups, we can guarantee that the decryption key will be released at the programmed time. Although it is possible to control the exact time of the decryption key release, we cannot guarantee the correct decryption key reconstruction. There must be additional procedures to ensure that the shares released are in fact those, which were produced when the secret was shared [10]. A system developed using this approach was proposed by Fernando Pereira [20].

Marco Mont et al. [17] proposed the use of Identity-based Encryption (IBE) to address this problem. The encryption key is constructed based on a public N that the TTP has published, the identity of the client and the time that the corresponding decryption key will be released. On a certain date, at a specified time, the TTP will calculate the decryption key as a function of the encryption key, the identity of the client, and the time of release. As an advantage, the decryption key is only calculated at the time it is released. The drawback to this approach is the necessity to protect the parameters, for instance, the master key, needed to calculate the decryption key. If one of these parameters is published or becomes public, so will all the decryption keys. This property makes it difficult to build such an infrastructure. Another drawback is the lack of standards related to Identifier-based Encryption (IBE).

Recently, Aldar C. -F. Chan and Ian F. Blake[5] proposed that the TTP be completely passive without interaction between it and the RO or RP, thus assuring the privacy of the document and the anonymity of both its RO and RP.

The present study was developed using known asymmetric cryptosystems such as RSA and standards like X.509v3 digital certificates [1] and PKCS [23] to represent key pairs.

3 Encryption Module

The electronic document encryption may be performed with the aid of symmetric cryptography, since this kind of algorithm is more efficient than asymmetric algorithms. A session key K_S, obtained with the use of a random number generator may be used in the symmetric cipher. This key would then be encrypted using the RPs public key. In this concept, only the RP, who is the owner of the corresponding private key, can decrypt the session key and then decrypt the document. This approach is well-known and used in many information systems to encrypt documents.

A detailed analysis of this scheme shows that it is not an efficient way to achieve the security requirements listed in section 2.1. This is because the user who encrypts the document can keep a copy of K_S. In this case, any entity, regardless of whether they have the decryption key necessary to release K_S or not, can decrypt the document using K_S. To solve this problem and allow more flexible encryption policies, we propose the use of an Encryption Module (EM). EM is a hardware and it must be developed according to FIPS140-2 secure cryptographic module recommendations [18]. This precludes access to information that could lead to the disclosure of the decryption keys, or loss of sensitive data from the EM. The basic EM is presented in Figure 1 and receives the document (DOC) to be encrypted, along with

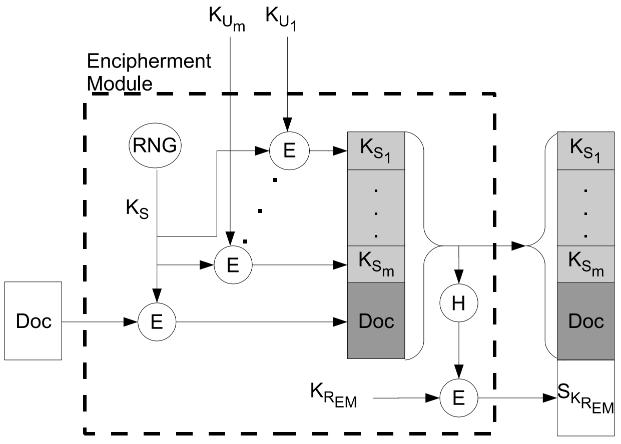


Figure 1: Basic Encryption Module.

digital certificates from the users who would decrypt the document once the decryption key has been released. The EM, using a Random Number Generator (RNG), generates a symmetric key K_S used to encrypt the document DOC, creating $E_{K_S}[DOC]$. The symmetric key K_S is then encrypted using asymmetric techniques for each of the m public keys K_{U_i} supplied by the client, and then destroyed. The different files containing encrypted $K_S s$ are then attached to the encrypted DOC, creating the encrypted document $DS = [E_{K_{U_1}}[K_S], ..., E_{K_{U_m}}[K_S], E_{K_S}[DOC]]$. Then DS and its digital signature $E_{K_{R_{EM}}}[H(DS)]$, are sent to the user that requested the encryption. In all figures of this paper, a shaded rectangle represents a encrypted data.

Only user who has K_{R_i} is able to access the content of the electronic document encrypted using the K_{U_i} public key used to encrypt K_S, as showed by Figure 2. The user having K_S can decrypt DOC but he can

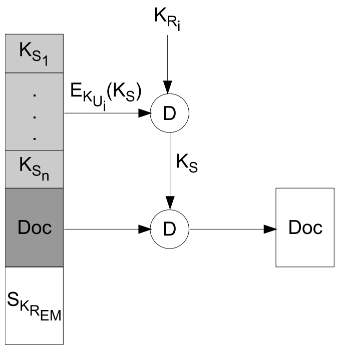


Figure 2: Electronic Document Disclosure When Using the Basic EM

also verify the DOC integrity through the EM digital signature attached to DOC.

However, this mechanism cannot guarantee the achievement of all functional requirements necessary for applications that require document confidentiality. There are applications where the document can only be decrypted under conditions previously specified by the decryption policies established for each document. One alternative to improve the encryption process is to aggregate secret sharing schemes. Let P be a set of participants. The decryption key is segmented in many parts called shares and each segment is given to a different entity of P. Let Γ be a set of subsets of P. The elements of Γ are the subsets of P capable of reconstructing the decryption key. Γ is called the access structure and its elements are the authorized subsets. The decryption key reconstruction is only possible with the participants of an authorized sub-group $B \subseteq P$ pooling their shares[26]. Many different algorithms have been proposed with respect to this problem. The simplest secret sharing scheme shares the secret in m parts using an exclusive-OR $\bigoplus$ operation. Let x be the secret to be shared in m parts, and y_i numbers randomly generated, with $i = 1 \dots m-1$. $z = x \bigoplus y_1 \bigoplus \dots \bigoplus y_{m-1}$ will be calculated. The shares are $z, y_1, ..., y_{m-1}$. To reconstruct the secret we calculate $x = z \bigoplus y_1 \bigoplus \dots \bigoplus y_{m-1}$. In this scheme $B = P$, and is the only element of Γ which implies that all participants must agree and contribute to the secret reconstruction. Another well-known scheme is based on polynomial interpolation [25]. In this approach a polynomial $F(x) = a_0 + a_1.x + ... + a_{n-1}.x^{n-1}$ of degree $(n-1)$ is constructed in such way that the coefficient a_0 is the secret to be shared. Let $m \geq n$ be the number of participants that will receive the shares. Each participant receives a point $(F(x_i), x_i)$ called a share with $x_i \neq 0$. To reconstruct the secret, n participants are required. With these n points, the secret a_0 can be reconstructed through polynomial interpolation.

Figure 3 presents an architecture for the EM that uses secret sharing schemes. An aspect that must be

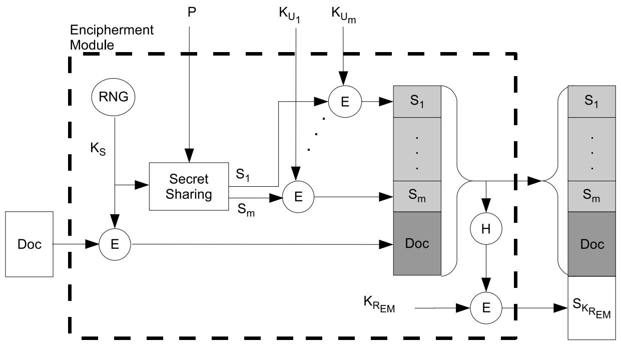


Figure 3: Secret Sharing Encryption Module

considered in secret sharing schemes is the possibility to verify the generated and disclosed shares. The secret sharing schemes using exclusive-OR or polynomial interpolation do not address this problem. It is necessary to use techniques such as the one proposed by Gennaro [9] to guarantee verifiability. However, in our approach, the encrypted parts are signed by the trusted EM so we do not need more elaborate schemes. The disclosure process of DOC is presented in Figure 4. The RP can verify the EM digital signature before choosing the set of encrypted shares. The private key owners decrypt the shares and send

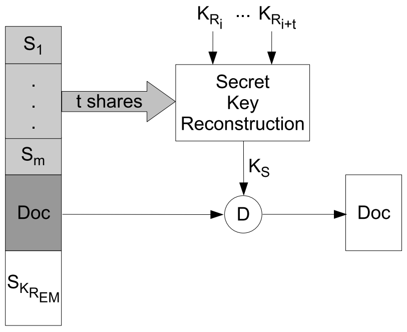


Figure 4: Electronic Document Disclosure When Using Secret Sharing EM.

the result to the user in charge of reconstructing the session key. The user that receives the shares can verify their validity using the public keys from the users that sent the shares.

3.1 Stamp

In order to enable a better control of the confidentiality the electronic document, the use of external information called a stamp is proposed. The cryptographic hash value from the stamp is combined with the session key K_S as presented in Figure 5. The cryptographic hash value from this combination, K_S^*,

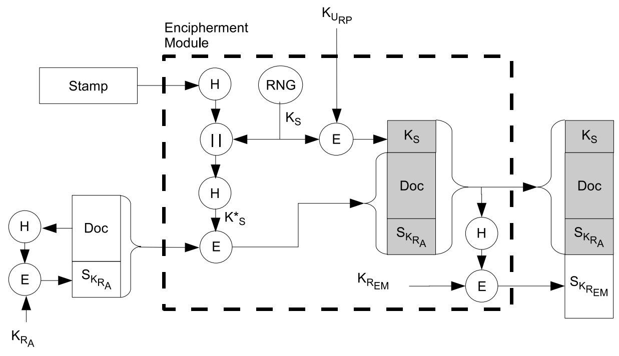


Figure 5: Stamp Insertion

is used as the session key to encrypt *DOC*. Note that it is necessary to have both K_S and the stamp in order to have access to the session key K_S^* that can be used to decrypt *DOC*.

The stamp can be kept secret and its owner can give it only to the authorized entities or it can be public, since the stamp knowledge is not sufficient to have access to K_S or K_S^*. Figure 6 shows how to disclose the document using a stamp.

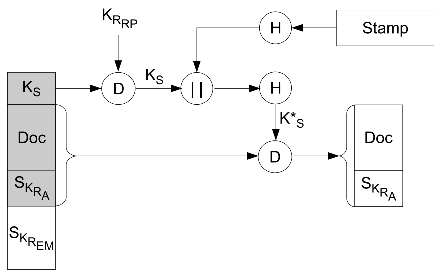


Figure 6: Electronic Document Disclosure When Using Stamp

Another approach is to use the stamp as an access control mechanism for the services provided by the infrastructure. In this way, the stamp, which can be made public, would be issued by the entity that controls the service being provided. The stamp would be obtained from this entity and would be used by the EM in the session key generation. Only signed stamps would be validated by the EM and would allow access to the encryption service. This provides a second mechanism to control the access to the *DOC* content and accountability.

3.2 Public Windows

Public windows, as illustrated in Figure 7, are supplied to attach public information regarding the en-

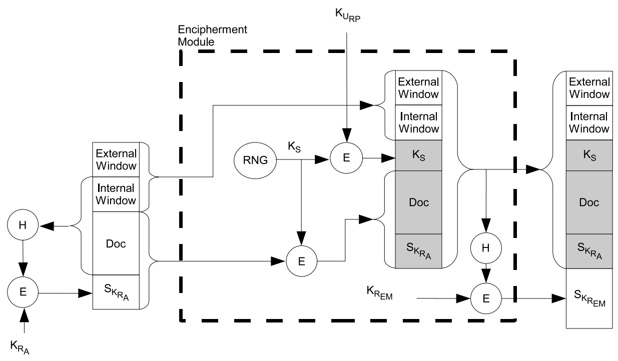


Figure 7: Electronic Document Information Window

crypted document subject. There are two types of windows: internal and external. The internal window is the part of the document that the author decides may be provided to the public. The external window is an electronic document that is attached to the original document and is not encrypted, allowing the user to read its content. These windows are present and are readable in the encrypted document. The internal window is bound to the electronic document *DOC* using a digital signature supplied by the author of the document. This signature is part of the document that is encrypted by the EM. The external window is bound to the sealed document through the digital signature by the EM using its private key $K_{R_{EM}}$. The sealed envelope contains the readable internal and external windows contents that can be accessed without any cryptographic operations. Figure 8 shows how to decrypt and verify the authenticity of the electronic document.

3.3 Complete Encryption Module

The complete encryption module illustrated in Figure 9 presents all the functionalities of the basic, se-

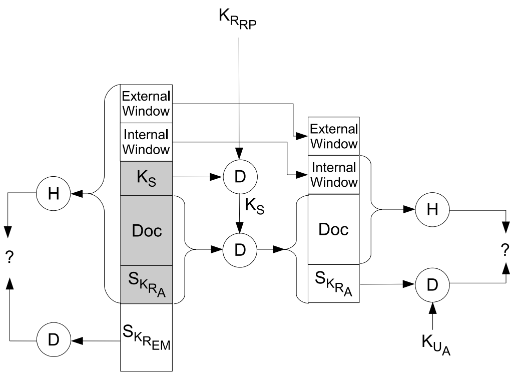


Figure 8: Window Verification.

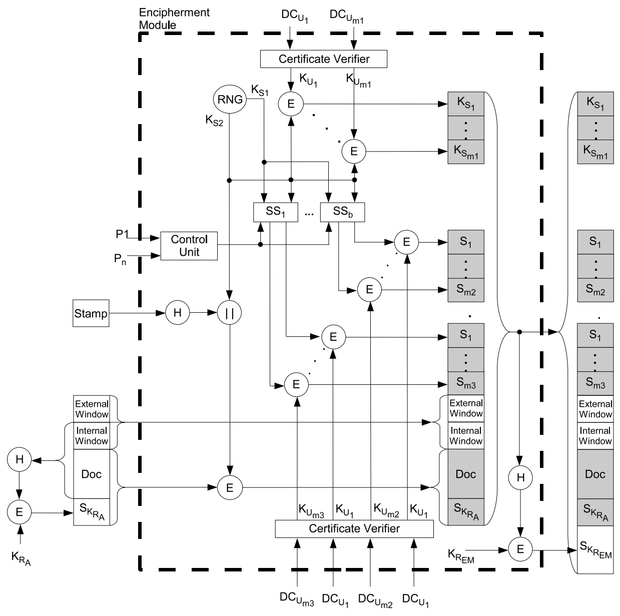


Figure 9: Complete Encryption Module

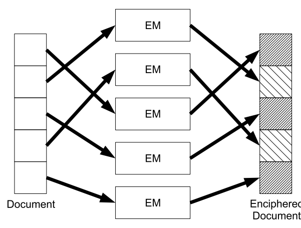


Figure 10: Use of Multiple EM

cret sharing, stamps and public windows encryption modules. The complete EM carries out the session key generation using external control mechanisms, the stamps. On the other hand, the insertion of public windows, internal or external, allows readable information to be attached to the encrypted document.

Module flexibility is achieved using a Control Unit that receives the encryption requests and a set of policies. This Control Unit is responsible for interpreting the policies and generating operational instructions for all the elements within the module. Therefore, the policies allow the user to establish the EM operational behavior.

It would be possible for a malicious EM to disclose document contents to other entities besides those authorized. We could also suppose that a malicious EM could disclose the document. Our model does not allow generation of evidence that could lead to disclosure of the electronic document by the author, the EM, or the computational platform. In order to address these threats, we propose that the document be partitioned and each part sent to a different EM. We propose the use of client software that divides the document into small blocks and submits then randomly to different EMs as presented in Figure 10, using a secure channel achieving communication confidentiality. Once the client has received all the encrypted blocks from the different EMs, these are tied together to form the encrypted document that is sent to the RPs.

In this way, the EM never has access to the complete electronic document content. The access to the complete document is possible only with all the EMs work together to defraud the system, which is not an easy task.

Encryption module management is achieved through configuration and auditing processes, carried out by the module manager, MG, only. The first time the module is turned on, the module owner must define the manager and users of the module. User authentication can be performed using passwords or digital certificates. Once the MG has been created, this user can begin the cryptographic key pair generation that is done in two steps.

In the first step, the asymmetric key pair of the EM is generated along with a certificate request formatted as a PKCS#10 file containing the public key. This request is sent to a Certification Authority that issues the X.509V3 digital certificate. The second step is the digital certificate importation by the EM. The private key will always be kept secret and will never leave the module. This private key is used by the EM to sign the encrypted data. The EM must also import the trusted Certification Authorities digital certificates as well as its Certificate Revocation Lists. Once this step is finished, MG specifies the operational policies. These policies can activate or not activate EM functions as required by the applications that will request EM services.

Periodically, MG must perform tasks such as the renewal of the digital certificate, the verification of the EM operational status, the creation of new users, the management of the trusted certification authorities, and auditing processes if required.

All the information required to verify the digital certificate validity must be supplied by the user requesting service to the EM, so it will not be necessary for the module to access any external entities. The Certificate Verifier in the EM is responsible for this task.

4 Temporal Key Release Infrastructure

The aim of the Temporal Key Release Infrastructure (TKRI) is the management of temporal digital certificates. The temporal certification authority (TCA), one of the infrastructure elements, generates a pair of asymmetric cryptographic keys. The private key is stored and kept secret until a future specified date when it is published. The public key is embedded in an electronic document called temporal digital certificate (TDC) as specified by X.509v3 standard. Users can use this digital certificate to encrypt documents. Once encrypted, the document, and its contents, will only be known when the private key is published.

The TKRI architecture is presented in Figure 11. We have three main components: TCA, working on-

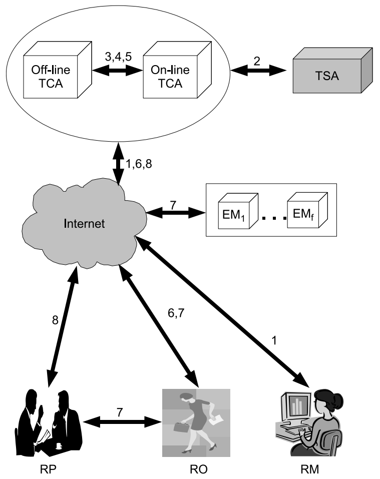


Figure 11: TKRI Architecture

line or off-line, which is responsible for the cryptographic key pair and the temporal digital certificate generation; EM, responsible for the secure encryption of the electronic document and the TSA, used to attach a time record token to the events and actions occurring in the TCA. Aspects about the security of these components are not examined in this work. However, related topics like safe code execution and hardware security modules, used to provide such security, are objects of study in parallel projects lead by our research group[2].

Additionaly, there are seven entities involved in the management, operation and use of a TKRI: the managers MG and operators OP of on-line and off-line TCAs, the request managers RM, the request originators (RO), who have an interest in achieving confidentiality, and RP to whom the encrypted electronic document is destined.

The off-line TCA is responsible for the generation of a long-term cryptographic key pair for the issuing of TDCs, for safe private key storage, and for event log auditing. This entity is kept isolated, usually in a safe room, without data communication connections. In this way, the off-line TCA is not subjected to threats from public data communication channels. Data insertion and retrieval in the off-line TCA are carried out using removable media.

The on-line TCA is responsible for the short-term cryptographic key pair generation, issuing of TDCs, and private key storage. The on-line TCA is also in charge of receiving the TDC requests and its policies and disclosing the private key at the time specified. The private keys that will be necessary in the long-term are sent to the off-line TCA. The on-line TCA can also receive private key publication delays or private key destruction requests. This entity must produce audit logs of its activities and publish them with the audit logs from the off-line TCA.

The TSA issues time stamps that are attached to the electronic document and thus can prove its existence at a specific time [11, 19, 6]. The TSA, in this infrastructure, is in charge of producing time-related evidence for the electronic documents received or sent by the on-line and off-line TCA. The TSA clock is synchronized with trusted-time sources [16, 7].

The use of the TSA guarantees that the TCA clock will be synchronized and all the transactions carried out will be associated with the correct date and time, bringing trust to the temporal aspects of the process.

As the on-line TCA is in charge of private key disclosure, its clock must be synchronized with a trusted time source. The off-line TCA does not require the same precision. The private keys are released in blocks by the off-line TCA to the on-line TCA. Each block has a group of private keys that must be dis-

[2]LabSEC - Computer Security Lab (www.labsec.ufsc.br)

closed in the next window of time to release private keys. The window of time is the time in which the on-line TCA can publish each one of the private keys stored. This period of time is previously configured by the MG of on-line TCA. Once all the necessary requirements for the private key disclosure are satisfied and the time of release has been reached, the private key is published.

The EM, described in section 3, is in charge of electronic document encryption. The confidential electronic document is sent to the EM, with the group of TDCs and decryption policies. It is essential that RO knows that he is sending the document to some known and trusted EM, but for the EM to know the client identity is not required, and in some situations it is not necessary. As described in section 3 the client can break the document into pieces that are submitted to different EM. After receiving the parts from the different EMs, the client can build a single encrypted document. This file must present all the information necessary for the RP to decrypt and access the document content at the specified time when the decryption keys are published.

The RM is in charge of requesting the issuing of the TDC from the on-line TCA, presenting all the corresponding private key release policies, authorizing the release, and delay or destruction of the private key if allowed by the release policies established.

The RO is the entity that uses the TKRI to make the electronic document confidential. The RO requests a previously issued TDC from the on-line TCA or asks the RM to request a new TDC that fulfills his requirements. When the client has the TDC, he can submit the document to the EM who encrypts, and sends it back to the client. The client having the encrypted document can send it to any authorized RP. In the cases where it will be necessary anonymity, the RO can use a generic TDC for a required time to release the associated private key. The generic TDC are a set of certificates for predefined time to release.

The RP, at the specified time, must request the private key from the on-line TCA in order to decrypt the document.

The on-line or off-line TCA manager is responsible for the TCA installation and configuration. The MG is also responsible for: OPs creating and delegating tasks to these entities; creating of the policies and defining of the operational parameters that establish the TCA operation and functionality; and backup and restoration procedures of the entities. The audit trail log is maintained by the MG and can be used by auditing and accounting procedures.

The on-line operator is in charge of tasks such as collecting requests received by the on-line TCA and the long-term private keys generated. These collected data are sent to the off-line TCA. This operator must also receives the block of TDCs and corresponding private keys from the off-line TCA to be inserted in the on-line TCA and published in the next release window.

The activities and processes related to the TCA manager and operators as well as the guarantees are present in the document TDCs Declaration Practices (TDCDP).

The TKRI is operational from the moment that we have an operational on-line TCA. The requirements of the requesters can lead to the off-line and EM deployment. The TCA deployment begins with the on-line TCA installation and configuration followed by the installation of one or more off-line TCA.

The decryption key publishing is carried out by the on-line TCA using a PKCS#12 electronic document that contains the TDC and the related private key.

4.1 TKRI Operation

To describe the TKRI operation it is necessary to understand the events associated with the life cycle of a TDC as shown in Table 1.

Table 1: Events associated with the time life of a TDC.

Time	Description
t_1	RM requests TDC.
t_2	Key pair is generated.
t_3	TDC is issued.
t_4	TDC is published.
t_5	RO requests TDC.
t_6	RO submits DOC and TDC to a set of EM.
t_7	RO receives the encrypted DOC and sends it to RP.
t_8	RM can destroy the decryption key, as stated in TDC.
t_9	TDC expires.
t_{10}	Decryption key is released.
t_{11}	DOC is decrypted by RP.

The issuing of the TDC is carried out by the TCA that generates a key pair, where the private key will be encrypted and kept secret. The public key is inserted into a public key certificate, along with information regarding the disclosure strategy and time. After being issued, the certificate can be accessed by the RO and used to make its documents confidential. These are the major steps, as shown in figure 4, of a TKRI:

1 An RM can request the issuing of a new TDC. The RM must supply, at the time of the request, the

time for the release of the decryption key related to the digital certificate to be issued, the purpose of the certificate, and the strategy to be used to release the private key;

2 All the requests received and documents issued by the TCAs are time stamped for auditing purposes;

3 The key pair related to TDC can be issued by the on-line or off-line TCA. If $(t_4 - t_0) <= L_1$ then the key pair is generated by the on-line TCA otherwise by the off-line TCA. L_1 is supplied by the TCA manager and represents the minimum time necessary to operate the off-line TCA.

4 If $(t_{10} - t_1) <= L_2$ then the decryption key is kept in the on-line TCA, otherwise it is kept in the off-line TCA. L_2 represents the minimum time necessary to manage the decryption key by the TKRI in the on-line TCA.

5 If the current time $> (t_{10} - L_2)$ then the decryption key, if it is in Key's Database of off-line TCA, must be sent to the on-line TCA;

6 Any RO interested in making its documents confidential for a specific period of time has to access the public interface of an on-line TCA, and download a TDC, prior to sending it to the encryption modules;

7 The document submission to the encryption module is performed by the RO through a program that breaks the document into blocks that can be sent to different EMs, as presented in Figure 10. The blocks are encrypted by the EM and sent back to the RO who can send it to a RP or store it. Once the document is encrypted, its contents can only be released after the disclosure of the private key digital certificate, which will be performed by the on-line TCA at the time specified by the RM;

8 The disclosure of the decryption key can be automatic, allowing direct access to the contents of the document or can be made using secret sharing techniques, allowing access only to a group of users that must agree with the disclosure of the document in order to reconstruct the decryption key.

Paper documents can be destroyed using fire or paper shredders that make it impossible for the document content to be read. In this proposal, the document RM can destroy the electronic document by asking the TCA to discard the decryption key corresponding to the public key used to make the document confidential. The decryption key is not discarded immediately, it remains stored during a period of time T after the disclosure time specified by the RM. At the disclosure time, the on-line TCA issues an electronic document advising that the private key temporal digital certificate will not be released because the RM has requested the destruction of the private key. The RM has until the end of T to request the private key disclosure. If the RM cancels the destruction before T the decryption key is immediately released, otherwise is destroyed. It is important to note that only digital certificates with the destruction possibility specified by the RM in their policies can be used this way. Thus, the RO who requests an encryption using this type of certificate is aware that the RM can negate the release of the private key.

The RP, having the encrypted document, wishing to disclose its contents must verify through the public key TDC at what time and under which conditions the document can be disclosed. At the time specified, following the instructions informed by the RM, the RP can request the PKCS#12 file with the decryption from the on-line TCA. After receiving the PKCS#12, the entity can use the private key in order to disclose the document content. Once the entity requests and receives the private key, the document is considered disclosed.

Another accepted possibility is the issuing of a private key temporal digital certificate by the on-line TCA only after an authorization is granted by the RM.

Two complementary documents are required for the operation of the system. They are: Temporal Digital Certificates Issuing Policy (TDCIP) and Temporal Certificates Practice Declaration (TDCDP). The TDCIP establishes the TCA operational constraints. In this document it is established that:

- All the off-line and on-line TCA operations must be registered;
- An issued certificate must be time stamped and this time stamp must be present along with the certificate;
- As soon as the off-line TCA receives a certificate request, this request must be time stamped and this time stamp must be present in the digital certificate as an extension.

The TDCDP presents the way that the TCA implements the TCDIP.

5 Implementation

The viability of the proposed structure was confirmed with the implementation of a TCA. The implemented system includes an on-line TCA functionality with three modules: management, client and public.

The management module allows clients configuration and managers configuration, as well as on-line TCA configuration and temporal private key control. Access to this module is allowed only to managers.

The client module implements request mechanisms and TDC management. Only registered clients can have access to this module.

The public module allows users to access the public TDCs issued by the on-line TCA. Through this module, users can have access to the corresponding private key temporal certificates. Access to this module can be restricted or not depending on the policies established by the MG.

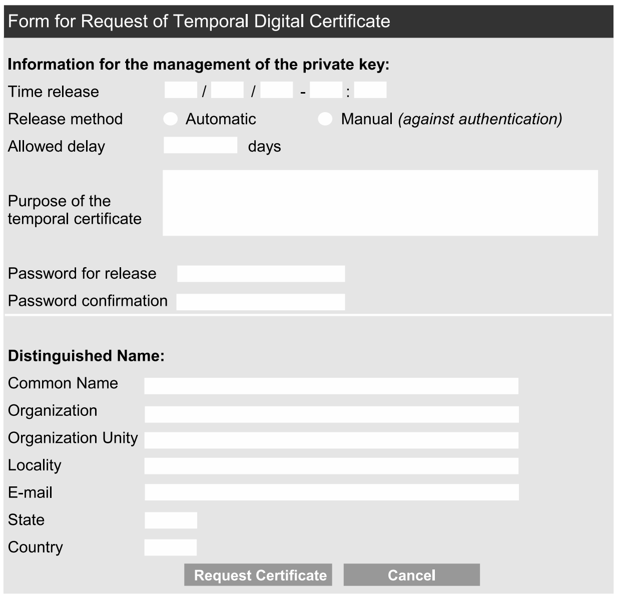
The client authentication required by the system is done through the use of digital certificates issued by the TCA. The server digital certificate allows secure communication with users using SSL/TLS protocol. The client digital certificate is used to identify the client to the server and also to determine the client access level through the extensions in the certificate.

The user enrollment is done by the TCA, using an appropriate form in the client module, in which the user data is collected for later confirmation by the TCA manager. If the data is confirmed, the MG can allow the digital certificate issuing that will permit access to the infrastructure by the user.

The public key TDC request is performed using the form presented in Figure 12, where the client can submit the necessary data to digital certificate issuing. After the TDC has been issued the MG can control the respective private key using the client module. It is possible for the MG to postpone the private key disclosure, and release or destroy the private key. However, this control is possible only following limits established by the on-line TCA policies.

The private key TDC issuing can be done in two ways. The first is through authorized client authentication, and the second is through automatic key disclosure. The prototype includes only automatic key disclosure.

The automatic disclosure mechanism verifies the on-line database once a minute in order to collect the temporal private keys to be released at that time. Once the key is collected, the system issues, for each key, a PKCS#12 file containing the temporal private key along with the public key temporal certificate. As soon as the PKCS#12 file is issued, the system generates a report describing the key pair life cycle. In this document, digitally signed by the on-line TCA, are the request, creation, destruction, and disclosure dates.


Form for Request of Temporal Digital Certificate

Information for the management of the private key:

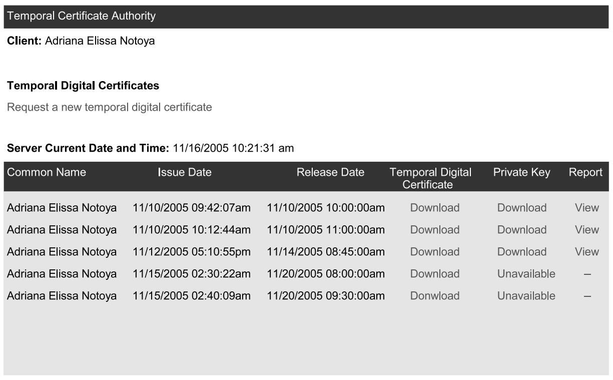
Time release	/ / - :
Release method	○ Automatic ○ Manual *(against authentication)*
Allowed delay	days
Purpose of the temporal certificate	
Password for release	
Password confirmation	

Distinguished Name:

Common Name	
Organization	
Organization Unity	
Locality	
E-mail	
State	
Country	

Figure 12: Temporal Digital Certificate Request Interface

The PCKS#12 file and the report can be downloaded by an authorized user through the client temporal certificate management interface, as presented in Figure 13, and also in the public module, if previously authorized by the solicitant of the certificate.


Temporal Certificate Authority

Client: Adriana Elissa Notoya

Temporal Digital Certificates

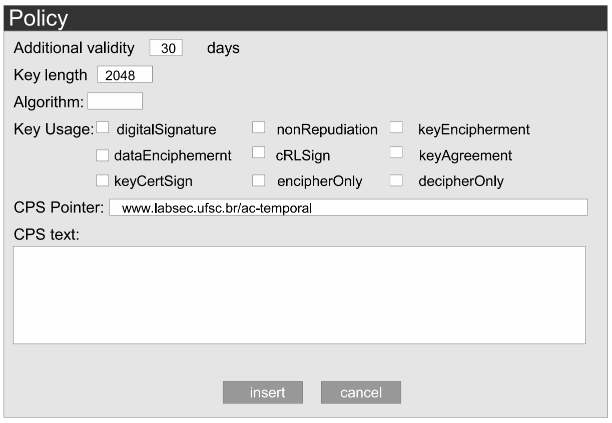
Request a new temporal digital certificate

Server Current Date and Time: 11/16/2005 10:21:31 am

Common Name	Issue Date	Release Date	Temporal Digital Certificate	Private Key	Report
Adriana Elissa Notoya	11/10/2005 09:42:07am	11/10/2005 10:00:00am	Download	Download	View
Adriana Elissa Notoya	11/10/2005 10:12:44am	11/10/2005 11:00:00am	Download	Download	View
Adriana Elissa Notoya	11/12/2005 05:10:55pm	11/14/2005 08:45:00am	Download	Download	View
Adriana Elissa Notoya	11/15/2005 02:30:22am	11/20/2005 08:00:00am	Download	Unavailable	–
Adriana Elissa Notoya	11/15/2005 02:40:09am	11/20/2005 09:30:00am	Donwload	Unavailable	–

Figure 13: Client TDC Management Interface.

The temporal certificate issuing policy, defining the on-line TCA operation, is configured by the system administrator through the use of the interface presented in Figure 14. All the functions performed by the system are constrained by the parameters established in the policy.


Policy

Additional validity 30 days

Key length 2048

Algorithm:

Key Usage: ☐ digitalSignature ☐ nonRepudiation ☐ keyEncipherment ☐ dataEnciphemernt ☐ cRLSign ☐ keyAgreement ☐ keyCertSign ☐ encipherOnly ☐ decipherOnly

CPS Pointer: www.labsec.ufsc.br/ac-temporal

CPS text:

Figure 14: TCA Administrator Interface

The TDC issued by the on-line TCA can be used in any application that supports X.509v3 certificates, examples include e-mail clients and Internet browsers. We performed several tests with different types of clients and we found that documents encrypted using the TDC were correctly received and were only disclosed at the time specified, when the TCA released the private key and this private key was imported by the application. The encryption and decryption operations were performed by the applications.

As soon as the private key temporal certificate was released by the on-line TCA, the certificate was installed in order to proceed with the decryption process.

We have proposed to use the system developed in public buying processes. In this application the proposals must present information related to the buying process, using a form which lists all the products being bought, with their prices. The public key TDC is requested by the on-line TCA and published by the person in charge of the buying process.

Each supplier that wants to take part in the buying process fills out the form with the appropriate data and sends it to the Encryption Module, along with the published public key TDC. After being encrypted, the proposal is sent to the buying commission, which will at the correct time, request from the on-line TCA the related private key temporal certificate required for proposal disclosure.

Figure 15 presents the relation of this system with the EM and on-line TCA components.

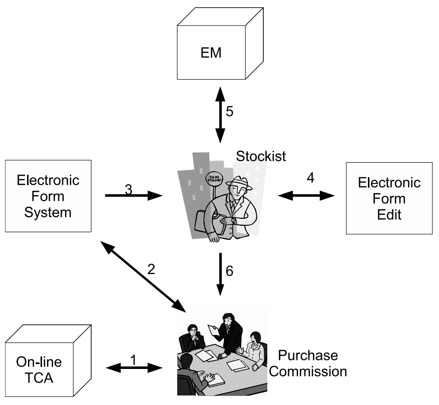


Figure 15: Bid System.

6 TKRI Analysis

The Temporal Key Release Infrastructure was developed according to the security requirements presented in section 2.1. In this section, we discuss the achievement of all the security requirements.

The private key, necessary for the electronic document decryption, is controlled by a TTP, called the TCA, that only releases it after a specified time and after the necessary authentication has been performed. In this way, the TKRI fulfills the security requirements *r1-a*, *r1-b*, and *r1-c*. The *r1-d* is fulfilled through the public window mechanism.

Using the private key destruction mechanism, it is possible for the MG of a specific private key to request the TCA to destroy this key if this possibility has been specified in the TDC request. After the private key is destroyed, the documents can be considered unavailable since they will remain secret until the cryptographic technique has been broken, which could occur at some future time. This mechanism allows the achievement of security requirement *r4*.

Access is controlled by the TCA and all the transactions performed by the entities are registered. Since it is possible to determine which entities have accessed, the temporal private key and when it was accessed, the security requirement *r5* is achieved.

Based on the TCA operating mechanism and policies, it is possible to determine that the electronic document content protected by a specific private key will be disclosed only after this private key has been released. In this way, the entities that have access to the encrypted document and to the private key used to decrypt the document cannot deny knowledge of the electronic document. If the TCA keeps the private key secret, the electronic document will remain confidential, fulfilling the security requirements *r2* and *r3*.

The encryption module usage does not allow the establishment of relationship between the plain text document and the encrypted document. This approach allows the fulfillment of the security requirement *r1*.

The plain text and encrypted documents are digitally signed. It is also possible to include plain text information that will remain readable to interested users. In this approach, it is possible to prove electronic document integrity and authenticity, fulfilling security requirement *r6*.

The use of a TSA, in conjunction with stamps and audit trail registration allows the auditing process to be performed in the infrastructure and in its services, fulfilling *r7*.

This analysis shows that the proposed Temporal Key Release Infrastructure fulfills all the security requirements specified in section 2.1.

7 Conclusions

Applications that require temporal confidentiality were studied in order to specify the relevant security requirements. Solutions found in the literature did not fulfill all the requirements listed. A Temporal Key Release Infrastructure was then proposed, and shown (a) fulfill all the security requirements specified in the initial study, and (b) the performance requirements of the applications. The functionality of the prototype developed was described in detail.

We believe that, using this approach to the temporal confidentiality problem, the use of electronic documents in applications such as public buying processes, wills, and confidential data storage is secure and viable. The authors agree that the policy model is flexible to satisfy any applications requirements without changes.

The TKRI was developed using known asymmetric cryptosystems such as RSA and standards like X.509v3 and PKCS. However, the inherent characteristics of the TKRI allow that these technologies could be replaced, if necessary, by new technologies like eliptic curve cryptosystems and one-time pad ciphers. These characteristics, of flexibility and adaptability, will be explored in a future work.

Another questions of our proposal need to be considered in the future: a) how ensure that an entity with a given public key really is a trustworthy TCA, and what if its key gets compromised? b) what performance loads and interoperability issues emerged in the implementation?

The authors would like to thanks the anonymous readers for their detailed comments and help us to improve considerably our paper.

Acknowledgments

We would like to thank Poorvi Vora and her students at George Washington University, Brazilian National Education and Research Network and Federal University of Santa Catarina, Brazil.

References

[1] C. Adams and S. Farrel. Internet x.509 public key infrastructure certificate management protocols. Request for comments: 2510, Network Working Group, 1999.

[2] Boris Balacheff, Liqun Chen, David Plaquin, and Graeme Proudler. A trusted process to digitally sign a document. In *Proceedings of the 2001 workshop on New security paradigms*, pages 79–86. ACM Press, 2001.

[3] Dejane Luiza Bortoli. O documento eletrônico no ofício de registro civil de pessoas naturais. Master's thesis, Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina, Florianópolis, Brazil, 2002.

[4] Ahto Buldas and Helger Lipmaa. Digital signatures, timestamping and corresponding infrastructure. Technical report, Küberneetika AS, 1998.

[5] Aldar C. F. Chan and Ian F. Blake. Scalable, server-passive, user-anonymous timed release cryptography. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 504–513, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Vanessa Costa. Um estudo da confiabilidade do processo de protocolação digital de documentos eletrônicos. Master's thesis, Universidade Federal de Santa Catarina, 2003.

[7] Júlio da Silva Dias, Denise Bendo Demétrio, Ricardo Felipe Custódio, and Carlos Roberto De Rolt. Reliable clock synchronization for electronic documents. In *Proceedings of III IEEE Latin American Network Mangment Systems*, pages 550–559, 2003.

[8] T. Dierks and C. Allen. The tls protocol version 1.0. Rfc 2246, Network Working Group, , United States, Jan 1999.

[9] Rabin M. O. Rabin T. Gennaro, R. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. *Proceedings of the 1998 ACM Symposium on Principles of Distributed Computing.*, 1998.

[10] Rosario Gennaro. *Theory and Practice of Verifiable Secret Sharing.* PhD thesis, Massachusetts Institute of Technology, May 1996.

[11] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. In *CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 437–455, London, UK, 1991. Springer-Verlag.

[12] Wenbo Mao. Timed-release cryptography. In *SAC '01: Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 342–358, London, UK, 2001. Springer-Verlag.

[13] Timothy C. May. Timed-release crypto. <http://cypherpunks.venona.com>, 1993.

[14] Adrian McCullagh, Peter Little, and William Caelli. Electronic signatures - understand the past to develop the future. *University of NSW Law Journal*, 21(2), 1998.

[15] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[16] David L. Mills. Improved algorithms for synchronizing computer network clocks. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 317–327, New York, NY, USA, 1994. ACM Press.

[17] Marco Casassa Mont, Keith Harrison, and Martin Sadler. The hp time vault service: exploiting ibe for timed release of confidential information. In *In Proceedings of the twelfth international conference on World Wide Web*, pages 160–169. ACM, 2003.

[18] NIST. Fips pub 140-2 security requirements for cryptographic modules. <http://csrc.nist.gov/cryptval/140-2.htm>, January 2002.

[19] Everton S. Pasqual, Júlio Da Silva Dias, and Ricardo Felipe Custódio. A new method for digital time-stamping of electronic document. In FIRST, editor, *Proceedings of the FIRST 14th Annual Computer Security*, 212 West Washington, Suite 1804 Chicago, IL 60606, 2002. Phoebe J. Boelter Conference and Publication Services, Ltd.

[20] Fernando Carlos Pereira. Criptografia temporal: Aplicação em processo de compras. Master's thesis, Universidade Federal de Santa Catarina, Florianópolis, Brazil, 2003.

[21] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, Cambridge, MA, USA, 1996.

[22] Ronald L. Rivest. Description of the lcs35 time capsule crypto-puzzle. <http://www.lcs.mit.edu/news/crypto.html>, 1999.

[23] RSA. <http://www.rsa.com/pkcs>, 2004.

[24] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition.* John Wiley & Sons, Inc., New York, NY, USA, 2 edition, 1995.

[25] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, Nov 1979.

[26] Douglas R. Stinson. *Cryptography : Theory and Practice*. CRC Press, 1995.

A Notation and Symbols

The notation and symbols to be used throughout this paper is summarized as follows:

- DOC - Electronic document.
- DC_{U_i} - User i's digital certificate.
- $E_K(x)$ - the ciphertext of data x, encrypted with key K.
- EM - Encryption Module.
- $H(x)$ - one-way hash function.
- K_S - Session key.
- K_{R_i} - i's Private key.
- K_{U_i} - i's Public key.
- L_i - a period of time.
- MG - a Manager (can configure a TCA or an EM, create keys and state policies but cannot operate a key).
- OP - an Operator (can use a cryptographic key).
- P - a Policy that state how a system or a secret sharing scheme work.
- RM - the Request Manager (responsible for requesting temporal digital certificates with a specific policy).
- RNG - Random Number Generator.
- RO - the Request Originator (uses a TDC and an EM to encrypt electronic documents).
- RP - the Recipient (receives an encrypted document and will decrypt it when he obtains the decryption key).
- S_i - a share i in a secret share scheme, SS.
- $S_{K_i}(x)$ - the digital signature of data x, by signer i.
- SS - Secret Sharing scheme.
- TCA - Temporal Certificate Authority.
- TDC - Temporal Digital Certificate.
- TDCIP - TDC Issuing Policy.
- TDCDP - TDC Declaration Practice.
- TKRI - Temporal Key Release Infrastructure.
- TSA - Time Stamp Authority.
- TTP - Trusted Third Party.

Limited Delegation for Client-Side SSL*

Nicholas Santos
nicholas.j.santos@gmail.com

Sean W. Smith
sws@cs.dartmouth.edu

Department of Computer Science
Dartmouth College

Abstract

Delegation is the process wherein an entity Alice designates an entity Bob to speak on her behalf. In password-based security systems, delegation is easy: Alice gives Bob her password. In the real world, end-users find this feature rather useful. However, security officers find it infuriating: by sharing her password, Alice gives *all* of her privileges to Bob, who then becomes indistinguishable from her. As enterprises move to PKI for client authentication, such secret sharing becomes impractical. Although security officers appreciate this, end-users may likely be frustrated, because this more secure approach to authentication and authorization prevents their ad hoc but reasonable delegation. In this paper, we present a solution that satisfies users as well as security officers: using X.509 proxy certificates (in a non-standard way) so that user Alice can delegate a subset of her privileges to user Bob in a secure, decentralized way, for Web-based applications. We validate this design with an SSL-based prototype: an extension for the Mozilla Firefox Web browser and a module for the Apache Web server that allow them to handle multiple chains of these certificates.

1 Introduction

In real-world situations, users often want to temporarily delegate some of their privileges to others, for reasons that are often rather legitimate. Most legacy computer systems implicitly tie a set of privileges to a password and thus make delegation surprisingly easy. If a user wants to use her computer—or read her e-mail, or sign onto her favorite chat account—she types in her password. If she wants to let her friend check her e-mail for her, she gives him her password. Like it or not, users are accustomed to this paradigm.

For many reasons, security experts promote PKI as a replacement for passwords. Our own university has rolled out an X.509 identity PKI to over 75% of the user population, and has migrated its Web-based applications from legacy passwords to client-side SSL for user identification and authentication. However, we fear that if PKI does not offer a way for users to delegate permissions for the scenarios they feel are reasonable, users will again force their own form of delegation into the system. PKI advocates may shudder to imagine one user lending another her *private key*, but unless there's an easy-to-understand way to delegate rights, that might be her only option.[1] Hence, in order to be usable in many real-world enterprises, client-side PKI authentication needs *a secure, generalizable way to allow for delegation.* This delegation mechanism should be *decentralized,* to avoid the cost and hassle of enterprise-wide or even application-specific databases that need to keep track of every user and every privilege. (That would negate many of the reasons for PKI in the first place.)

In this paper, we develop a system—Web-based delegated authentication via proxy certificates—that empowers Alice to unambiguously specify a limited subset of her privileges to pass to Bob, so that he can take care of business on her behalf. We equip Web browsers with the ability to issue proxy certificates carrying security policies, and the ability to pass proxy certificates to a Web server via client-side SSL. We equip Web servers with the ability to pass the credentials encoded in these proxy certificates to server-side scripts, which can then make their own security decisions.

Pushing the delegation process below the application layer makes this solution generalizable. If Alice wants to delegate privileges to Bob, she does not have to visit each one of her Web applications and explicitly delegate to Bob. She can issue one proxy certificate encoded with the policies for all these applications. Also, developers can build secure Web applica-

*This work was supported in part by the NSF, under grant CNS-0448499. The views and conclusions do not necessarily represent those of the sponsors. A preliminary version of this work appeared as the technical report [13]. The first author is now affiliated with Google.

[1]Indeed, we've already seen this happen.

tions on top of this Web server, and take advantage of delegated authentication without implementing it themselves.

This Paper. Section 2 discusses the high-level goals of our system. Section 3 discusses the PKI framework we used. Section 4 discusses our design. Section 5 discusses our prototype. Section 6 discusses related work. Section 7 concludes with some directions for future work.

2 Goal

We're considering an enterprise with a standard X.509 identity PKI for its users. We consider two classes of entities: end users (like Alice and Bob), and service providers (Web sites that Alice visits). The service providers follow the PKI gospel and use client-side SSL to identify and authenticate users.

Alice has privileges on these Web sites, and may wish to delegate some of these privileges to Bob. To support this action, we need three things. For the PKI, we need a format for a *delegation certificate*, a digitally signed statement from Alice giving Bob some rights. For the end users, we need a Web browser plug-in to issue and manage delegation certificates. Finally, for the service providers, we need a server module to verify delegation certificates during client-side SSL, and interpret the delegation appropriately.

The Web browser plug-in is easily distributed. Most modern browsers have a system for installing such a plug-in automatically by clicking a link, and users are accustomed to installing such add-ons. Mozilla Firefox, for example, has "extensions," and a similar plug-in could be written for Internet Explorer.

The module for the service provider will be more cumbersome to install, and will vary depending on the particular Web server software. Apache servers provide support for configurable modules that are dynamically loaded on start-up. The SSL-handling code is one such Apache module. So a service provider with Apache would have to replace the SSL-handling module with one equipped to handle delegation certificates.

We imagine a typical end user scenario might work as follows. Alice asks Bob to check her Web-based e-mail account while she's out of the country. He agrees. Bob e-mails Alice his *public key certificate*. Alice inspects this certificate, then uses it as the basis for a new certificate for Bob: a delegation certificate signed by Alice's secret key. This new certificate contains Bob's name and public key, and explicitly authorizes Bob to log into Alice's e-mail account and read e-mail. It contains no statement that authorizes him to send e-mail, nor to log into the university record system and view her grades. Alice e-mails this certificate to Bob, along with the certificate chain attesting to her own public key certificate. Bob installs this certificate in his Web browser. When he logs in to the Web-based e-mail account via an SSL session, he presents the delegation certificate issued by Alice. The Web server logs the fact that Bob logged in with Alice's identity. The server's environment variables indicate to the Web application that Bob has permission to read but not send Alice's e-mail. If it is a well-designed application, it will check these permissions and act accordingly.

Rejected Options. In our protocol, Alice issues the certificate herself. She then transmits her certificate to Bob on her own, or via a disinterested third party like a public certificate database. But this isn't the only way to solve the same problem. The delegation of privileges could also be handled through the enterprise's CA or through the service provider. For example, the CA could provide a Web-based service; Alice submits authenticated delegation requests, and the CA then issues the certificate. Or, alternatively, Alice could log into the service provider's Web application, and tell that application explicitly that Bob has permission to speak on her behalf. This second method bypasses certificates completely.

These other approaches have disadvantages. First, they both put a burden on a central server. Decentralization is a boon to all parties—the process is less complicated for Alice, and it relieves the server of the responsibility. Consider banks that charge ridiculously large "service fees" for printing an on-line bank statements, in the hopes that customers will just print the bank statement out at home. They want users to take care of business on their own. Second, they may have privacy problems. Suppose that Alice delegates access to Bob for emergencies—she may not want anyone to know about this delegation until he steps forward. Direct correspondence between Alice and Bob allows them to keep this arrangement (relatively) secret. Lastly, these approaches may have scalability problems. For example, it would be inefficient for each service provider to keep its own list of the parties to whom Alice has delegated privileges. In the approach where the CA issued the delegation certificate, the CA delegation service would have to change to accommodate every new service provider and every new privilege that service might provide. But if Alice and Bob issue their certificates to each other directly, then the scalability issues aren't so bad—Alice only has to keep track of which delegation-enabled service-providers she has visited.

Orthogonal Issues. Before we move on, we should clarify what problems we're *not* trying to solve.

Once we give Alice the ability to delegate her privileges to Bob, Bob may want the ability to act on behalf of two people at once. He may want to read both his mail and Alice's mail at the same time—in other words, he may want to assert multiple identities. There are some tricky semantics involved in how applications should deal with a user with multiple delegated identities. We recognize that these semantics are difficult. And different applications will have different ways of handling such users. But the scope of this project does not extend beyond the lowest level of multiple-identity authentication. We will, however, provide a framework for application developers to deal with multiple identities.

Additionally, the goal of this project is not to explore how we can specify delegation in policy statements. There are many standardized languages, such as XACML, that allow security professionals to precisely specify authentication and access control rules. They are a useful tool for security administrators. But we assume that most users will not care for such a fine level of access control when they are determining which privileges to delegate in a proxy certificate. We need a simple mechanism for *users* to describe this delegation. This simplicity should be reflected in the server-side directives as well. A set of rudimentary directives—based loosely on the directives defined for access control in Apache—will be enough to demonstrate the possibilities of delegation. For the purposes of this project, that's what we care for.

3 Delegation Certificates

As Section 2 described, we need a format for "delegation certificates." We chose X.509 proxy certificates.

SDSI-SPKI is attractive because it provides a much more straightforward and simple syntax for the delegation of credentials [3]. However, it's an X.509 world, and that's what the standard infrastructure supports. Prior experience in our lab (e.g., [4]) suggested that swimming against the current is not productive.

The ruling certificate standard, X.509, is rigidly hierarchical and does not allow the average user to issue certificates. In X.509, there are certificate authorities (CAs), and there are end entities (normal users like Alice). CAs can issue certificates, but only to entities that are "subordinate" to the CA. End entities do not have the authority to issue any certificates—the reason that they are called "end entities" is because their certificates can only appear at the end of a certificate chain [6]. To delegate her privileges to Bob in the X.509 system, Alice would need to find a CA that she and Bob had in common, and ask this CA to sign her privileges over to Bob. Such a common trusted CA might not even exist. And even if Alice does find a common CA, delegation may be difficult. CAs are the bureaucrats of the X.509 world—it can be cumbersome (and often financially expensive) to get their approval

The Globus Toolkit (http://www.globus.org/) ran into this problem while building a secure framework for distributed computing. But part of its goal was to share "securely," and "without sacrificing local autonomy." A process sitting on a remote, autonomous machine may need access to restricted resources, so it needs a mechanism to authorize this access dynamically. The CA approval process was unsatisfactory, for the exact reasons noted above. CAs were too cumbersome to be practical for authorizing short-lived processes [15]. Thus, the Globus Toolkit developers invented *proxy certificates* for delegation. This is probably the most widespread use of PKI-based delegation in real-world applications today. After some evolution, proxy certificates were standardized for X.509 in RFC 3820.

Proxy certificates are an extension to the X.509 certificate standard that allow end entities to sign certificate statements that delegate their own privileges to other entities. By the standard, an end entity generates temporary private and public keys, signs a short-lived proxy certificate that passes on some of her privileges to the temporary keypair, then gives those credentials to a third party entity. The identity of the proxy certificate is derived from the identity of the end entity. Because a proxy certificate can also testify to another proxy certificate, the identity of a chain of proxy certificates is the last non-proxy certificate in the chain (the end entity certificate).

Notice that when we say that these credentials are "temporary," this is merely a convention. There is no rigorous definition for the length of a "temporary" period of time [14]. This flexibility is intentional, because simplicity is of the essence. On the Grid, these proxy certificates can be issued to dynamically created processes without requiring the approval of a CA.

The X.509 proxy certificate offers numerous advantages for our scheme. Because it contains so much auxiliary information, the server can keep comprehensive server logs on who Bob is and which identities he's assuming. It's also explicitly intended for delegation, as opposed to X.509 attribute certificates, which can handle more general attributes. But most importantly, current tools actually contain support for X.509 proxy certificates. The OpenSSL libraries can issue and verify them [8]. The same support is not behind X.509 attribute certificates. And it certainly cannot be said for SDSI/SPKI, for which there is little support in major applications, with the exception of some closed environments.

X.509 proxy certificates piggy-back off standard X.509 certificates. The major technical differences are that proxy certificates can be signed by end entities, and a proxy certificate must define a critical `ProxyCertInfo` extension [14].

4 Our Design

To achieve the vision of Section 2, we need several things. Alice needs a way to issue proxy certificates. The Web application needs a way to tell Alice what permissions she can delegate, so that Alice can select which of these permissions to encode in her proxy certificate. Bob's Web browser needs to be able to send multiple chains of proxy certificates in an SSL session. Bob must be able to choose which identities he would like to assert. (In this example, he has a choice between his own identity "Bob" and his delegated identity "Alice.") The Web server must understand proxy certificates, and be equipped to deal with multiple chains of them.

This section explains the design of these tools for a suite of Web applications and Web standards: X.509 proxy certificates (Section 4.1), Mozilla Firefox (Section 4.2), SSL/TLS (Section 4.3), and the Apache Web server (Section 4.4).

4.1 Non-standard Proxy Certificates

For our project, we decided to depart from the standard that a proxy certificate must testify to the public key of a *temporary* keypair generated exclusively for that certificate [14]. Instead, in our system, proxy certificates will testify to an existing public key, and for which previous certificates—an identity certificate, and perhaps other proxy certificates—exist.

This departure from the standard gave us several advantages. It does not require Alice to send a new temporary private key to Bob. In fact, no secret information is exchanged between them. Only their public key certificates are transmitted. As long as Alice can verify Bob's certificate, this will be secure. Secondly, in our application scenarios, having lots of temporary keypairs will not be appealing for users. In a human-usable delegation system, simplicity should be a major goal, and a single keypair for each keystore is much more simple. Lastly, notice that we can repeat the delegation process with other users each delegating their own privileges to Bob's public key. This allows Bob to obtain a grab bag of certificates, all with the same name and public key, but corresponding to different delegated identities. This could be useful in scenarios when Bob needs to represent more than one party in a service request authenticated via client-side SSL—which, by design, allows Bob to prove knowledge of only one private key. (The first author actually has done this in a process that has not yet made it to the Web: Dartmouth's on-campus housing auction. Two friends wished to share a room with each other. Neither could make it to the event, so they both delegated to the author, who then made a selection representing both of them.)

Revocation. Because proxy certificates are usually short-lived, researchers often wave their hands at the problem of revocation, as the potential for damage is reduced greatly by the certificate's early expiration date. In some projects, delegation is used to make the revocation process obsolete—the proxy certificates expire more quickly than a certificate revocation list (CRL) could be issued. For this project, we take this approach.

Privilege Attributes. In order to give a user the ability to pick and choose which applications the delegate can use on their behalf, we allow them to define attributes in terms of a *service* (the URL of a service provider) and an *ability* (an arbitrary string expression).

This will become clearer with an example. Suppose Alice wants to delegate to Bob the ability to read her mail from her Web-based `www.mail.gov` account, and to edit and post on her blog `www.aliceblog.com.` So she gives him the attributes "`www.mail.gov`: read" and "`www.aliceblog.com`: edit post." In other words, the attributes will consist of a list of permissions, and each list will be tied to a service provider. Proxy certificates have a space allotted to specify such a list of permissions as a `policy OCTET-STRING` in the `ProxyCertInfo` extension [14].

This list of attributes constitutes a list of privileges granted to Bob. Alice must explicitly name each service provider that Bob can interact with on her behalf. This allows each service provider to define its own set of privileges at any granularity. We tie privileges to service providers to avoid the trouble that would arise if two service providers use the same privilege name. For example, this prevents Alice from confusing "edit" privileges on `mail.gov` with "edit" privileges on `aliceblog.com.` Because the privileges are tied to the URL of the service provider, they remain unique. In effect, we solve the name collision problem by leveraging somebody else's infrastructure that has *already* solved the problem.

At least, that's how we'll think about the situation. URL addresses are not unambiguous. First, some Web pages use server farms, with one address mapping to multiple servers. Secondly, an adversary can spoof a URL. But for our purposes, these nuances are orthogonal. When we use the URL in this context, we are not assuming a trusted relationship with the service provider at that address. The URL simply allows us to differentiate between different service providers and the privilege sets that they offer.

We have not yet answered the question: How do service providers notify Alice of their set of privilege names? We will do that in Section 4.4 below.

4.2 The Browser

The Mozilla Framework is an open source software development framework. The most famous (current) application to come out of it is the Firefox Web browser. The framework strives to be cross-platform, programming-language-independent, and locality-independent. The framework also has useful properties that make it easier to modify—most notably the availability of the source code.

The Framework. First, we quickly review Mozilla's high-level code architecture to help the reader to understand how we modified Firefox.

Mozilla's organizes its code via the *XPCOM (the Cross-Platform Component Object Model).* XPCOM is the system for organizing all of the software libraries underlying Mozilla. In XPCOM, a central component manager keeps track of a number of exclusive, encapsulated components that each implement a well-modularized set of functions. These components can be written in any language for which XPCOM language bindings are defined, including Python and Java—but are typically written in C++ or Javascript. The methods and attributes of an XPCOM component can only be accessed by defining a public interface through a second language called *XPIDL* (longwise, that's the *Cross-Platform Interface Description Language*). This interface then allows the component to be used as an object in any XPCOM-supported language. One can define an interface in XPIDL, then implement it with several different components (possibly in different programming languages), that satisfy the interface in different ways. For example, the `nsISocketProvider` interface is implemented by one component that handles SSL sockets, and another component that handles TLS sockets. See Chapter 8 of [1] for more information.

The components are managed by a component manager, which keeps a hash table with entries for each component. The entries of the hash table are indexed by human-readable URIs called contract IDs. Each entry also contains a universally unique identifier (UUID) as a sequence of integers, and the memory location of a constructor for this component. When one component wants to use another, it gives the component manager the URI, and the component manager gives it back an object.

This structure is relevant to our project. If we could overwrite an entry in the hash table, we could replace any native Firefox component with our own component. As long as the custom component implements all the XPIDL-defined functions, the rest of the Mozilla Framework will treat it exactly like the native component.

Extensions. The Mozilla Framework provides a simple mechanism for installing "extensions" from over a network. The modification for proxy certificates should be packaged as such an extension. After all, few users would be willing to download a custom browser with modified source code to use delegated authentication.

We create new XPCOM components that handle proxy certificates. We then develop a GUI for this application to interact with the local XPCOM components, and by extension, the proxy certificate library. In this way, our extension can be divided into three pieces.

First, we need an interface to allow Alice to *issue proxy certificates.* At first glance, there's no reason for this to be built into the browser—it could easily be a stand-alone application. The reason it's in the Web browser is not for Alice's benefit, but for the service provider's benefit. As we will soon see (Section 4.4), each service provider will propagate the set of privilege names that it defines by talking to this extension. Then the user interface can show Alice a list of delegation-enabled service providers she's visited, as well as the privileges she can delegate for them.

Secondly, we need a back-end database to *manage proxy certificates* issued to Bob. *Network Security Services (NSS)*, the cryptographic library underlying the Mozilla Framework, does not behave properly around proxy certificates. At best, it's schizophrenic. NSS will often accept them at first—but as soon as it realizes that the proxy certificates have been signed by an end entity, it may immediately trash them. We simply need a database that can handle proxy certificates properly, and will safely store them outside of NSS.

Finally, we need a way for Bob to use his proxy certificate(s) in *client-side SSL authentication*. This part of the extension will be responsible for getting the proxy certificates to the server during an SSL session. This is more difficult than it sounds, because this will require slight changes to the SSL protocol.

4.3 SSL/TLS

SSL/TLS is the ubiquitous protocol for secure communication on the Internet[2]. It consists of three essential pieces. In the "Hello" phase, the client and server initiate communication. In the "Handshake" phase, the server and client exchange information using a chosen asymmetric-key algorithm, with the goal of establishing a session secret. This information may optionally include a certificate exchange, and the server and client may optionally verify each other's certificates before they agree to connect. Finally, in the "Application" phase, we can now send data across the network encrypted and MAC'd with the session secret via our favorite symmetric-key algorithm [2]. To incorporate delegation into this protocol, we only need change a narrow segment of the client behavior during the Handshake phase. We can leave the rest of the protocol alone.

The Handshake phase changes because SSL/TLS expects the client to transmit no more than one chain of certificates. In this chain, each certificate testifies to the public key of the keypair that signed the certificate that came before it [2]. But for delegation, the client might need to transmit several certificate chains, with one chain corresponding to each delegated identity. To make this work, we have the client transmit the certificate chains for each delegated identity in serial, and assert that the first proxy certificate in each chain must testify to the same public key. Figures 1 and 2 demonstrate the change in the protocol when we add multiple certificate chains[3]. We do not permit Bob to use two different keypairs in the same session.

[2]SSL/TLS is a suite of several different standardized protocols. All these protocols are just variations on the same high-level ideas, though. For our purposes, they're interchangeable.

The "one public key" rule gives us an easy way to distinguish between certificate chains—when the validator sees a certificate in the chain that contains the same public key as the first certificate, this is the bottom of a new chain. As an additional bonus, this ensures that legacy certificate-validation code (applications that don't know about multiple-identity delegation) will reject any user that tries to assert multiple delegated identities. A certificate chain, after all, should not have a cycle.

From a theory standpoint, the idea that "public key" is a unique identifier of the user is also a cleaner way to think about PKI. The SDSI/SPKI certificate model makes this observation elegantly. The security of public-key crypto-systems implicitly depends on the assumption that public keys are unique. If two users had the same public key, then their cryptographic operations would be indistinguishable [3].

4.4 The Web Server

For the Web server, we focused on Apache, which is both open-source and market-dominant. For Apache, we only have to modify mod_ssl, which can hook into the Apache server from a dynamically loaded library. We can easily distribute this library to server administrators to enable delegation.

Code Additions. We need to modify the certificate validation code to accept proxy certificates and to be able to recognize when the client is sending multiple chains of proxy certificates. Recognizing the proxy certificates is simple—that functionality comes standard with OpenSSL, the cryptographic library underlying Apache. The validation of multiple chains

[3] TLS protocol extensions (RFC 4366) could make this change more graceful by allowing the client to explicitly ask the server to accept multiple certificate chains. These standards are recent, and were not available at the implementation phase of this project.

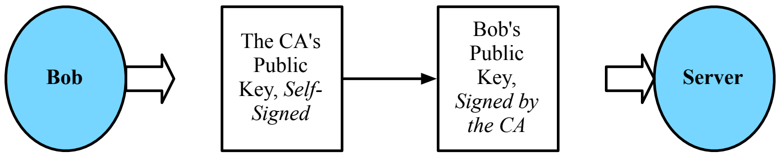

Bob
The CA's Public Key, Self-Signed
Bob's Public Key, Signed by the CA
Server

Figure 1: Passing client certificates to the server by the TLS 1.0 standard. Notice that Bob's public key certificate is sent first, while the CA certificate that testifies to it comes afterwards. (The self-signed CA certificate is optional. We include it here to enhance the illustration.)

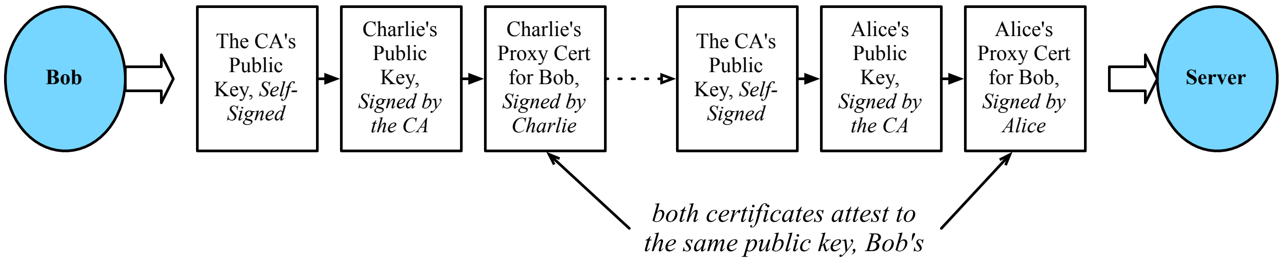

Bob
The CA's Public Key, Self-Signed
Charlie's Public Key, Signed by the CA
Charlie's Proxy Cert for Bob, Signed by Charlie
The CA's Public Key, Self-Signed
Alice's Public Key, Signed by the CA
Alice's Proxy Cert for Bob, Signed by Alice
Server
both certificates attest to the same public key, Bob's

Figure 2: Passing multiple certificate chains to the server. As before, each certificate in the same chain testifies to the one sent before it. The dashed arrow represents the point where a traditional TLS server would register an error, because Bob did not sign the CA's public key certificate.

of certificates is not so easy to implement, because it requires changes to both Apache and OpenSSL. It has also some tricky semantics. What happens if the client sends two certificate chains, and only one of them is valid? On one hand, the SSL protocols imply that if the server sees an invalid client certificate, it should notify the client by sending an error message, and should cut short the SSL session. But it seems more natural for the server to accept the valid chain, and quietly fail to grant the privileges specified in the invalid chain. In this implementation, the tie goes to the specification—if one certificate chain is invalid, the whole authentication should fail. This will make it clearer from a user interface perspective that something has gone wrong.

Access Control Directives. We also define some additional directives in the Apache configuration files. These directives will tell Apache *how* to respond to proxy certificates. A legacy server will only accept a single certificate chain. So the modified server will, by default, only accept a single identity. It will consider multiple identities only when it sees a special directive in the configuration files.

These new directives are as follows.

The `SSLMultipleIdentities` directive tells the server to allow a user to assert multiple identities at once. The server will accept multiple certificate chains, one for each identity. And for each user, it will keep track of the list of privileges delegated by that user.[4] To ensure the same privilege is not counted twice, we ensure that if there are multiple certificate chains, no two chains derive authority from the same end entity.

The `SSLExclusiveIdentity` directive tells the server to accept only the first identity. In truth, it is the default case. But because these directives are interpreted on a per-directory basis, this directive is useful for overriding the `SSLMultipleIdentities` directive in a parent directory.

A server connection environment variable

```
_SERVER[ ''SSL_DELEGATED_IDENTITIES'' ]
```

will be set to `STRING`, an ASCII character string encoded as a Lisp s-expression. It will contain the common name of each certificate in an identity asserted by Bob. Obviously, this encoding doesn't work if common names have parentheses. So if the server sees a common name with a parenthesis, it simply refuses to grant this identity. Application-level scripts can read this environment variable, and thus find out easily which identities Bob has.

We will also have directives for interpreting the `policy` field of the `ProxyCertInfo` extension.

[4]This is the simple-minded way to enumerate sets of privileges delegated by a set of users. There are more specialized ways to model multiple simultaneous identities. We've explored this a bit elsewhere [13].

The `SSLRequirePrivilege` directive takes a *name* and a *Description*. The *name* must be an alphabetic character string (with no white space), and must not conflict with any other privilege names. It will specifying the name of a privilege as it appears in the proxy certificate policy. If this directive is used, then Bob will be allowed access in the current directory subtree iff he has this privilege. If `SSLMultipleIdentities` is given as well, Bob will need to have this privilege for every identity that he tries to assert, or he will be rejected. The *Description* portion of the directive will be used for the propagation of the privilege set, which we will discuss shortly.

The `SSLRequestPrivilege` directive also takes a *name* and a *Description*. This directive is similar to the `SSLRequirePrivilege` directive. But in this case, Bob can access the directory whether or not he has the described privilege. The directive is used to specify privileges that are not used to restrict access at the server level, but are exposed via the environment variables anyway. (An application-level script might use this privilege as part of an XACML-based decision request.)

Notice that with these directives, every server can define a set of privileges. After the server verifies Bob's proxy certificate chain(s), it will check the `policy` field of each `ProxyCertInfo` extension in the chain, and grant Bob the privileges encoded in it. It will then set a connection environment variable `_SERVER[ ''SSL_DELEGATED_PRIVILEGES'' ]` to be `STRING`, a Lisp s-expression. Bob's privileges will be stored as a list of lists. Each sub-list will start with the identity name (the user delegating to Bob), followed by the privileges granted by that end entity.

Privileges in the Grand Scheme. To determine its set of privileges, the server parses all of its access files for the `SSLRequirePrivilege` and `SSLRequestPrivilege` directives described above, then builds a set of `( name , description )` ordered pairs of privileges. They are keyed by the name, so if the same name appears twice, one pair is rejected. All servers implicitly define the reserved pair `( all , ''All privileges'' )`. When Alice visits the server, the server gives her a cookie containing the privileges defined as Lisp s-expressions. The Firefox extension can then read these cookies, and add them to its list of (service provider, privilege) pairs. When Alice wants to sign a proxy certificate for Bob, the user interface will provide her with a list of all the servers that she's ever visited that support this form of delegation.

When Bob wishes to assert the privileges granted by Alice, he authenticates with this certificate in a client-side SSL session. He will pass the server the entire certificate chain, so that the server will see both Alice's end-entity certificate, and the proxy certificate she signed for Bob. The server then interprets the policy in the proxy certificate, and records in an environment variable that Alice granted Bob those privileges.

The reader should take note that this still leaves the Web application with a lot of responsibility to make authorization decisions. Our system simply records what Alice has granted in the proxy certificate policy—it makes no claim that Alice had the authority to grant Bob this privilege. An application can use the identity and privilege information to make authorization decisions; our directives are part of a server-level system that make it a easier to process and manage this information.

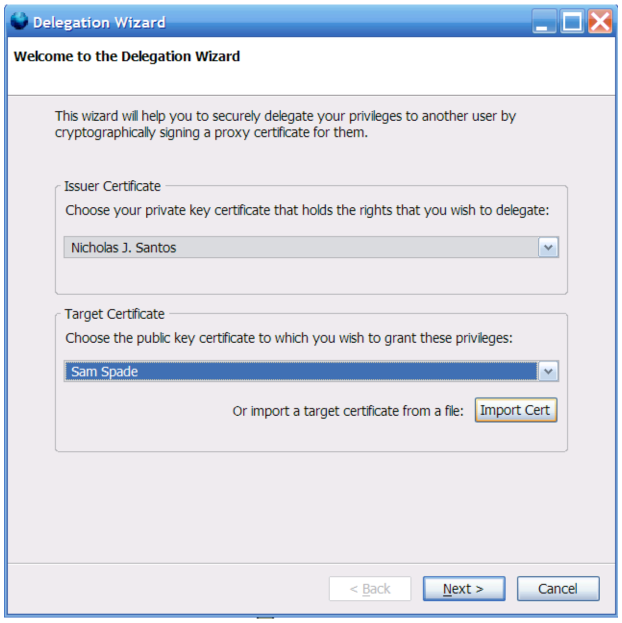
5 Our Prototype

To illustrate the functionality of our prototype, we'll consider an example. Nicholas Santos has hired Detective Sam Spade to represent him. He would like to sign a proxy certificate for Spade that will delegate all his permissions to Spade for a period of five days. (He's trying to get back a jeweled falcon, and he's asked Spade to negotiate on his behalf.) The next five figures illustrate the process he goes through to do this.

We will also discuss how to manipulate and issue proxy certificates with NSS, the Mozilla Framework's cryptography library, and to understand them with Apache. The point of this discussion is to examine the particular successes and pitfalls of our implementation, so that the reader has an idea of what it would take to reproduce such a system.

5.1 The Mozilla Extension

Making Room for Proxy Certificates. The `ProxyCertInfo` X.509 extension must be attached to all proxy certificates and marked critical [14]. By specification, cryptographic libraries must trash any certificate with unrecognized critical extensions [6]. So before we load any components that handle proxy certificates, the *Object Identifier (OID)* for the `ProxyCertInfo` extension must be dynamically registered with NSS. RFC 3820 additionally lists a number of OIDs for proxy policy languages that must be



Welcome to the Delegation Wizard

This wizard will help you to securely delegate your privileges to another user by cryptographically signing a proxy certificate for them.

Issuer Certificate

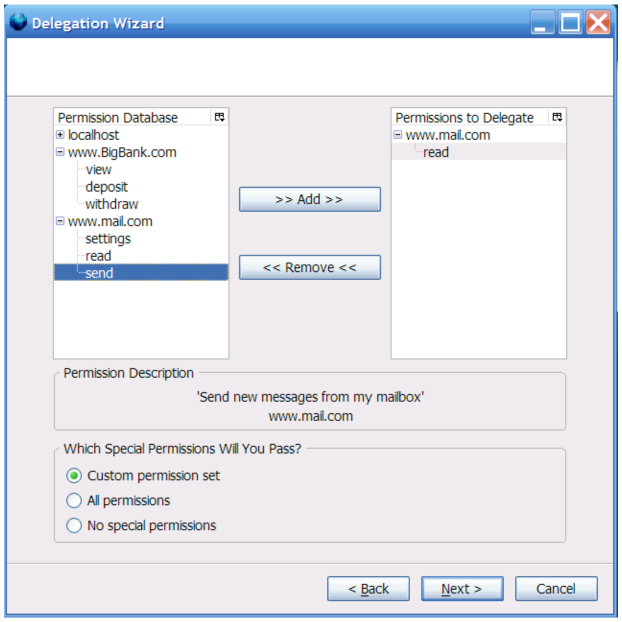
Choose your private key certificate that holds the rights that you wish to delegate: Nicholas J. Santos

Target Certificate

Choose the public key certificate to which you wish to grant these privileges: Sam Spade

Or import a target certificate from a file: Import Cert

Figure 3: Using our Delegation Wizard (part 1): Choosing a user certificate (any certificate for which we have the private key) and a target certificate to which its identity is delegated.



Permission Database
- localhost
- www.BigBank.com
  - view
  - deposit
  - withdraw
- www.mail.com
  - settings
  - read
  - send

Permissions to Delegate
- www.mail.com
  - read

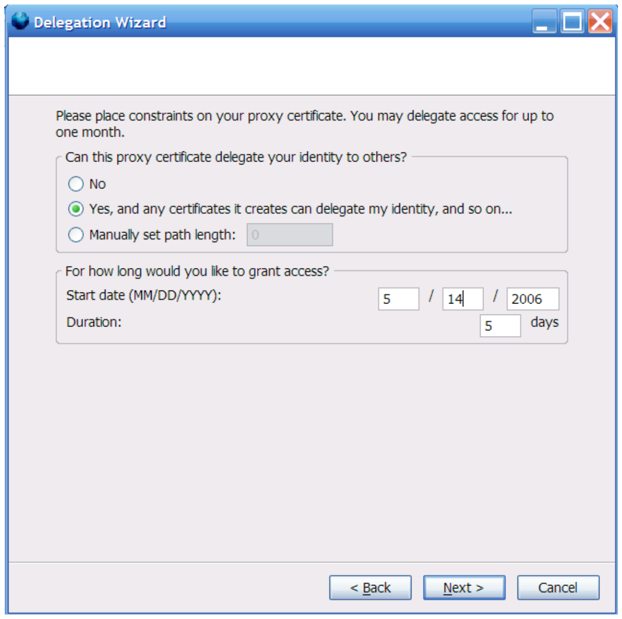
Permission Description

'Send new messages from my mailbox'
www.mail.com

Which Special Permissions Will You Pass?

[✓ Custom permission set] All permissions No special permissions

Figure 4: Using our Delegation Wizard (part 2): Choosing permissions to delegate. Each tree organizes the permissions by service provider. The tree on the left is a list of all available permissions; the tree on the right is a list of the permissions that will be delegated. The tree of available permissions is built by iterating through the cookies, and parsing them for the permissions.



Please place constraints on your proxy certificate. You may delegate access for up to one month.

Can this proxy certificate delegate your identity to others?

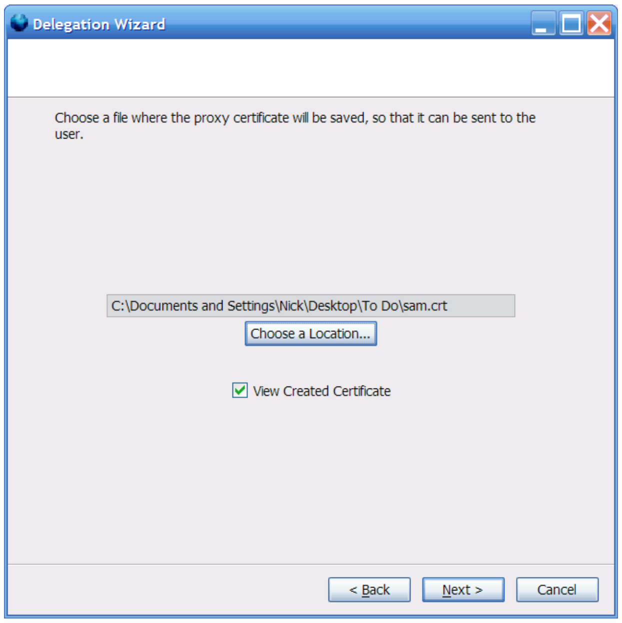
No [✓ Yes, and any certificates it creates can delegate my identity, and so on...] Manually set path length: 0

For how long would you like to grant access?

Start date (MM/DD/YYYY): 5 / 14 / 2006

Duration: 5 days

Figure 5: Using our Delegation Wizard (part 3): The constraints page, which allows setting a path length constraint and a validity period.

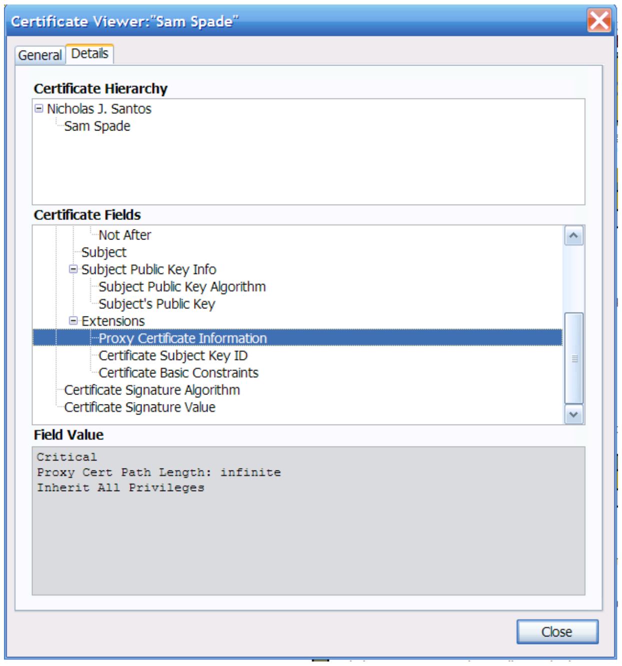


Choose a file where the proxy certificate will be saved, so that it can be sent to the user.

C:\Documents and Settings\Nick\Desktop\To Do\sam.crt

[✓ View Created Certificate]

Figure 6: Using our Delegation Wizard (part 4): Saving the certificate to a file, so it can be e-mailed to Detective Spade.


Certificate Viewer:"Sam Spade"
General
Details
Certificate Hierarchy
Nicholas J. Santos
Sam Spade
Certificate Fields
Not After
Subject
Subject Public Key Info
Subject Public Key Algorithm
Subject's Public Key
Extensions
Proxy Certificate Information
Certificate Subject Key ID
Certificate Basic Constraints
Certificate Signature Algorithm
Certificate Signature Value
Field Value
Close

```
Critical
Proxy Cert Path Length: infinite
Inherit All Privileges
```

Figure 7: Using our Delegation Wizard (part 5): The ASN.1 structure of the new proxy certificate.

understood by any proxy certificate implementation. Those OIDs must be registered as well.

The `ProxyCertInfo` extension contains three fields. The optional `pCPathLenConstraint` describes the depth of the cert chain below this one. The required `policyLanguage` is an OID for a policy language. The optional `policy` is, as noted earlier, the designated place for issuers to record policy info on what permissions they're delegating.

NSS allows developers to write templates—arrays of constants—that tell the ASN.1 encoder how to encode and decode types. A few wrapper functions and a `ProxyCertInfo` template are required to encode and decode that extension. The Mozilla Framework also has a separate ASN.1 handler that pretty prints ASN.1 sequences for GUIs. A few more functions on top of that object will make the `ProxyCertInfo` extension readable from a dialog box, as shown in Figure 7.

The reader should be aware that in order to handle proxy certificates adequately, our extension needed a lot more infrastructure than this. To handle the certificates internally, the extension needed access to raw data structures, including wrapper objects for the certificates, for their validity periods, and for some certificate-based GUI objects. But these needs were satisfied by simply copying the existing certificate-handling code, modifying it slightly for proxy certificates (and for publicly exported APIs), and compiling it into the extension. It's not academically interesting, and we will say no more of it.

Issuing Proxy Certificates. To issue a proxy certificate, we need an issuer and a target. The issuer testifies to the private key that will sign the new certificate. The target testifies to the name and public key that will be the subject of the new certificate (see Figure 3). This information is used to construct a certificate request internally. We call an NSS function to transform this certificate request into an unsigned certificate, at the same time adding an issuer and a validity period.

When creating any certificate—not just proxy certificates—there are standards and there are styles. The first is mandated in writing, the second is mandated by strong social pressure and convention. For standards, it's best to work straight from the source, RFCs 3280 and 3820. For style, we recommend Peter Gutmann's "X.509 Style Guide."[5]

For every proxy certificate we issue, we copy the name and public key directly from the target certificate. Per Gutmann's recommendation, we use the time in seconds since the UNIX epoch as the serial number, to ensure unique serial numbers[5].

We additionally attach several extensions. We have already covered the `ProxyCertInfo` extension. On the advice of OpenSSL's proxy certificate guide [8], we include a `BasicConstraints` extension that indicates that this certificate may *not* be a CA. We include a `SubjectKeyIdentifier` extension that contains a SHA-1 hash of the target certificate's DER-encoded public key data. Finally, we attach an `AuthorityKeyIdentifier` if and only if the issuer certificate has the `SubjectKeyIdentifier` extension. If it does, the key identifier from the issuer is simply copied into the key identifier field of this `AuthorityKeyIdentifier`.

The `SubjectKeyIdentifier` and `AuthorityKeyIdentifier` extensions are not really necessary, and may just be another example of redundancy in the X.509 standard. However, they do reinforce the idea that the public key identifier is a better indicator of identity than a X.500 distinguished name, because an identity is only as unique as its keypair. We mainly include

[5]A malicious user could certainly issue multiple certificates with the same serial number by changing the system clock, but this would do no actual damage. It would merely spite the standard.

this extension for ideological reasons. But we should mention that these extensions made the certificate validation code easier to debug, because the crypto library code that compared two key identifiers was usually much simpler than the crypto library code that compared X.500 names.

NSS did not encode the `AuthorityKeyIdentifier` correctly, so we copied the encoding function, fixed it, and compiled it into our library.

Once the extensions are added, the certificate is ready to be signed. A Mozilla XPCOM component provides functionality to log into any PKCS #11 interfaces (cryptographic tokens), asking the user for a password if we need one. Other publicly-exposed NSS functions allow us to get a handle on the private key, and use it to sign the data in a DER encoding.

To our knowledge, this is the first implementation of a proxy certificate issuer with NSS.

Storing Proxy Certificates. Once we have a DER encoding of the proxy certificate, we need a place to store it. NSS is no good, for numerous reasons. It is not aware that end-entities can sign certificates. Furthermore, this Firefox extension may be uninstalled, and if it is, it shouldn't be abandoning proxy certificates in the regular NSS database.

The key insight to storing proxy certificates is that the storage medium does not need to store any secrets (such as private keys). It can leave the private keys in the NSS secure storage, and only needs to remember where those private keys are located. The only disadvantage of this method is that the user could delete his private key from the NSS database, thus rendering his proxy certificates useless. Our implementation does not protect against this case; it just blames the user for the problem.

Because proxy certificates do not need to be protected for confidentiality, it would have sufficed to keep them in any non-volatile storage mechanism. In our extension, they are simply stored in an SQLite database[6]. The database key for each proxy certificate is derived from the serial number and the issuer name. Because issuers *should* issue certificates with unique serial numbers, this database key should be unique. Each entry in the database also contains a DER encoding of the certificate, as well as database keys to access the issuer certificate, "delegator" certificate, and private key in the regular database. The "delegator" certificate is the end-entity certificate in the chain ending in this proxy certificate. It names the end entity from which this proxy certificate derives its identity. (In chains with only one proxy certificate, the issuer is the delegator.)

[6]SQLite is a open source file-based database engine with a C API that accepts and executes queries in a subset of the SQL language. More information can be found at http://www.sqlite.org/.

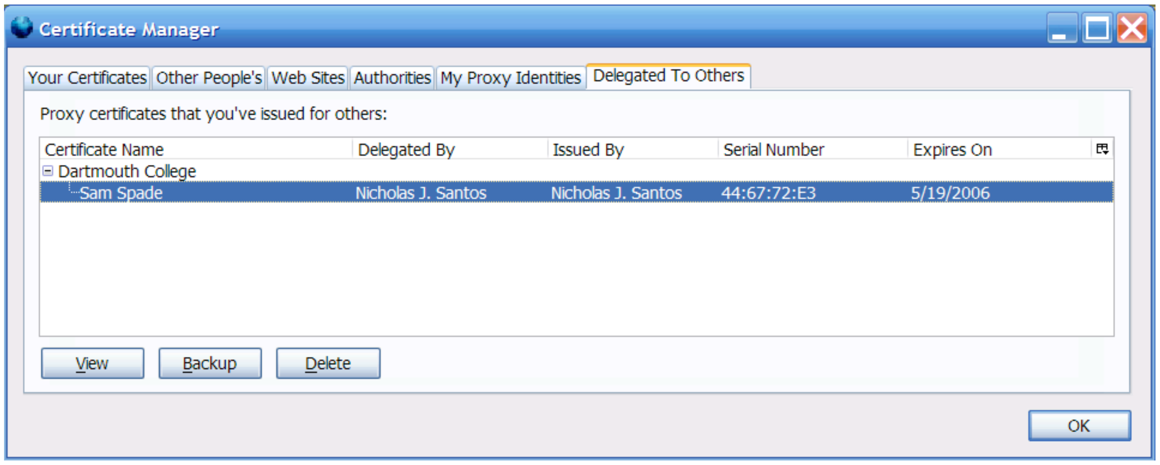
The database can be described in two sections: delegated certificates and the user's proxy certificates. The sections must not be assumed to be mutually exclusive, although they likely will be for most users. The portion for delegated certificates is merely a log. It tells Alice which proxy certificates she has issued, and allows her to re-export them if she needs to send them again (Figure 8). The other section of the database holds certificates delegated to the user, certificates for which the user has the private key (Figure 9). These are the identities that can be used in an SSL session.

Using Proxy Certificates, in Theory. The code to inject proxy certificates into an SSL session performs an interesting acrobatic stunt.

Recall from the original discussion of the Cross-Platform Component Object Model (XPCOM) that the Mozilla Framework keeps all its components in a hash table, hashed by a human-readable contract ID. If we ask the component registrar to load a component with the same contract ID, the registrar will, by default, simply overwrite that entry of the hash table. From reading the source code, this feature seems to be intentional, although it is not otherwise documented. But this also means that there is no documentation assuring us that this is a safe mechanism for extension development. Thus, we use it cautiously.

This feature allows us to play man-in-the-middle with Firefox's SSL/TLS handling code. First, the XPCOM objects that handle SSL and TLS sessions are registered at a second contract ID. Then, custom SSL/TLS handlers are registered at the first contract IDs, overwriting the entries there. These custom handlers intercept method calls intended for the original handlers, change the passed arguments, and then pass the altered arguments along to the traditional handlers by looking them up at the second contract ID. The same man-in-the-middle game can be played with return values. Thus, we can change the method arguments and return values at will to produce the desired effect. That's the theory—but it's not so simple in practice.

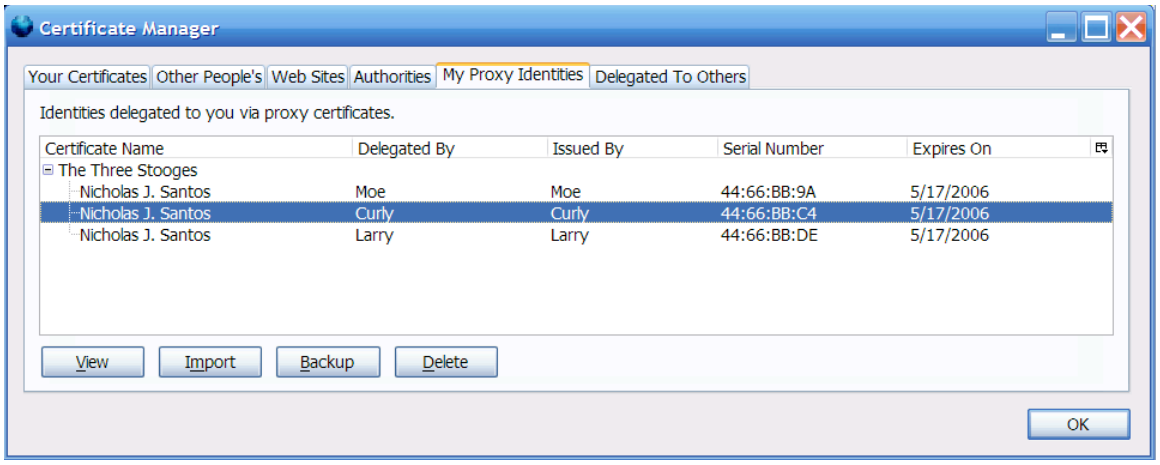
Using Proxy Certificates, in Practice. The actual implementation is far more complicated and involves several levels of indirection. The flow can be



Proxy certificates that you've issued for others:

Certificate Name	Delegated By	Issued By	Serial Number	Expires On
Dartmouth College				
Sam Spade	Nicholas J. Santos	Nicholas J. Santos	44:67:72:E3	5/19/2006

Figure 8: Viewing the certificates in the database that have been issued *by* this user.



Identities delegated to you via proxy certificates.

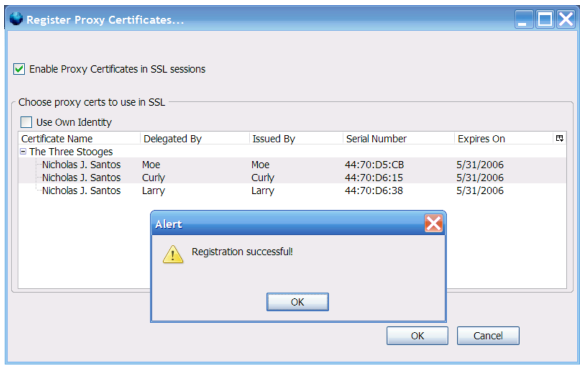
Certificate Name	Delegated By	Issued By	Serial Number	Expires On
The Three Stooges				
Nicholas J. Santos	Moe	Moe	44:66:BB:9A	5/17/2006
Nicholas J. Santos	Curly	Curly	44:66:BB:C4	5/17/2006
Nicholas J. Santos	Larry	Larry	44:66:BB:DE	5/17/2006

Figure 9: Viewing the certificates in the database that have been issued *to* this user. Observe that only fools delegate their privileges to this particular user.

confusing. There are custom SSL/TLS handlers that intercept calls to the traditional SSL/TLS handlers in XPCOM. But those handlers turn around and use the pure C NSS libraries to handle the SSL handshake. Our method only allows us to intercept method calls between the object-oriented XPCOM components. Pure C method calls cannot be intercepted by the same trick. So we need to use a different trick.

The SSL/TLS handler objects allow clients to set a callback function that retrieves the client certificate for the SSL handshake. (In the NSS documentation, this callback is known as the `ClientAuthDataHook` [12].) We can apply a second man-in-the-middle strategy to *this* function, by changing the function pointer to point to a function of our choosing. NSS calls the callback function when it needs a certificate.

Unfortunately, that's not the end of it. The custom certificate-retrieval callback only returns a single certificate—NSS builds the rest of the chain. Fortunately, there is a way to fool NSS into building an arbitrary chain. NSS stores its certificates in data structures with a lot of redundant information. These data structures contain a DER encoding of the certificate, as well as pre-computed fields so that it doesn't have to decode and re-encode the certificate repeatedly. But there's the rub: it uses the values of the pre-computed fields to decide which certificates to push onto the certificate chain, but the it uses the DER encodings to construct the bits of data actually sent across the network. And it assumes these fields are in-sync. By feeding it out-of-sync data, we can fool NSS to build a certificate chain based on the mock-up certificate fields, and NSS will end up sending arbitrary DER data across the SSL session. This DER data will, by more than coincidence, be the proxy


Register Proxy Certificates...

☑ Enable Proxy Certificates in SSL sessions

Choose proxy certs to use in SSL

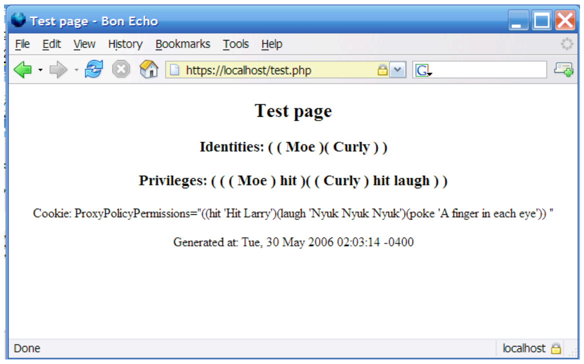
☐ Use Own Identity

Certificate Name	Delegated By	Issued By	Serial Number	Expires On
The Three Stooges				
Nicholas J. Santos	Moe	Moe	44:70:D5:CB	5/31/2006
Nicholas J. Santos	Curly	Curly	44:70:D6:15	5/31/2006
Nicholas J. Santos	Larry	Larry	44:70:D6:38	5/31/2006

Alert

Registration successful!

Figure 10: Registering two proxy certificates to use in an SSL session. The chains for both will be sent in client-side SSL/TLS authentication.



https://localhost/test.php

Test page

Identities: ((Moe)(Curly))

Privileges: (((Moe) hit)((Curly) hit laugh))

Cookie: ProxyPolicyPermissions="((hit 'Hit Larry')(laugh 'Nyuk Nyuk Nyuk')(poke 'A finger in each eye')) "

Generated at: Tue, 30 May 2006 02:03:14 -0400

Figure 11: This test page shows the values of environment variables SSL_DELEGATED_IDENTITIES and SSL_DELEGATED_PRIVILEGES.

certificates that we intend to transmit.

And that's how an extension can add limited delegation with proxy certificates to Firefox without modifying any existing code.[7]

5.2 The Apache Codebase

The Apache codebase is much smaller than the Mozilla codebase, and our application allows its source code to be modified. The changes made to Apache are thus much more lightweight.

[7] A lot of our reviewers were surprised by this result—in particular, that a Firefox extension could throw so much weight around and behave so intrusively. It's worth pointing out that the term "extension" is slightly misleading. "Extension" suggests that the software is sandboxed; that it can only "extend" the browser. But in reality, installing an extension is just as dangerous as executing a binary.

New Directives. The Apache build system leans heavily on an automated parser generator. A single macro can add a new configuration directive, and define the callback function that will process the arguments to that directive. The new proxy certificate-handling directives defined in Section 4.4 were designed to take advantage of this existing infrastructure. Adding them is not complicated. The `SSLMultipleIdentities` and `SSLExclusiveIdentity` directives are processed by changing global context variables. The two privilege directives are processed by adding them to a global privilege table, sorted by unique privilege name. For each directory-specific `SSLRequirePrivilege` directive, we take the corresponding directory context structure and give it a pointer to the required entry in the global privilege table.

Privilege Propagation. Apache comes packaged with a module, mod_usertrack, that enables tracking cookies. This module supplies the basis for the code needed to pass the supported privilege set to the client via a cookie. We register a hook function with the main Apache module to get called every time a client connects. When this function gets called, we can iterate through the permission table, encode it in a cookie, and add that cookie to the HTTP reply headers. (Actually, since the same cookie is transmitted each time, and no permissions can be added after the server starts, we just compute this cookie once.)

Certificate Verification. Because there may be multiple chains of proxy certificates in an SSL session, OpenSSL needs to be modified to accept a) proxy certificates, and accept b) multiple chains of them.

Proxy certificate support in OpenSSL is contingent upon the state of a particular environment variable. But the Windows code for reading this environment variable did not appear to be working correctly, so OpenSSL was modified to accept proxy certificates all the time.

Apache lets OpenSSL take care of the standard certificate verification, but sets a callback function to go through the certificates after OpenSSL has verified them, and do any custom verification. The OpenSSL verification function normally stops immediately as soon as it can't find the issuer for a certificate in the chain, then looks in the local store for a trusted chain of issuers. The verification succeeds iff it finds such a chain. If there are multiple chains, OpenSSL accepts the first chain and says that it is satisfied. This appears to be non-standard. We modify the OpenSSL verify function so that after it verifies the first chain, it looks at the first certificate in this chain and the first certificate in the chain of the "unverified" certs. If these two certs match, and the global flag for multiple chains is set, it calls itself recursively on the "unverified" cert chain to verify the other chains.

When the Apache callback function receives the verified certificate chains, it iterates through them again. For each chain, it looks for the end-entity (the first non-proxy certificate) in each chain, and pushes it onto an identity list. It also interprets the `ProxyCertInfo` extension's `policy` field, and determines which privileges are delegated all the way down the chain. The implementation of this part should be self-evident.

6 Related Work

SDSI/SPKI SDSI/SPKI is an alternative certificate standard that stresses simplicity over the complication of X.509.

The SDSI/SPKI group at MIT developed an Apache module and Netscape Communicator plug-in that allowed users to authenticate with the server using SDSI/SPKI certificates. (Project Geronimo was the name of the Apache module.) To accomplish this goal, they developed an entire new protocol on top of HTTP that performed this authentication. In their system, the server notified the client of the permissions it supported by sending an access control list (ACL) to the client during the authentication handshake. (Thus, the protocol handshake was used for authorization as well as authentication.) The client could then use this ACL to determine which certificates to send back [10]. This ACL solved the problem that we addressed by sending the permissions in cookies.

Our system differs in that it is also concerned with providing the user with an interface to delegate their credentials. We also depend more on the existing protocols and standards (X.509 and SSL/TLS) when we can, rather than creating new ones.

Greenpass The Greenpass project grafted SDSI-SPKI delegation on top of X.509 identity certificates for EAP-TLS. In that project, system administrators could maintain a secure wireless network without the hassle of verifying the identity of temporary guests. Regular users could delegate access to the network to their guests. This made the network more manageable and usable from both the administrative and end-user standpoints.

In order to sign these delegation certificates, both the regular user and her guest would visit a Web site (the guest via a captive portal). This site provided an interface wherein the regular user could verify the guest and create the certificate, and the guest could import the new certificate into her browser. Neither had to install new software; they only needed to run a trusted Java applet [4].

Notice that the Greenpass project and our project ran into a similar problem: how does Alice transmit a delegation certificate to Bob? We could circumnavigate this problem by using public e-mail. Because the guests in Greenpass did not have access to the network, they accomplished the same task with the assistance of an internal Web server.

Distributed Systems. Delegation offers a decentralized way to propagate privileges. It can also be used to delegate a limited set of privileges for a very limited time to a less trustworthy key. For these reasons, people working in distributed systems love delegation. They use it as a lightweight mechanism for granting privileges to temporary processes. It's lightweight because it doesn't require a central authority, and no new identities need to be created. The Grid created proxy certificates to take advantage of these features of delegation [15, 11]. Marchesini et al married Grid-style MyProxy to hardware trustworthiness levels [9]. Howell specifically extended Lampson's access control calculus to include delegation, so that he could use formal semantics to analyze distributed systems that lacked a central authority [7].

7 Conclusions

In the real world, users like to delegate privileges. If next-generation authentication systems (such as PKI) do not allow for this delegation, users will find a way to work around them—and undermine the security that drove adoption of the strong system in the first place. In this paper, we have presented both the design and prototype of a way to extend PKI (via standard client-side SSL) to permit this delegation.

When users can delegate rights to each other, we can end up with a user with a set of delegated rights from multiple sources. This raises the question: how can applications make sense of this heterogeneous set of rights? Consider some real-world examples. When a group of people elect a delegate to the U.S. Electoral College, they delegate a simple duty—to elect a president. But when a number of persons sign their power of attorney over to a single lawyer, a complex set of rules governs how the lawyer can use these rights, to prevent a conflict of interest. Clearly then, some applications need fine-grained controls over how to enforce delegated rights, and some don't. Further exploration there is one area of future work. Another area is exploration of the user interfaces involved in delegation.

There is a lot of political theory behind the design of X.509 and how it propagates authority. Engineers have politics too. Indeed, certificate theory raises questions about authority and trust that have been wrestled with since the Greeks. We have no hope of setting those issues to rest here. But the reader should be aware that one motivation behind this paper is the political ideal that Alice and Bob should have the authority to delegate their own privileges. This paper seeks to empower them with that authority.

Code Availability

We plan to make the code available for public download in 1Q2007.

References

[1] David Boswell, Brian King, Ian Oeschger, Pete Collins, and Eric Murphy. *Creating Applications with Mozilla*. O'Reilly, September 2002. Retrieved on-line from http://books.mozdev.org/index.html.

[2] T. Dierks and C. Allen. *TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.

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

[4] Nicholas C. Goffee, Sung Hoon Kim, Sean Smith, Punch Taylor, Meiyuan Zhao, and John Marchesini. Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In *3rd Annual PKI Research and Development Workshop Proceedings*, pages 26–41. NIST/NIH/Internet2, 2004.

[5] Peter Gutmann. X.509 style guide. October 200. Retrieved on March 9, 2006 from http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt.

[6] R. Housley, W. Polk, W. Ford, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. IETF RFC 3280, April 2002.

[7] Jon Howell and David Kotz. An Access Control Calculus for Spanning Administrative Domains. Technical Report PCS-TR99-361, Dartmouth College, 1999.

[8] Richard Levitte. *HOWTO Proxy Certificates*, May 2005. Retrieved on March 11, 2006 from http://www.openssl.org/docs/HOWTO/proxy_certificates.txt.

[9] J. Marchesini and S. W. Smith. SHEMP: Secure Hardware Enhanced MyProxy. In *Proceedings of Third Annual Conference on Privacy, Security and Trust*, October 2005.

[10] Andrew J. Maywah. An Implementation of a Secure Web Client Using SPKI/SDSI Certificates. Master's thesis, Massachusetts Institute of Technology, May 2000. Retrieved on-line from http://theory.lcs.mit.edu/~cis/theses/maywah-masters.ps.

[11] J. Novotny, S. Tueke, and V. Welch. An Online Credential Repository for the Grid: MyProxy. In *Proceedings of the 10th International Symposium on High Performance Distributed Computing (HPDC-10)*, pages 104–111. IEEE, 2001.

[12] Bob Relyea, editor. *Network Security Services (NSS)*. The Mozilla Foundation, May 2005. Retrieved on March 11, 2006 from http://www.mozilla.org/projects/security/pki/nss/.

[13] Nicholas Santos. Limited Delegation (Without Sharing Secrets) for Web Applications. Technical Report TR2006-574, Dartmouth College, May 2006.

[14] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*. IETF RFC 3820, June 2004.

[15] 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.

A Scalable PKI for a National Grid Service

Jens Jensen, David Spence, Matthew Viljoen
Rutherford Appleton Laboratory
The National Grid Service for the United Kingdom

February, 2007

Abstract

In this paper we describe work to expand the PKI for the UK National Grid Service (NGS), to integrate it with site authentication and improve usability. This work is complementary to the UK Shibboleth deployment. As the NGS grows to support wider and larger scientific communities, we investigate how we can improve usability by tying in Virtual Organisation management into the PKI framework.

1 Introduction

1.1 General Introduction

The UK National Grid Service (NGS) [13] runs Globus-based Grid middleware which depends on X.509 certificates for user authentication (Globus Security Infrastructure, GSI [31]). The UK e-Science Certification Authority [24] provides medium assurance [6] certificates for Grid users and e-Science projects in the UK, including the NGS. This Certification Authority (CA) must provide medium assurance certificates (section 2.4.1) because it is approved internationally (section 1.3) to identify users and hosts in the UK to international Grid collaborations. Conversely, although it primarily serves the UK, scientists using the NGS have collaborators across the world, and the NGS trusts certificates from other internationally approved CAs in order to facilitate these collaborations. In particular, interoperability between NGS and TeraGrid [22] is considered important.

Medium assurance, among other things, implies that users will have shown photo id to a Registration Authority (RA) operator, and that certificates have a maximal lifetime of 395 days (one year, plus 30 days within which users should request rekeying). Furthermore, many relying parties that use these credentials insist on having “meaningful” commonNames (i.e., bearing a reasonable resemblance to the person’s authenticated identity). For many purposes, this is too strong an authentication and doesn’t scale well to a large number of users ($\gg 10^4$ certificates), so we are deploying a scalable hierarchy primarily for NGS to expand the user base, with features to ease the account management. It is deployed alongside the UK Shibboleth federation [25], and is complementary to it, but is still entirely independent of it. We explain how they will interoperate.

Deploying a PKI for academic institutions is of course not a new idea. The innovation presented in this paper lies mainly in deploying it specifically for a national Grid, so we can tie in attribute and Grid account management, and in deploying the PKI alongside, and interoperating with, the Shibboleth federation. We will also briefly discuss other usability issues.

1.2 Related Work in this Area

From a high-level view, the architecture of the work presented in this paper is similar to that of SWITCHaai [21], partly because this work aims to solve some of the same problems. The principal difference is that SWITCHaai is entirely Shibboleth based.

From a more practical point of view, this work is similar to USHER, the US Higher Education Root[26]. We will look closer at this similarity in section 2.6.

The work presented here depends on technology developed in other similar projects, namely, MyProxy [14], SHEBANGS [17], and ShibGrid [18], as well as other related single sign-on work [7, 9].

1.3 International Accreditation

Although not fundamental to this paper, it will be helpful to briefly mention as explanatory background information that international Grid CAs are accredited by so-called Policy Management Authorities (PMAs). The Grid world is currently covered by three such, who together form the International

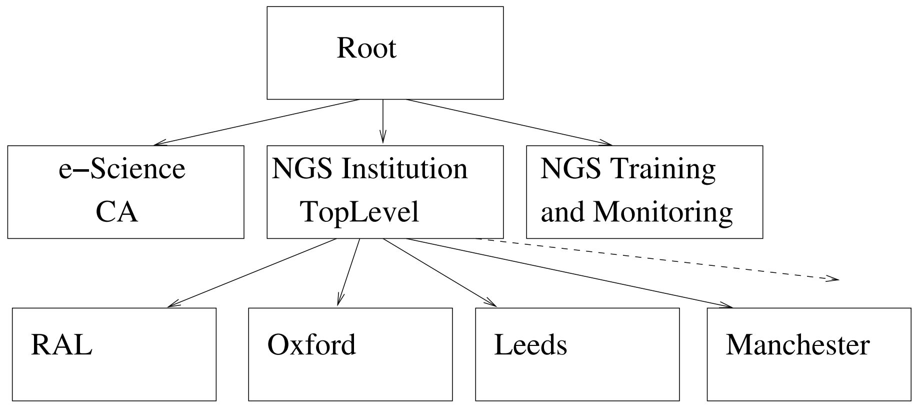


Figure 1: The UK e-Science Hierarchy

Grid Trust Federation, IGTF. Accredited CAs are then trusted by national and multinational Relying Parties (RPs), including the NGS, but the RPs are of course free to trust unaccredited CAs—indeed, this is often necessary to enable certain communities to access the Grids. Loosely speaking, it is the PMAs who impose upon their members that they operate to what we have called "medium assurance" in this paper, as a condition for accreditation.

2 Deployment

This paper discusses deployment of a hierarchy of *credential conversion* CAs, where each institution effectively runs its own CA which converts the institutions' site authentication to a short-lived X.509 credential (in the Grid CA context, such a CA is often referred to as a SLCS, a Short-Lived Credentials Service [8] (pronounced "slicks")). "Short-lived" usually means 12 hours, but is allowed to be "anything up to 10^6 seconds." [8]. One core difference is that although a SLCS acts as a CA, it does not need to issue CRLs.

This deployment brings us a wide user base, where essentially anyone from any such institution can get a certificate, but we lose the right to manage and control the user data that originally identified the user.

Indeed, our principal challenge was to widen the user base for the NGS, enabling also students and visiting scientists to obtain accounts via the authentication framework.

A secondary challenge was to improve the usability of the PKI. Usability is often seen as an obstacle to widespread Grid use; in some scientific communities users are unable or unwilling to learn basic PKI. The security required by a medium assurance CA will prevent these communities from engaging with Grid work.

2.1 The UK PKI Hierarchy

In July 2006 the UK e-Science CA deployed a new PKI based on a hierarchical model. This hierarchy was introduced at the same time as rolling over the certificate of the UK e-Science medium assurance CA. The new certificate for this CA is now subordinate to a root. Other CAs offering different services or providing other assurance levels have been deployed as part of this hierarchy. An example is a low-assurance training CA used for people new to the NGS.

Figure 1 on this page shows the UK hierarchy: a root CA ties together the internationally approved (medium assurance) CA, as well as the specialised hierarchy for NGS: the institutional credential conversion hierarchy and the training and monitoring CA.

The institutional credential conversion CAs, the SLCS, are anchored by a common SLCS trust anchor which itself is subordinate to the root. This anchor CA represents a single point of trust for any relying party (RP) that wishes to trust *all* credential conversion services in the UK, at least in principle. Its policy also ensures that a minimum set of requirements or level of assurance can be enforced between all credential conversion services, because it can impose specific policy and practices constraints upon its subordinates. These, of course, should not be set too high, or we would lose participant institutions.

Since a SLCS needs to be an online service in order to function, it is imperative that the private key be adequately protected to prevent it from being compromised. The SLCS anchor's policy states that subordinate certificates can only be issued when they are requested by a security device conforming to the FIPS 140-2 level 2 standard and the private key must not be exportable in any unencrypted form from that device (such devices can be obtained as USB tokens and are relatively inexpensive). Although no recovery is possible in case of hardware failure, the extra assurance won by so securing the key far outweighs the risks. Indeed, should the key be lost, a new certificate can quickly be issued by the parent CA.

Security concerns are also addressed in the SLCS anchor's policy by imposing the requirement upon the SLCSes that communication channels between them and their local authentication service, as well as the NGS authorisation services, are secure. In addition to this, each SLCS is required to log all credential conversion so that NGS traceability and accountability requirements of access mechanisms are satisfied.

2.2 Architecture

Figure 2 on the next page gives an overview of the architecture described so far and how this fits in to the wider NGS and site infrastructures.

As we are seeking to lower the barriers for users to user the NGS, we assume that the user will be using some easy-to-user *User Interface* software to manage their access to the Grid, to which we can make minor changes to support the SSO infrastructure (although we do not preclude the use of common command line tools).

The key component of the infrastructure is the *Credential Translation Service* or SLCS. There is one of these per site and its main function is to validate the user's site identity by calling out to the site's authentication infrastructure and then generates a Grid credential (a short-lived X.509 certificate) for the user. In this respect, it is similar to the Shibboleth IdP—see section 2.7.

The Credential Translation Service can also call out to a VOMS server to obtain a VOMS attribute certificate for the user. This would normally be to the NGS VOMS server, but others could be imagined, such as a site VOMS server.

2.2.1 Where Are You From?

Most of the work described in this paper applies to local clients, running within the site. However, we have also thought about central interfaces (shared between sites).

Such interfaces could also run on the user's machine (e.g. an applet), but would typically be on an NGS server (e.g. the NGS portal), or a third party server (e.g. a project portal). Unlike local clients which "know" which credential conversion service to contact, a discovery mechanism is needed for non-local services, and, worse yet, potentially a different mechanisms for each one. Any such service would of course play a role of the WAYF in a Shibboleth Federation where each client selects his or her home institution and is redirected, but for this project we wanted to simplify the selection, so the client does not even need to select a home institution.

The easiest way to accomplish this is to configure a lookup mechanism which notices which address the client is coming from, and uses it to contact the right credential conversion server. We do indeed lose the ability to support "roaming" clients with such a simple scheme, but gain the simplification of not asking the user for redirection. For central portals, the conversion service must also be reachable through the site firewall which complicates site deployment slightly.

On the server side, a trusted repository of the SLCS sub-CA certificates will have to be provided, along with the repository provided by the PMAs for traditional CAs.

2.3 Implementation

While there is no requirement to do so, the "obvious" way to implement a site SLCS is via MyProxy [14].

We discuss the MyProxy solution further in this section, but it is also worth mentioning Microsoft's Windows Server 2003 which also contains features to run a CA, but will not, as far as the authors are aware, be able to contact an external VO attribute server to add attributes to the certificates.

By using MyProxy we can bridge many of the site authentication infrastructures in use to the GSI/PKI world. MyProxy can be configured to provide certificates generated by an internal or external CA. We can also support advanced users who have long-term certificates by enabling them to upload proxies generated from their certificates (they are not permitted to upload the certificates and private keys themselves, by the CA's policy). This was the approach taken in the ShibGrid project [18]; see also Shibboleth discussion in section 2.7. For users who do not have long-term certificates, the service generates short term certificates and keys.

MyProxy can also support SASL [12] (allowing Kerberos [19] authentication), PAM (allowing LDAP [29], RADIUS[16] and many other authentication systems) and PubCookie.

Although it is simple to use the MyProxy command-line tools to leverage this functionality, this could potentially present users with quite a barrier to overcome. Many of the target audience traditionally balk at security in general, certificates in particular, and anything that cannot be clicked with a mouse. Therefore, we aim to simplify the process as much as possible, and hide the certificate/proxy process. For our own site's authentication infrastructure, Microsoft Active Directory (which happily, for the purposes of this work, is

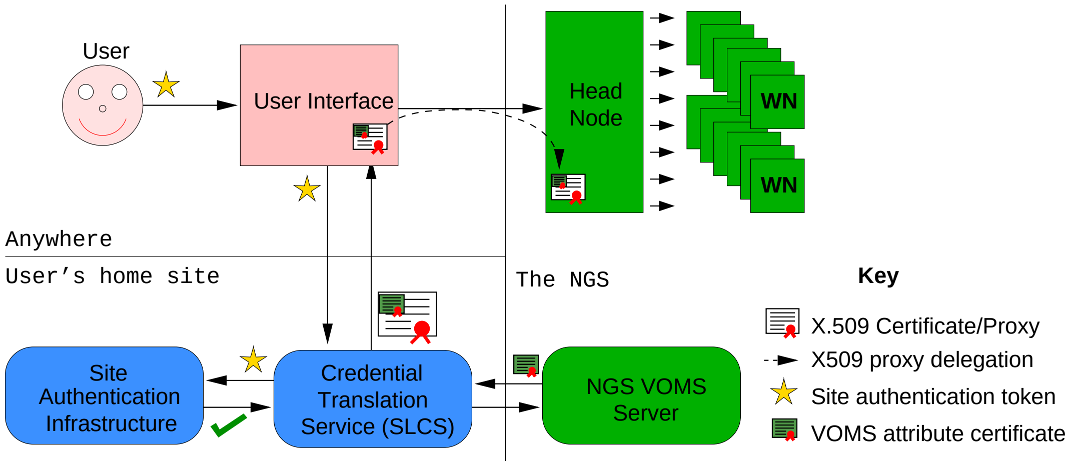


Figure 2: The architecture for credential conversion in the NGS

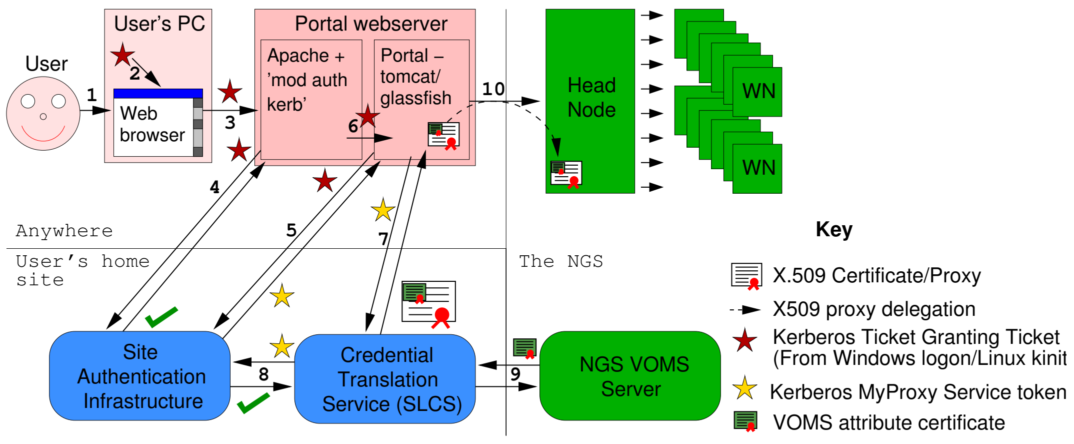

User
User's PC
Web browser
Portal webserver
Apache + 'mod auth kerb'
Portal – tomcat/ glassfish
Head Node
WN
WN
1
2
3
4
5
6
7
8
9
10
Anywhere
User's home site
The NGS
Key
Site Authentication Infrastructure
Credential Translation Service (SLCS)
NGS VOMS Server
X.509 Certificate/Proxy
X509 proxy delegation
Kerberos Ticket Granting Ticket (From Windows logon/Linux kinit
Kerberos MyProxy Service token
VOMS attribute certificate

Figure 3: Authentication process from a portal.

equivalent to Kerberos 5), we have integrated support for Kerberos authentication to MyProxy into two easy to use Grid access methods:

- **Portal access** (Figure 3.) Normally, users who wish to use a Grid portal would have to upload a proxy of their Grid certificate to a MyProxy server before logging on to the portal. At the portal they would then have to provide the hostname of the MyProxy server along with the username and passphrase they used to store their proxy. The portal would then contact that MyProxy server with the given detail to obtain a certificate. In our set-up they instead simply visit the portal which picks up their Kerberos Ticket Granting Ticket (TGT) and it uses this to contact the Kerberos-enabled MyProxy-with-CA, which generates a certificate for the user. The portal has to be trusted for delegation by the Kerberos Key Distribution Centre (KDC) for the user to delegate its TGT to the portal.

- **GSISSH Terminal** (Figure 4 on the next page.) We have also developed additions to a Java-based Grid Security Infrastructure enabled secure shell (GSI-SSH) terminal, GSI-SSHTerm [20], which runs in a user's web-browser or as a standalone Java application and provides terminal access to Grid resources. These additions automatically call out to the Kerberos-enabled MyProxy with CA to attempt a conversion of the user's local Kerberos credential to a Grid credential, when a user tries to log on to a remote resource.

Both of these methods rely on a specially patched version of the Java Commodity Grid (CoG) Kit [28] which allows SASL+Kerberos authentication to MyProxy servers. In choosing to start with Kerberos infrastructure support we hope to support

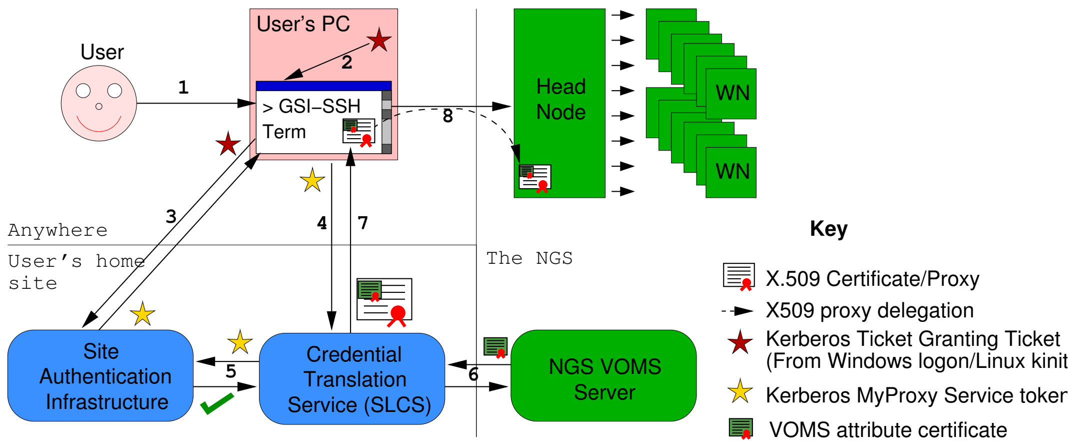


Figure 4: Authentication process from the GSISSH Terminal.

many institutions as this also encompasses systems built on Active Directory. Both these methods fall-back to username/password authentication (but using site passwords) to the MyProxy with CA if Kerberos tokens are absent.

In the case of Portal access we have had experience with both the Tomcat and Glassfish servlet engines. In both cases these would run behind an Apache instance running on the same machine and utilising mod_auth_kerb.

On the server side each institution must install their own dedicated MyProxy server which is configured to work correctly with their site authentication infrastructure. Although the configuration will be different depending on the site authentication infrastructure there should be enough commonality within the same technology to make the provision of example configurations useful.

The policy for the MyProxy's CA certificate means that it must be stored on a hardware token, for example a USB key-token. We have already undertaken the necessary changes to the MyProxy server code to allow it to connect to any hardware token supported through the openssl "engine" mechanism[1]. This set includes nCipher products, CryptoSwift and key-tokens supporting PKCS#11 and PKCS#15 (through components from the OpenSC project[2]). We selected to use the Aladdin eToken, through the PKCS#15/PKCS#11 interface. We have of course contributed these changes to MyProxy.

Another issue is which Distinguished Name (DN) to give to users. The policy for the CA certificate requires that the MyProxy CA will sign its certificate in a particular namespace. Further to this we only require that each user's DN is unique to that user, traceable to that user and consistent over time for that user. For some institutions it may simply be that they append `/CN=<userID>` to the CA's namespace, for others they will call out via LDAP to obtain more user information to generate a DN that is closer to those currently in use the UK e-Science CA. Within our own Active Directory domain, for example, we chose to use `/UID=<userID>/CN=<firstName> <lastName>`.

2.4 Scalability, Policy, and Assurance

2.4.1 Levels of Assurance

The Level of Assurance (LoA) of a CA or of a certificate issued by that CA is an indication of the extent to which an entity has been identified as the owner of a credential issued by that CA. Whereas the US government has proposed using four LoAs [15], Grid CAs have traditionally employed two LoAs. These map approximately to the US governments LoA levels 1 and 2 (*ibid*, sections 2.1, 2.3) with only level 2 being accepted internationally, level 1 is usually for internal use. This government-proposed mapping policy was expanded to recommended practice by the US National Institute of Standards and Technology in NIST-800-63 [5], which may well be indirectly behind the Grid requirements for "medium assurance" described below.

1. **Low LoA** Little or no identity verification has taken place during the issuance of the certificate. Such CAs are typically used to issue training or test certificates to access resources.
2. **Medium LoA** The certificate applicant is required to meet a representative of the CA, the Registration Authority (RA) in person during

[1]`http://www.openssl.org`
[2]`http://www.opensc-project.org/`

the certificate application process. Furthermore, the applicant is required to present an original photo id document as proof of identity, while the RA is required to retain a copy of the document for a specified period of time (at least as long as the lifetime of the certificate). The RA may stipulate the accepted forms of documents, typically an id card of the institution where the RA is situated, a driving license or passport. The Classic Authentication Profile of the IGTF requires all Grid CAs to be at least Medium LoA to be accredited to the IGTF.

3. **High LoA** If a Medium LoA CA is considered inadequate for particularly high risk relying parties then additional requirements may be stipulated covering identity verification or how certificates may be issued. For example, resources protecting particularly high-risk data may choose only to accept certificates from a CA which includes biometric verification of applicants and who only issues certificates protected by secure cryptographic devices which can prevent the private key of the certificate being exported.

It is worth noting that the Grid's medium assurance falls short of meeting NIST's Level 3. For example, most CAs make "best efforts" to revoke (issue a new CRL) within "one working day" (of the revocation request being approved) rather than 24 hours. Globus proxies and delegations permit credentials to be passed around and may live longer than 24 hours ("freshly issued", [5] 8.2.3.1.) "Picture id" is required for all, but many CAs accept photo id issued with a site account in addition to driver's licence or passport. Indeed, the emerging MICS profile (work in progress, mainly led by TAGPMA) enables a CA to skip the RA step and issue certificates directly for site accounts. On the other hand, no Grid CA except the project catch-all CAs accepts remote verification. A full comparison is interesting but beyond the scope of this paper.

HEBCA has started looking at the Grid assurance levels, to evaluate the feasibility of bridging the Grid PKI to the US Higher Education. Although in progress, we describe this work further in section 2.6.

2.4.2 Scalability

Consider a medium assurance CA issuing certificates to 10^6 users (the estimated number of higher and further education users in the UK is of this order of magnitude). Even assuming that no new users are added, the Registration Authorities (RAs) must process 10^6 renewal requests per year, which translates to over 2500 per day (if every day is a working day). With 250 operators, that is 10 per day per operator, assuming it is distributed evenly (even though the RA need not verify the user's identity, they must still perform a simple check that the user is still associated with the project or organisation).

Moreover, a new requirement is being added by the internal Grid CA community, that users must reauthenticate with their RA every N years, where N depends on how the private key is stored, as well as other factors; $N = 5$ is typical. That means on average 200000 reauthentications every year.

Even if people didn't have to reauthenticate, a medium level Grid CA still requires them to generate their own key pairs, and on relatively secure systems, e.g. their own desktop machine, not a shared service. Thus, people have to manage and convert their own keys and certificates, and the support load required to support 10^6 PKI-novice users managing their own keys would be beyond the capacity of NGS support.

One of the aims of this project is to *be scalable to* a million users distributed over 10^3 institutions, increasing the usability at the cost of the assurance that the CA can guarantee.

Another aspect of a medium assurance CA, as mentioned above, is that people have to show photo id during the identification process. People may argue that site authentication is typically at least as good as that: your employer probably saw your passport, birth certificate, etc. However, there are two problems with this approach: firstly, the CA has no access to this information, so cannot, for example, rely on it to ensure that the distinguished name (DN) in the certificate is never reallocated to another person. Secondly, the CA cannot *guarantee* that this process has taken place: many sites have visitors or contractors who are also in the site database, and may not be managed as strictly as those in the payroll database.

The other aspect of scalability is that of the number of CAs. In this model, we have one CA per institution. However, current Grid middleware must have *each* CA installed, whether it is an end entity issuing CA or not. This is fine for the "standard model" with one per country [4], but not with $N \gg 1$ per country for a global collaboration. It should be feasible for NGS to trust M countries and N institutes, though—the number of certificates is $M + N$, rather than MN. A similar situation is found in TeraGrid which is itself served by more than one CA. Furthermore, several related middleware problems have been found where the software has "mysteriously" failed once the number of CAs has reached a certain limit (usually around 60), but these are mostly fixed.

2.4.3 Account management

Another scalability problem is that of authorisation, which has traditionally been done by the Subject DN, in the gridmap files.

To get around this problem, NGS will, like many other Grids, be using VOMS to manage its user authorisation. VOMS [2] provides a central Virtual Organisations (VOs) service that manages VO membership and roles. Ordinarily the user, using either their certificates, or more commonly a Globus proxy [23], accesses the VOMS server and gets another proxy, this time with attributes describing their VO membership and optional roles.

One of the advantages of generating the certificates specifically for the NGS Grid, and the fact that they are short-lived, is that we can embed authorisation attributes in them, without requiring the user to perform the second step of contacting a VOMS server. This is particularly important because we aim to simplify or even hide the process from the user, so if the attributes can be embedded in the proxy generation step, that simplifies the process for us and improves usability for the user.

Such work was done in the Manchester "SHEBANGS" [17] ("Shibboleth Enabled Bridge to Access the NGS") project, where a MyProxy server is contacted for user attributes which are then embedded in a certificate (independently of VOMS, but compatible). We can leverage this work to provide attributes for NGS.

The site service needs to call out to, or cache information from, the NGS VOMS server, but should of course be able to fall back to "plain" (non-attribute) certificates in case the VOMS server is unreachable. This attribute management will have to be independent of the site (Shibboleth) Attribute Authorities, because they pertain to NGS work, not to site databases; nevertheless, there may be cases where NGS software will need to authorise based on both types of attributes. Combining these attributes is future work, but may be able to use the work from GridShib [30] which aims to make site attributes available to Grid resources.

2.5 Usability

X.509 digital certificates are the most widely used method of authenticating users to Grids. Prior to using the Grid, users are often required to request certificates using some form of a user agent, typically a web browser. Once issued, the certificate typically needs to be converted to a form that is usable by Grid middleware, stored in an adequately secured manner. New Grid users therefore need to learn the fundamentals of key management as well as the processes of revocation and renewal and under which circumstances these must be done.

It is thus clear that, unlike other applications of PKI such as smart cards where the mechanics of PKI are shielded from the end user, Grid users are required to have a basic grasp of using digital certificates. Whilst this may be reasonable for the majority of Grid users at present who come from a scientific computing background, there are increasing trends, not only in the UK but worldwide, for Grid computing to be used in multidisciplinary research, and particularly so for the NGS.

The authors of this paper, who not only manage and run the UK e-Science CA but also work with the helpdesk which deals with user queries related to the CA, frequently encounter frustration from end users who view digital certificates and their usability issues as a barrier to using the Grid. This may be partly alleviated with a CA that is easy to use and well documented; work has been done to address these issues [10]. However, the overhead of learning about digital certificates remains, and if the current authentication methods continue to be used, these usability problems will be encountered by the increasing number Grid users from non-computing domains.

In the work described in this paper, usability is improved because basic certificate management can be done by portals and other tools on behalf of the user, and locally at the user's site. It enables sites to provide the "single sign-on," i.e., single password, mechanism by integrating the site's credential conversion MyProxy with the site's authentication system; this is also one of the advantages of Shibboleth. Moreover, we can improve usability further by removing the need for the user to call out separately to a VOMS server.

For certain types of client tools, we can even hide the certificate/proxy generation from the user completely—every single step is performed transparently by the tool on the user's desktop, contacting the local conversion service using the user's cached *desktop* login credential, and the conversion service in turn contacts the global NGS VOMS server. Thus, the user need not even know that the client tool has generated a certificate on behalf of the user.

As discussed in section 2.3 we can do this with Microsoft Active Directory—and equally Kerberos V—but the account management step is currently missing. We have the account request step in ShibGrid, but of course it requires Shibboleth. The obvious solution is to build a desktop registration client which the user can use to request a personal account. A smarter solution would see the user registering with GridShib-exported site attributes, being joined to an NGS VO by a site-local administrator, and would access NGS resources on the basis of attributes alone. However, as discussed in section 3 this requires a greater trust in the site's operations.

2.6 The USHER Hierarchy and Levels of Assurance revisited

In this section, we briefly cover the USHER work since it is similar to the PKI-aspect of the work described in this paper.

The US Higher Education Root [26] is operated by Internet2 to establish a PKI for educational institutions. It consists of a root which issues certificates to institutional CAs, and it imposes requirements on the institutional CAs. USHER imposes the requirement that certificates are issued by subordinates

> "using a process that is at least as strong as its existing practice for managing accounts for central services such as electronic mail, calendaring, and access to central file storage[27]"

This is equivalent to the assurance provided in Shibboleth, and to what we have in our project—with the important difference that we do not have an explicit commitment from the site. In fact, we do not know the exact practices implemented by the site to meet these requirements, nor do we have the ability to audit the site's practices.

The principal difference between USHER and our PKI deployment is that our PKI deployment is much simpler, partly because we have no need for the institutions to trust each other's credentials. Only the resource providers need to trust the credentials of all the institutions, and the resource providers, although individually members of participant institutions, are all part of the NGS. There is a world of difference between asking the University of Oxford, say, to trust a CA, or to ask the NGS administrator at Oxford to trust the same CA. In a sense, we achieve the same result as USHER via the back door—via the project, not via the institution. The drawback is that our CAs are not widely trusted, but we can live with that because we need them only for the NGS.

Another simplifying fact is that we have no need for long-term credentials, so can rely on the institutional CAs to perform the conversion as and when it is needed.

Of course, our work is more than a PKI deployment: we are helping sites deploy MyProxy-based credential conversion servers, along with client tools and NGS-specific software such as portals that use it. Thus, our PKI deployment is tailored to fit the project rather than being a general purpose PKI.

HEBCA/USHER started work [3] to map Grid (more precisely, IGTF) *authentication profiles* to their own assurance levels (HEBCA is the EDUCAUSE US Higher Education Bridge CA)—note that Grid authentication profiles are all roughly "medium assurance." This sort of policy mapping exercise is commonly done for bridge CAs [11], although profiles are often not directly comparable even when they both claim to follow RFC 3647. In this case, the exercise (*ibid*, p. 9) showed the IGTF "classic" authentication profile (the first implementation of what we have referred to as "medium assurance" in this paper for Grid CAs), being equivalent or slightly worse to HEBCA level "Rudimentary" and the Federal PKI level C-4.

Although interesting from an academic point of view, this work should not yet be relied upon for mappings. Usually clarifications and policy adjustments bring partner policies closer together.

From our (NGS') perspective, the importantance lies in establishing the policy mappings, even just tentatively. NGS has in the past been required to "interoperate" with TeraGrid, and this has so far been accomplished between the IGTF-approved CAs, with lengthy separate reviews for those pending such accreditation. For example, the UK e-Science CA is trusted by TeraGrid for this reason. Experience has shown, however, that the US is a large country with many diverse PKIs, and usually the NGS has to trust one or more non-IGTF-accredited CAs. We have had requests from users in the US with no "obvious" CA for NGS access, and in the future it would be convenient if their institutions could get CAs via USHER, at least if they have a need for more than a few certificates (otherwise a project-related "catch-all" CA could do the job). Mapping the USHER policies with known Grid policies (namely, medium assurance) will greatly help the NGS evaluate the trust of those CAs.

As regards the UK and the PKI described in this paper with the institutions participating in NGS, there is in general not much we can do about getting the institution to commit to a certain level of assurance. As we described above and discuss in more detail in section 3 we are not exposed to the institution's user identity process, nor can we demand legally binding documents to this effect, since this was supposed to be a lightweight PKI to complement the Shibboleth deployment.

2.7 Shibboleth Interoperability

2.7.1 Credential Conversion with Shibboleth

One could create a central CA portal to which each user connects to obtain a certificate. To ensure that the home site database is queried, users must use Shibboleth to access this CA. Indeed, this is more or less the model SWITCH used for one of their CAs. In the UK, we chose the approach of distributing the subordinate credential conversion CAs to sites, for three primary reasons:

Firstly, the practical reason, because there isn't yet a widespread Shibboleth deployment in the UK.

Secondly, because of the personal data; we cannot rely on sites being able (or willing) to release sufficient personal data from their Attribute Authorities to uniquely identify the user and map them to the same DN every time: at the time of writing (Oct 06) it is not clear whether sites in the UK Federation are required to publish anything other than eduPersonScopedAffiliation (for an explanation of the eduPerson schema please see [1]). We would need at least eduPersonTargetedID but that in turn will not be sufficient to satisfy those Grid resources that require "meaningful" commonName (CN). Even if NGS itself chooses to accept pseudonymous identities, which could be imposed, if at all, only on the core sites, some affiliated resources have already that pseudonymous identities will not be accepted.

In fact, the base UK Shibboleth Federation aims to be pseudonymous, with explicit agreement between Identity Providers (IdPs) and Service Providers (SPs) whenever extra attributes are required. This means the institutions have to worry about data protection issues. For some, the provision of an institution's IdP may even be out-sourced with no link back to the institutions user database, which implies that these attributes cannot be provided. Finally, some sites with NGS users may just not join the Shibboleth federation.

Thirdly, using "smart clients" we can leverage the sites' internal authentication infrastructure without compromising site security. Unlike the Shibboleth portals where users need to select their home site and then log in again to their home site, our smart clients can pick up the user's site authentication token and transparently generate a VOMS-proxy with which the clients can access the NGS Grid on behalf of the user. Not all clients are "smart" enough to do that, but as mentioned earlier (sections 2.2 and 2.5) for parts of the userbase we aim to even hide that the proxy exists. As mentioned in section 2.2, compared to Shibboleth, we lose the ability to support roaming (off-site) users with the work described in this paper.

It is worth looking at other aspects of the distributed vs. central issues in more detail:

For the central model, running a central high-availability CA is a big commitment. The distributed model distributes the burden: a site's server can still go down, but at least only that site is affected, not the whole Grid. Another advantage of the model proposed in this paper is that not even the VOMS server needs to be high availability: if it is unavailable, sites can fall back to cached information. Information that is potentially slightly out of date is better than none at all.

Furthermore, if there is a central CA portal creating credentials for the users, then users need to use that credential either wholly within that portal, or export it from the portal. For NGS, though, there will be more than one portal, and not all NGS work will be done via portals. Nevertheless, using portals that call out to a central high-availability MyProxy is an option we may use in the future, when we are sure that all of the NGS is covered by the Shibboleth federation. This, too, would alleviate the lack of support for roaming users, since they could use the site credential conversion when on-site, and Shibboleth as a more complex alternative when off-site. In that case, a Shibboleth federation covering the NGS userbase adds value to the work described in this paper—or vice versa—although there is a danger, with more than one issuing authority, that the user will have more than one DN. This is discussed further in section 2.7.2 below.

In the distributed model the certificates are generated locally (within the user's home institution) and can be used locally, on the user's desktop, as we mentioned above, and the service does not need to be exposed to the outside world. The credential conversion service can pick up attributes that the institution's Shibboleth IdP may not publish, such as the commonName (CN). Whether the service is allowed to expose this to the NGS is a question of site data protection policy. The local credential conversion services also allow more robust traceability if pseudonymous DNs are used.

Moreover, an institution providing user account management with higher assurance can use that, internally or externally, without having to live with certificates created with a Shibboleth federation's lowest common denominator level of assurance.

Finally, it is relatively easy to add a new institution to the NGS framework whereas joining the UK Shibboleth federation is more work—the latter requires a legally binding commitment on behalf of the institution as well as setting up and running a high-availability IdP.

2.7.2 Accessing the Grid via Shibboleth Portals

Some version of the central Shibboleth-portal model, as discussed in the previous section, is likely to be implemented in the long term, via an NGS portal. Work is already being done to "Shib-enable" one NGS portal [18], and it will be feasible to integrate certificate generation into the portals. In this central model, the private key will be generated remotely (by the portal), and the certificate by a central service accessed by the portal. The challenge here is to ensure that both views are consistent: the portal views — there may be more than one portal — and that of the site's credential conversion service.

Within this world a user may have many DN-based identities: a DN from the UK e-Science medium assurance CA or one of its international peers, a DN from the Shibboleth credential conversion service (usually from a central Shibboleth portal), a DN from her institution's local credential conversion service and maybe even DNs from other institutions where she may have accounts. How does a Grid resource know that all these DNs are the same user?

Maybe it doesn't have to know. As long as authorisation *only* depends on the VO attributes (neither site attributes, nor the identity), the user will have the same access rights, assuming of course that the user gets the same VO attributes for each identity. However, current middleware deployed on NGS sites still depends on gridmap files, i.e., identity-based authentication.

Tying the same VOMS attributes to several identities, or more generally run a database that knows about the identity mappings, is a problem we cannot currently solve. It relies on the user's collaboration, but even honest users may be put off by the work required to register DNs centrally—as we explained (section 2.5), many target users don't want to know about certificates. We could go some way towards this goal by comparing commonNames (CNs); for the less common names (less common commonNames if you will) it would give us an indication of when two different DNs represent the same user, and this task could even be automated but would still need human review.

Meanwhile, our assumption is that this will not be a problem as most users will, in general, stick with the same identity. Users might make changes to which way they log on, but they probably will not keep changing between different methods, especially if their institution only supports one form of credential conversion service. This behaviour will be enhanced by allowing the same set of resources, services and access methods through all types of identity.

2.8 Status

We finally get to the status of the deployment. The hierarchy has been set up, but the access has only been tested locally within two of the core sites (namely, University of Oxford and Rutherford Appleton Laboratory). Independently, we have tested the attribute services with both VOMS and SHEBANGS, but this work has so far only been in the test phase because the NGS does not yet use VOMS for VO management in production.

Furthermore, we already provide both "normal" and single sign-on MyProxy services for the NGS. Ongoing work over the next months will see wider deployments to core sites and further integration of the independent components, primarily combining the SHEBANGS work with the site credential conversion service at University of Manchester.

Deployment will be further enhanced by deploying the required software—all but the keys—on a single CD image, similar to the NAREGI CA-on-a-CD or Scott Rea's (Dartmouth College) OpenCA-on-a-CD (but of course using MyProxy instead of OpenCA).

3 Security Issues

No CRLs are published. It is common practice that proxies [23] live about 12 hours, but as we mentioned, they could be valid for anything up to 10^6 seconds. This leaves a window in which a compromised credential could be misused, but the same problem is found in general with long lived X.509 certificate where the user must first notice that the credential has been compromised, must then request revocation, under some circumstances that request will have to be approved (usually when it hasn't been signed with the user's private key), then the CRL has to be issued, and finally, the RPs will have to download the CRL. We thus do not consider the lack of CRL a particular security risk, although the *architecture* should encourage users to protect their private keys, since, as mentioned above, we cannot expect users to be experts. Hiding the key from the user helps as long as the local client is careful to store it on local disk. Backups are not necessary because the client will be downloadable and the identity can be easily regenerated by the user.

The quality of site identification has been a contentious issue for many years, particularly for credential conversion CAs seeking to be accepted by international Grid projects. It is hard to persuade collaborators that your identification process is good enough when the CA manager is not responsible for the user data, and in general has no access to it. It is all the more difficult when the site database contains external users, contractors, or temporary staff. In the few cases where Grid projects have accepted such a CA, the credential conversion has been at the *same site* as the CA (FNAL, CERN), and the CAs have been able to enforce a security level necessary for medium assurance by using existing assurance level flagging in the site database (or very occasionally the Grid project has pragmatically decided to trust the CA until a problem occurs). Indeed, most sites would be reluctant to share their user data with a CA auditor, because it would violate data protection policies. However, although the sites provide no such assertion to the NGS in this framework, they do in general use their site databases also for access to more "precious" resources, e.g., internal financial information. We have thus decided that we can reasonably expect sites to operate the databases to a "satisfactory" level, even if we have neither documentation for what this level is, nor a binding commitment to operate to it. This trust is particularly important with pseudonymous credentials where we rely on the site to maintain the mapping to the user's real world identity, but also for sites where access is potentially granted to the site using site affiliation such as the Shibboleth site attribute (scopedAffiliation), i.e., access is granted to all users on site at the site's discretion. A site found to abuse this access privilege can be revoked from accessing the NGS, and the embarrassment factor will most likely fall upon the site rather than the NGS itself.

The scalability issue, as described in section 2.4.2 means that, for the purposes of this NGS deployment, medium assurance is too strong. However, there may well be cases where the NGS *will* require a higher level of assurance. People requiring such access will either have to get a certificate from the UK e-Science CA (for example, this is required to access TeraGrid), or will have to prove independently that their site authentication was sufficiently strong. This assurance level can subsequently be managed as a VO attribute. CERN has taken this approach with their internationally approved MICS CA: it issues certificates only to users in the site database with one of fourteen different status, and one type of external contractors. Each of those status flags ensures that the user has shown appropriate photo id to the CERN user office.

Finally, for access to, e.g., certain medical images (non-anonymised), or financial data, it may be that medium assurance is not strong enough. High assurance cannot be provided within this PKI, partly because we have no explicit commitment from the institutions. Rather, it would have to be provided by either a separate CA, or with special certificates issued by the e-Science CA with flags (e.g., policy OID) to mark it as high assurance.

4 Acknowledgments

This work is funded by the National Grid Service in line with its provision of a national Grid CA and Grid support; the closely related work in ShibGrid and SHEBANGS is funded by the UK Joint Information Systems Committee (JISC) under the current (Apr '04-Apr '07) core middleware development programme.

This document typeset with LaTeX.

5 Conclusion

In this paper, we have described an architecture for a Grid PKI for the UK National Grid Service, NGS, deployed alongside and interoperating with the national Shibboleth deployment, as well as the existing global Grid PKI. The work aims to provide "NGS access for the masses" and be scalable to a large number of users, and to improve usability for non-technical users by making—for certain types of client tools—the entire certificate and proxy issuance process hidden from the user. We have described how the deployment is lightweight, requiring no legally binding commitment on behalf of the institutions. We have described scenarios where we grant access to the NGS based on site or Virtual Organisation membership attributes. We have discussed how we weigh the associated security concerns, improving them where possible and accepting them where not.

References

[1] Internet 2. Eduperson specification. Identifier: Internet2-mace-dir-eduPerson-200312, December 2003. Version 200312. Available at `http://www.nmi-edit.org/eduPerson/internet2-mace-dir-eduperson-200312.p%df`.

[2] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K. Lrentey, and F. Spataro. VOMS, an authorization system for virtual organizations. *Lecture notes in Computer Science*, (2970):33–40, 2004. Grid Computing, First European Across Grids Conference, Santiago de Compostela, Spain, February 13-14, 2003.

[3] P Alterman and S Rea. Policy Mapping Grid CAs. `http://indico.na-df.rnp.br/indico/materialDisplay.py?contribId=10\&amp;%sessionId=1\&amp;materialId=slides\&amp;confId=15`, Nov 2006. 3*rd* TAGPMA meeting, TACC, Austin Texas (URL checked Feb 07).

[4] J. Astalos, R. Cecchini, B.A. Coghlan, R.D. Cowles, U. Epting, T.J. Genovese, J. Gomes, D. Groep, M. Gug, A.B. Hanushevsky, M. Helm, J.G. Jensen, C. Kanellopoulos, D.P. Kelsey, R. Marco, I. Neilson, S. Nicoud, D. O'Callaghan, D. Quesnel, I. Schaeffner, L. Shamardin, D. Skow, M. Sova, A. Wäänänen, P. Wolniewicz, and W. Xing. International grid CA interworking, peer review and policy management through the European DataGrid certification authority coordination group. In P. Sloot, A. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Advances in Grid Computing — EGC 2005*, LNCS3470, pages 285–295, Amsterdam, The Netherlands, February 2005. Springer.

[5] William Burr, Tim Polk, and Donna Dodson. Electronic authentication guideline. NIST Special Publication 800-63 Version 1.0.2, April 2006.

[6] R Butler and T Genovese. Global grid forum certificate policy model. http://www.ggf.org/documents/GFD.16.pdf, June 2003.

[7] D. Byard and J. Jensen. Single sign-on to the grid. In *Proceedings of the 2005 UK e-Science All Hands Meeting*, September 2005.

[8] TAGPMA. Editor T. Genovese. Profile for short lived credential services X.509 public key certification authorities with secured infrastructure. Reference: IGTF-AP-SLCS-20051115-1-1, November 2005. Version 1.1. Available at http://www.tagpma.org/files/IGTF-AP-SLCS-20051115-1-1.pdf.

[9] J. Jensen, D. Spence, and M. Viljoen. Grid single sign-on in CCLRC. In *to appear in the proceedings of the 2006 UK e-Science All Hands Meeting*, September 2006.

[10] J. Jensen and M. Viljoen. Usability of the UK e-Science Certification Authority. *UK e-Science All-Hands meeting*, 2005.

[11] Mark Luker. A bridge for trusted electronic communications in higher education and federal government. http://www.educause.edu/ir/library/pdf/ERM0203.pdf.

[12] J. Myers. Simple authentication and security layer (SASL). Request for Comments (RFC) 2222, October 1997.

[13] National Grid Service. http://www.ngs.ac.uk/, Oct 2006.

[14] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the Grid: MyProxy. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), San Francisco, California, USA*, pages 104–114, August 2001.

[15] US office of management and budget. E-authentication guidance for federal agencies. http://www.whitehouse.gov/omb/memoranda/fy04/m04-04.pdf, December 2003.

[16] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote authentication dial in user service (radius). Request for Comments (RFC) 2865, June 2000.

[17] SHEBANGS (Shibboleth Enabled Bridge to Access the National Grid Service). Details at http://www.sve.man.ac.uk/Research/AtoZ/SHEBANGS, June 2006.

[18] D. Spence, K. Tang, R. Allan, M. Dovey, N. Geddes, J. Jensen, A. Martin, D. Meredith, M. Norman, A. Richards, A. Trefethen, M. Viljoen, and D. Wallom. Shibgrid: Shibboleth Access for the National Grid Service. In *Proc. IEEE 2nd Int'l Conf. on e-Science and Grid computing*, 2006.

[19] J. Steiner, C. Neuman, and J. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the USENIX Winter Conference, Dallas, Texas, USA*, pages 191–202, February 1988.

[20] NGS Grid Support. GSI-SSH Terminal. http://www.grid-support.ac.uk/content/view/81/61/.

[21] SWITCHaai. http://www.switch.ch/aai/.

[22] TeraGrid. http://www.teragrid.org/.

[23] S. Tuecke, D. Engert, I. Foster, M. Thompson, L. Pearlman, and C. Kesselman. Internet X.509 public key infrastructure proxy certificate profile. Request for Comments (RFC) 3820, June 2004.

[24] UK e-Science Certification Authority. http://www.grid-support.ac.uk/ca/, Oct 06.

[25] UK Shibboleth Federation. http://www.ukfederation.org.uk/, Oct 06.

[26] Us Higher Education Root pki. http://usher.internet2.edu/.

[27] Usher Subscriber Expected Practices. http://usher.internet2.edu/docs/USHER-Expected-Practices-final.htm, Nov 2006.

[28] G. von Laszewski, J. Gawor, P. Lane, N. Rehn, M. Russell, and K. Jackson. Features of the Java commodity Grid kit. *Concurrency and Computation: Practice and Experience*, 14:1045–1055, 2002.

[29] M. Wahl, T. Howes, and S. Kille. Lightweight directory access protocol (v3). Request for Comments (RFC) 2251, December 1997.

[30] V. Welch, T. Barton, K. Keahey, and F. Siebenlist. Attributes, anonymity, and access: Shibboleth and Globus integration to facilitate Grid collaboration. In *Proceedings of the 4th Annual PKI R&D Workshop*, April 2005.

[31] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for Grid services. In *12th International Symposium on High-Performance Distributed Computing (HPDC-12), Seattle, Washington, USA*, pages 48–57, 2003.

A Certificate-Free Grid Security Infrastructure Supporting Password-Based User Authentication[*]

Jason Crampton, Hoon Wei Lim, Kenneth G. Paterson, and Geraint Price
Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
{jason.crampton, h.lim, kenny.paterson, geraint.price}@rhul.ac.uk

Abstract

Password-based authentication is still the most widely-used authentication mechanism, largely because of the ease with which it can be understood by end users and implemented. In this paper, we propose a security infrastructure for grid applications, in which users are authenticated using passwords. Our infrastructure allows users to perform single sign-on based only on passwords, without requiring a public key infrastructure. Nevertheless, our infrastructure supports essential grid security services, such as mutual authentication and delegation, using public key cryptographic techniques. Moreover, hosting servers in our infrastructure are not required to have public key certificates, meaning mutual authentication and delegation of proxy credentials can be performed in a lightweight and efficient manner.

1 Introduction

The vision of grid computing [17, 19] is to provide easy access to "unlimited" resources, thereby enabling computationally complex tasks to be performed and huge amounts of data to be stored and shared. There has been some suspicion, since the term *grid* was first used about a decade ago, that grid computing might be another technological vision that turns out to be more hype than substance. Despite that, the vision prevails and the gap between vision and reality is narrowing quickly. This is evident from the large and growing number of grid projects and testbeds worldwide [25]. TeraGrid [47], one of the pioneering grid projects, which was completed in 2004, is currently capable of providing 102 teraflops[1] of computing power and more than 15 petabytes[2] of online data storage. Despite the promising signs, many believe that a lot still needs to be done in order to realise computational grids analogous to the pervasive electrical power grid. In particular, as commercial interest grows in grid computing, grid security is an issue that will become increasingly important.

Currently, the grid security infrastructure (GSI) of the Globus Toolkit (GT) [16], proposed by Foster *et al.* [18], plays an essential role in supporting various grid security services, such as single sign-on, mutual authentication and delegation. Being based on public key infrastructure (PKI),[3] GSI users are required to possess and manage long-term credentials (typically RSA public/private key pairs), which are usually renewed yearly. Inevitably, some machines within the scope of a grid community may lack up-to-date protection in the form of the latest vulnerability patches and virus definitions. This may lead to such machines falling under the partial or complete control of attackers who are able to remotely exploit vulnerabilities and hence obtain long-term user credentials. To minimise the risk of compromise, many recent grid implementations make use of the MyProxy system [8, 37] to securely store and protect long-term user credentials. MyProxy also offers the benefit of "credential mobility", enabling users to access their credentials from any machine through, for example, a web browser.

Motivations. It is desirable that remote computing resources be accessible from various platforms, including handheld devices, such as personal digital assistants (PDAs) and mobile phones, as well as desktop and laptop machines. In fact, due to the increasing availability of wireless devices in recent years, *wireless grids* [2, 34, 41] can offer additional untapped resources to existing wired computing re-

[*]The research in this paper was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through Grant EP/D051878/1.

[1]A teraflop is one trillion floating-point operations per second.

[2]1 petabyte = 10^3 terabytes = 10^6 gigabytes.

[3]Here, we assume that existing PKIs make use of certificates. However, we will later show that a PKI can be certificate-free.

sources. In particular, wireless devices can offer different types of resources through their embedded objects, such as cameras, microphones, sensors and global positioning system (GPS) receivers. More importantly, wireless devices can supply information – on temperature, health, pollution levels, etc. – from geographic locations and social settings that are difficult to access through conventional wired networks [34].

The PKI-based GSI is a rather heavyweight apparatus, mainly because of the extensive use of public key certificates [28] and proxy certificates [48]. Generation, certification and verification of public keys, distribution of certificates, and other aspects of traditional public key management using PKIs incur non-trivial overheads. Wireless devices are often battery-powered and the energy required for transmission of a single bit of data is over 1000 times greater than that required by a single 32-bit computation [6]. Therefore, it is necessary to minimise the communication overheads of any grid security infrastructure if we are to exploit the full potential of wireless grids. In short, the emergence of wireless grids has prompted the need for a more lightweight architecture.

Lim and Paterson recently proposed a fully identity-based security infrastructure for grid applications [32] using identity-based public key cryptography (ID-PKC) [14, 44]. Key management in this approach is simpler than in the PKI-based GSI because it does not use certificates and key sizes are relatively small. For instance, the communication bandwidth requirement for mutual authentication and delegation between two entities can be reduced by up to 90%, when appropriately chosen elliptic curves and system parameters are used [32].

Nevertheless, *key revocation* in the identity-based setting can be complicated. Boneh and Franklin [14] proposed the use of a date concatenated with a user's identifier (the construction of a public key from an identifier will be explained in Section 2.1.1) to achieve automated key expiry. However, this approach has the disadvantage of increasing the workload of a Private Key Generator (PKG), since the PKG is required to regularly issue private keys to its users. Alternatively, the PKG could issue private keys less frequently, for example monthly or yearly. In this case, however, it would be necessary to adapt conventional key revocation mechanisms, such as Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP), to the identity-based setting, in order to provide timely revocation of an identifier and its associated public key.

Moreover, *key escrow* is inevitable in the identity-based setting because the PKG uses a master secret to extract private keys of its users. This may not be desirable in some grid applications.

MyProxy continues to play a major role in the GSI by offering better credential protection and mobility to grid users. However, its architecture has a subtle but crucial drawback. In the MyProxy protocol [8], although users are authenticated to their respective MyProxy servers using conventional username/password techniques, server authentication is achieved using the server-authenticated version of the TLS handshake protocol [15]. This implies the need to protect the root Certificate Authority's public key certificates on the users' machines. There are ways for an attacker to install a bogus root key in the user's browser [5, 27]. Hence, if a desktop is vulnerable to stealing of a private key, then the desktop may also be at risk from replacement of the associated Certificate Authority's certificate by the attacker.

The above issues and observations have led us to our investigation of a grid security infrastructure which is not only certificate-free, but also "PKI-free" from the user perspective.

Contributions. In this paper, we propose a password-enabled and certificate-free grid security infrastructure (PECF-GSI). Briefly, our proposal enhances the earlier work of Lim and Paterson [32] so that users are *only* authenticated using passwords, with the authentication taking place between users and a centralised authentication server. This server plays a similar, but not identical, role to the MyProxy server in the PKI-based GSI. Our approach has the benefit that neither client nor server certificates are required during user authentication. Our proposal also completely removes the need for long-term user public keys, and hence the need for a revocation mechanism for these public keys too.[4] Instead, users are given short-lived, identity-based credentials by the authentication server upon successful authentication. All subsequent security services are carried out using these credentials on behalf of users, without requiring direct user intervention. Thus our proposal separates security functions into two "zones": a user-friendly zone, where only passwords are involved, and a certificate-free zone, which is hidden from the users' view, and makes use of full-strength public key techniques. In addition, we show how to solve the key escrow issue by adopting certificateless public key cryptography (CL-PKC) [3]. Our contributions can be summarised as follows:

- We design a lightweight and user-friendly grid security infrastructure. Our proposal inherits attractive properties of the identity-based approach, in particular being certificate-free and using small key sizes. Mutual authentication of a user and a server is based only on a provably secure password-based authentication protocol. Yet, our architecture still supports various grid security services, such as single sign-on, mutual authentication and delegation.

[4]We still require mechanisms for handling revocation of server public keys, however.

- Key revocation in the identity-based setting is a well-known issue [14, 40]. We employ "just-in-time" issuance of short-lived keys to avoid any complications related to revoking users' long-term public keys. Our approach is rather similar to the use of short-lived symmetric keys in Kerberos [36]. This, and the fact that system parameters of the identity-based primitives do not necessarily need to be pre-distributed or bootstrapped, gives rise to easy, flexible and user-friendly deployment of ID-PKC. We also show how timely revocation for hosting servers' long-term public keys can be carried out very simply in our architecture, by pushing up-to-date revocation information to users during authentication.
- We develop an escrow-free grid security infrastructure using CL-PKC. Key escrow is inevitable in the identity-based setting because the PKG extracts private keys on behalf of its users, and is a feature of the identity-based grid security architecture of Lim and Paterson [32]. In applications where high-value or commercially sensitive resources are to be shared, an escrow capability at the Certificate Authority (CA) or MyProxy level is unlikely to be acceptable.
- We devise a more efficient and natural delegation protocol than the current technique used in the GSI [49] and the original approach of Lim and Paterson [32]. This can be achieved by exploiting the properties of hierarchical ID-PKC [24] and CL-PKC [4]. The mathematical properties of hierarchical ID-PKC and CL-PKC allow very efficient credential verification of a delegatee for a particular delegation. A verifier needs *only* to check the credential of the delegatee, instead of having to verify the credentials of the delegatee *and* all of his ancestors along the delegation chain, as in existing proposals [32, 49].

Organisation. The remainder of this paper is organised as follows. In Section 2, we introduce background material relevant to this paper. In Section 3, we present our proposal for a password-enabled and certificate-free grid security infrastructure. This includes a description of the architecture and its underlying protocols. In Section 4, we explain how key escrow can be removed from the security architecture proposed in the previous section. Section 5 discusses some performance issues of our proposal. Finally, we conclude in Section 6.

2 Background

In this section, we first give a brief introduction to pairings, which are bilinear maps fundamental to identity-based public key cryptography (ID-PKC) and certificateless public key cryptography (CL-PKC). We then describe a provably secure password-based authentication protocol due to Abdalla *et al.* [1]. We also give a brief overview of the existing grid security infrastructure (GSI) of the Globus Toolkit (GT), and review some related work.

2.1 Cryptographic Preliminaries

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q for some large prime q, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ denotes a related multiplicative group.

An *admissible pairing* in the context of identity-based and certificateless public key cryptography is a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

- *Bilinear*: Given $P, Q, R \in \mathbb{G}_1$, we have
$$\hat{e}(P, Q+R) = \hat{e}(P,Q) \cdot \hat{e}(P,R) \text{ and}$$
$$\hat{e}(P+Q, R) = \hat{e}(P,R) \cdot \hat{e}(Q,R).$$
Hence, for any $a, b \in \mathbb{Z}_q^*$, we have
$$\begin{aligned}\hat{e}(aP, bQ) &= \hat{e}(abP, Q) = \hat{e}(P, abQ) \\ &= \hat{e}(aP, Q)^b = \hat{e}(P,Q)^{ab}.\end{aligned}$$
- *Non-degenerate*: There exists a $P \in \mathbb{G}_1$ such that $\hat{e}(P,P) \neq 1$.
- *Computable*: If $P, Q \in \mathbb{G}_1$, $\hat{e}(P,Q)$ can be efficiently computed.

Typically, $\mathbb{G}_1$ is a subgroup of the group of points on a suitable elliptic curve over a finite field, $\mathbb{G}_2$ is obtained from a related finite field, and $\hat{e}$ is obtained from the Weil or Tate pairing on the curve. Given $P, Q \in \mathbb{G}_1$ and $a \in \mathbb{Z}_q^*$, $P+Q$ denotes elliptic curve point addition, and aP denotes elliptic curve point (or scalar) multiplication. Note that aP can be computed very efficiently. However, the problem of finding a given aP is believed to be intractable, when the curve is appropriately chosen. This problem is known as the *elliptic curve discrete logarithm* (ECDL) problem. The reader is referred to [21] for more mathematical background on pairings.

2.1.1 Identity-Based Public Key Cryptography

In 1984, Shamir [44] proposed the idea of identity-based public key cryptography (ID-PKC). Instead of generating and using a random public/private key pair in a public key cryptosystem such as RSA or ElGamal, Shamir proposed using a user's name or other unique identifier (such as an email address) as a public key, with the corresponding private component being generated by a trusted Private Key Generator (PKG). Since a user's public key is based on some publicly available information that uniquely identifies the user, an identity-based cryptosystem does not require a mechanism for authenticating public keys. However, Shamir was only able to develop an identity-based signature (IBS) scheme based on the RSA primitive.

Only in the early 2000s did the emergence of cryptographic schemes based on pairings on elliptic curves result in the construction of a feasible and secure IBE scheme [14, 29, 42]. Further details can be found in [39].

Gentry and Silverberg [24] proposed hierarchical identity-based encryption (HIBE) and hierarchical identity-based signature (HIBS) schemes with total collusion resistance, regardless of the number of levels in the hierarchy. In the hierarchical setting, a root PKG produces private keys for PKGs in the next level of the tree, who in turn generate private keys for PKGs or users in the next level (and so on). It is this scheme on which our proposal is based.[5] We now sketch Gentry and Silverberg's HIBE and HIBS schemes (see [24] for full details).

ROOT SETUP: The root PKG chooses a generator $P_0 \in \mathbb{G}_1$, picks a random $s_0 \in \mathbb{Z}_q^*$, and sets $Q_0 = s_0 P_0$. It also selects cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ for some n, $H_3 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_4 : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$ and $H_5 : \{0,1\}^n \rightarrow \{0,1\}^n$. The root PKG's master secret is s_0 and the system parameters are $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$.

LOWER-LEVEL SETUP: A lower-level entity (lower-level PKG or user) at level t picks a random $s_t \in \mathbb{Z}_q^*$ which will be kept secret.

EXTRACT: For an entity at level t with ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, where $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_i \rangle$ is the ID-tuple of the entity's ancestor at level i ($1 \leqslant i \leqslant t-1$), the entity's parent computes $P_t = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_t) \in \mathbb{G}_1$, sets the secret point S_t to be $\sum_{i=1}^{t} s_{i-1} P_i = S_{t-1} + s_{t-1} P_t$ (note that S_{t-1} is the parent's secret point given by the parent's ancestor and s_{t-1} is a secret value only known to the parent), and defines Q-values by setting $Q_i = s_i P_0$ for $1 \leqslant i \leqslant t-1$. The entity at level t is given both S_t, as his private key, and the Q-values by its parent.

ENCRYPT: Given a message m and ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, this algorithm computes the ciphertext $\langle rP_0, rP_2, \ldots, rP_t, z \oplus H_2(\hat{e}(Q_0, P_1)^r), m \oplus H_5(z) \rangle$, where $z \in \{0,1\}^n$ and $r = H_4(z, m)$. Note that $rP_0, rP_2, \ldots, rP_t$ are all elements of $\mathbb{G}_1$ and the sizes of the last two components of the ciphertext are dependent on n.

[5]It is worth noting that other HIBE and HIBS schemes are available. We chose the Gentry/Silverberg schemes because they are efficient and their security is based on reasonable computational assumptions.

DECRYPT: Given a ciphertext $\langle U_0, U_2, \ldots, U_t, V, W \rangle$, this algorithm takes as input the associated private key S_t and recovers m. It also checks if $U_0, U_2, \ldots, U_t$ have the correct structure. Otherwise, the recovered m is rejected.

SIGN: Given a private key S_t and a message $m \in \{0,1\}^*$, the signer with ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$ computes $h = H_3(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, m) \in \mathbb{G}_1$ and $\sigma = S_t + s_t h$. The algorithm outputs the signature $\langle \sigma, Q_1, \ldots, Q_t \rangle$, where each component is in $\mathbb{G}_1$.

VERIFY: Given a signature $\langle \sigma, Q_1, \ldots, Q_t \rangle$ of a message m, this algorithm takes as input the associated public key, computed from ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, and returns a message indicating the success or failure of the verification.

2.1.2 Certificateless Public Key Cryptography

There are applications which do not tolerate key escrow, which is a feature of identity-based cryptosystems. This has led to the development of escrow-free variants of pairing-based public key cryptography, including Al-Riyami and Paterson's certificateless public key cryptography (CL-PKC) [3] and the certificate-based encryption (CBE) concept of Gentry [23]. It is the former approach that we use in this paper.

A user's private key in the certificateless setting consists of two components: (i) an identity-dependent partial private key (generated in the same way as in the normal identity-based approach); and (ii) a full private key which can be produced using the partial private key and some secret known only to the user. Succinctly, this approach uses input from the PKG and the user to generate a private key, thereby eliminating key escrow. We now briefly describe a hierarchical certificateless encryption (HCLE) scheme and a hierarchical certificateless signature (HCLS) scheme [4].

ROOT SETUP: As in Section 2.1.1.

LOWER-LEVEL SETUP: As in Section 2.1.1.

PARTIAL-PRIVATE-KEY EXTRACT: As with the EXTRACT algorithm in Section 2.1.1, except that this algorithm sets the entity's partial private key D_t to be $\sum_{i=1}^{t} s_{i-1} P_i = D_{t-1} + s_{t-1} P_t$, where D_{t-1} is the parent's partial private key and s_{t-1} is a secret value only known to the parent. The entity's parent also defines Q-values by setting $\langle Q_{X_i}, Q_{Y_i} \rangle = \langle s_i P_0, s_i Q_0 \rangle$ for $1 \leqslant i \leqslant t-1$.

SET-PRIVATE-KEY: This algorithm transforms a partial private key D_t of an entity at level t with ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$ into a private key $S_t = s_t D_t$, where s_t is the secret value that the entity has chosen in LOWER-LEVEL SETUP.

SET-PUBLIC-KEY: This algorithm sets a public key of an entity at level t with ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$ as $\langle s_t P_0, s_t Q_0 \rangle$.

ENCRYPT: This algorithm first checks if the Q-values have the correct structure. It then performs similar steps to the ENCRYPT algorithm in Section 2.1.1, except that this algorithm takes as input the ID-tuple $\langle \mathrm{ID}_1, \ldots, \mathrm{ID}_t \rangle$, the public key $\langle s_t P_0, s_t Q_0 \rangle$ and the related Q-values to compute the ciphertext.

DECRYPT: As with the DECRYPT algorithm in Section 2.1.1, except that this algorithm takes as input a different set of Q-values.

SIGN: As with the SIGN algorithm in Section 2.1.1, except that this algorithm uses different Q-values.

VERIFY: This algorithm first validates the format of the Q-values. It then performs the similar steps as with the VERIFY algorithm in Section 2.1.1, except that this algorithm uses a different public key and Q-values.

We remark that the above schemes do not have formal security models and proofs. Nevertheless, they are straightforward adaptations of the provably secure HIBE and HIBS schemes of Gentry and Silverberg described in Section 2.1.1.

2.1.3 A Password-Based TLS Protocol

Abdalla *et al.* [1] recently proposed a provably secure password-based TLS protocol, based on earlier work of Steiner *et al.* [46]. The protocol makes use of a discrete logarithm based mask generation function to instantiate a symmetric encryption primitive, as suggested by Bellare *et al.* [10, 11]. The protocol also makes use of a hash function H, mapping onto a Diffie-Hellman group generated by g. Then, A with password PW_A encrypts a Diffie-Hellman component g^a by calculating $\{g^a\}_{\pi_A}$, where $\pi_A = H(PW_A)$ and $\{g^a\}_{\pi_A} = g^a \cdot \pi_A$. Thus the result of the encryption is a group element. To decrypt and recover g^a, one can simply divide the ciphertext by π_A. We describe a modified version of this protocol and explain how it is used to support single sign-on in Section 3.3.1.

The important point about this protocol is that dictionary attacks are of little value to an adversary. If the adversary guesses a password and uses it to decrypt $g^a \cdot \pi_A$, he simply obtains a group element. In effect, the group element g^a masks the (hash of the) password.

2.2 The Grid Security Infrastructure

The PKI-based GSI focuses on authentication, message protection, and the use of proxy credentials to support single sign-on and credential delegation [18, 49, 50]. In grid applications that employ the GSI, each entity is assigned a unique identity or distinguished name and given a public key certificate signed by a Grid CA. Public key certificates are used to support authentication and key agreement protocols, such as the TLS protocol. Proxy certificates are used for single sign-on and delegation.

Before a user submits a job request, he must create a proxy certificate which includes generating a new public/private key pair and signing the proxy certificate with his long-term private key. This newly created proxy certificate can then be used for repeated authentication with other grid entities. The user's long-term private key does not need to be accessed again until the expiry of the proxy certificate. For rights delegation from a user A to a target service provider X, three steps are required [49]:

1. X generates a new public/private key pair and sends a request (that is signed with the new private key) to A;
2. A verifies the request using the new public key, creates a new proxy certificate, and signs it with her current proxy credential (short-lived private key);
3. A forwards the new proxy certificate to X.

Note that A can impose some constraints on what X can and can't do, using the `ProxyPolicy` field of the proxy certificate. A has to trust that an entity to which X presents this proxy certificate will impose the constraints specified.

The GSI has been built on the Generic Security Service Application Program Interface (GSS-API) [33] and incorporates GSI-enabled OpenSSL [38] to support proxy certificates. Examples of the RSA-based cipher suites include `TLS_RSA_WITH_RC4_128_MD5` and `TLS_RSA_WITH_DES_CBC_SHA`.

In the GSI setting, each user has a long-term RSA public/private key pair with a 1024-bit modulus. The short-term keys for the user's proxy credential have only 512-bit moduli. This substantial reduction of key sizes is driven by the fact that generating an RSA key pair is a computationally expensive operation. It has been shown that generating a key pair with 512-bit moduli can reduce the processing time by approximately 77% of the time required for a 1024-bit key pair [49]. Since the proxy credential has a relatively short lifetime, it is currently believed that the reduction in security implied by using only 512-bit moduli poses an acceptably low risk in grid systems.

There are a small number of grid projects that use Kerberos [36] as the backbone of their security infrastructures. It is generally believed that Kerberos, being based on symmetric key cryptography, is more efficient than PKI-based approaches. However, Kerberos is unlikely to be a suitable long-term solution because many computational grids have a dynamic entity population, and the establishment and management of shared symmetric keys will be impractical. Furthermore, it is not clear how the dynamic delegation mechanism of [49] can be supported using Kerberos. Therefore, PKI is preferred for grid applications, while Kerberos seems to be best suited for intra-domain security. In order to achieve inter-operability with PKI-based systems, some Kerberos-based grid projects make use of a Kerberised client-side program, called KX.509, to acquire X.509 certificates using a client's existing Kerberos ticket [30, 35].

2.3 Related Work

MyProxy [8, 37] is an online credential repository that implements the virtual smart card concept [43]. As with storing keys in a smart card, a MyProxy server is expected to provide better protection for long-term user private keys than desktop computing environments.

To create a proxy credential, a user authenticates himself to the MyProxy server using a password which he shares with the MyProxy server by performing the following steps [8].

1. The user establishes a TCP connection to the server and initiates a server-authenticated TLS handshake protocol.
2. Once the TLS handshake is complete and a secure channel is established, the user sends a request message to the server. The request contains information, such as a username, a password and a lifetime.
3. If all checks succeed, the server will return '0' to indicate success or '1' with an error text that suggests otherwise.
4. The user then generates a new public/private key pair and forwards the public key to the server through the established secure channel.
5. Subsequently, the server creates a new proxy certificate signed with the user's stored private key and returns it to the user.

This approach relies on a certificate-based PKI and the user must ensure that the associated certificates bootstrapped in his machine are trustworthy and have not been replaced.

Recently, Beckles *et al.* [9] considered issues related to the usability of the PKI-based GSI, noting that managing certificates can be burdensome and tedious for general grid users. In an effort to improve the usability of the PKI-based GSI, they adopted Gutmann's plug-and-play PKI concept [26], which emphasises automated and transparent setup of PKI for the end user. In so doing, Beckles *et al.* make use of the PKIBoot service of [26] to allow a user to authenticate himself to a PKIBoot server with the standard username/password method. Subsequently, the user can securely retrieve his public key certificate (and optionally his private key) and/or CAs' certificates. This approach can eliminate the difficult tasks involved in correctly establishing trust roots of CAs from the user side. It can also minimise the user's direct involvement in certificate management. Our proposal for a user-friendly and certificate-free security architecture is influenced by Beckles *et al.*'s work.

Although the plug-and-play PKI concept seems to make PKI more usable for the users, there are still many aspects of PKI that need to be addressed. For example, how can we improve the effectiveness of current key revocation mechanisms, such as CRLs, by exploiting the advantages that the plug-and-play PKI concept could bring? Furthermore, the application of the plug-and-play PKI to the GSI does not reduce the extensive use of certificates, and certificate chain verification is still required for all the grid security services which involve certificates.

3 Our Proposal

Here, we propose a password-enabled and certificate-free GSI (PECF-GSI). We begin by giving a conceptual view of the PECF-GSI design. We explain how PECF-GSI can support various grid security services. Then, we provide details of the protocols which underpin PECF-GSI.

3.1 Architectural Overview

PECF-GSI employs a Trusted Authority (TA), instead of a CA, as the root of trust within a grid environment. The TA's roles include acting as the PKG in the identity-based setting and providing a key management service. In PECF-GSI, a user's long-term credential is simply a password, which he shares with an authentication server. We assume that the user delivers his password to the authentication server during a one-off user registration phase.[6]

The authentication server is assumed to be accredited by the TA and hosting servers (or resource providers) within the grid environment. Unlike the user, who only has to remember a password, the authentication and hosting servers must obtain the TA's authenticated parameter set through out-of-band mechanisms. This hybrid approach divides our architecture into two zones: (i) a user-centric zone which employs password-based authentication, and (ii) a server-centric zone which makes use of identity-based PKI (non-certificate-based PKI). This is illustrated in Figure 1.

As with the current GSI, we make use of proxy credentials when providing security services such as mutual authentication and delegation. In Figure 1, a user proxy is a short-lived agent created by the user to perform security services on the user's behalf. Similarly, a resource proxy is

[6]Note that user registration in PECF-GSI setting may well be much simpler than applying for an X.509 certificate in the GSI setting. This is because the user does not have to obtain a certified public key from a CA.

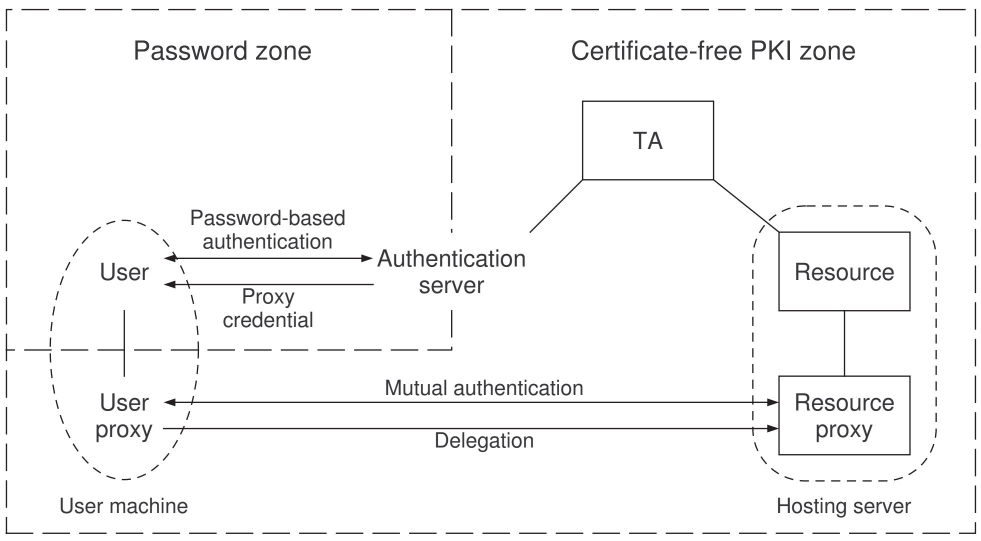


Figure 1. A conceptual view of PECF-GSI.

created by a resource provider to help manage a job submission from a user.

We map the entities in Figure 1, each of which will require some form of credentials to interact with another entity, into the hierarchical setting of the Gentry–Silverberg HIBE and HIBS schemes (as introduced in Section 2.1.1). Let R be an authentication server, A be a user, and X and Y be hosting servers. We write $\bar{A}$ and $\bar{X}$ to denote proxies for A and X, respectively. Then, the TA is a level 0 entity in the hierarchy, and issues private keys to R, X and Y at level 1. These entities, in turn, issue private keys to their respective children at level 2, as shown in Figure 2. Note that A does not possess any long-term credential issued by the TA; instead she obtains proxy credentials from R. Hence, $\bar{A}$, a user proxy for A, becomes a child of R.

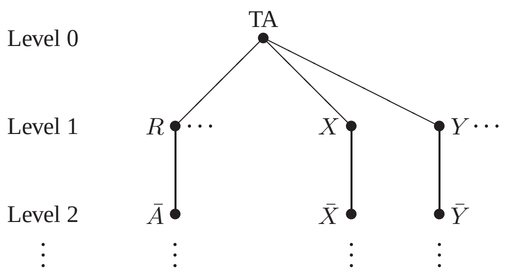


Figure 2. The hierarchical relationships between entities in PECF-GSI.

Using the above notation, we now briefly explain how PECF-GSI can be used to support essential security services for grid applications. Further details of the underlying protocols will be provided in Section 3.3.

Before user A submits a job to resource X, for example, through the Grid Resource Allocation and Management (GRAM) module of the GT [50], she authenticates herself to R using a secure username/password mechanism. When the password-based authentication is successful and a secure channel between A and R is established, R extracts a proxy credential for use by $\bar{A}$. The proxy credential, which comprises a short-term public/private key pair, is transmitted to A, along with other required information, such as the TA's system parameters, through the secure channel.

Subsequently, $\bar{A}$ signs her job request (with the new proxy private key), which is then submitted to X. X verifies $\bar{A}$'s signed request and checks if A is an authorized user. If the checks are successful, X creates $\bar{X}$ and the associated managed job service [50]. This is followed by the mutual authentication of $\bar{A}$ and $\bar{X}$ through an identity-based and certificate-free TLS handshake protocol.

A may, at her discretion, delegate her credential through $\bar{A}$ to $\bar{X}$ for later use [50]. $\bar{A}$ can achieve this in PECF-GSI by simply issuing a new proxy private key to $\bar{X}$ through the established TLS channel. This short-lived private key is generated based on the relevant delegation information. Now the delegatee $\bar{X}$ effectively becomes a child of the delegator $\bar{A}$ in the hierarchy depicted in Figure 2. Similarly, if $\bar{X}$ further delegates some rights to Y, then $\bar{Y}$ will become a child of $\bar{X}$. The resulting delegation chain rooted at the TA is TA $\rightarrow R \rightarrow \bar{A} \rightarrow \bar{X} \rightarrow \bar{Y}$.

In this paper, we use a hierarchy with a single TA for ease of exposition. In actual implementations, we can expand the hierarchy of Figure 2 to support multiple TAs. This can be achieved by adding a root TA at the top of the hierarchy with the TAs becoming level 1 entities. Similarly, lower-level entities are moved down to the next level in the hierarchy.

3.2 On System Parameters and Keys

We now describe the system parameters and keys that will be used in our protocols, which are described in Section 3.3.

3.2.1 Parameter Generation and Distribution

During the system setup phase, the TA runs a Bilinear Diffie-Hellman (BDH) parameter generator to obtain groups $\mathbb{G}_1$, $\mathbb{G}_2$ of large prime order q and an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. It then performs the ROOT SETUP of the Gentry–Silverberg HIBE and HIBS schemes to produce a master secret s_0. The system parameters are $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$. We remark that an authentic set of the TA parameters must be made available to the authentication and hosting servers. One way to achieve this is by bootstrapping these parameters into the grid system. Alternatively, distribution of the parameters is also possible through the use of a certificate obtained from a conventional CA that certifies the parameters. We will discuss concrete parameter choices in Section 5.

3.2.2 Key Generation

Once the system parameters have been set up, the TA can issue private keys to its subordinates at level 1 (see Figure 2) using its master secret s_0 computed by the ROOT SETUP algorithm. For example, authentication server R's long-term private key is $S_R = s_0 P_R$, where $P_R = H_1(\mathrm{ID}_R)$ is the matching public key. Hosting servers' long-term public/private private keys are generated in a similar way. A proxy's public key at level 2 can be computed based on its ancestor's identifier and its own identifier concatenated with a lifetime LT in some fixed format. For example, user A's proxy public key would be $P_{\bar{A}} = H_1(\mathrm{ID}_R, \mathrm{ID}_A \| \mathrm{LT}_A)$, and the corresponding private key can be obtained from R, who will run the EXTRACT algorithm of the Gentry–Silverberg HIBE (or HIBS) scheme to generate $S_{\bar{A}} = S_R + s_R P_{\bar{A}}$. Here, s_R is a secret value chosen by R when it performs the LOWER-LEVEL SETUP algorithm. The upper part of Table 1 summarises the credentials possessed by the authentication server R, user A and hosting server X.

It is worth noting that A does not possess any long-term credential, except a password which she shares with R. In fact, R's proxies are proxies of the users with whom it shares passwords.

3.2.3 Key Revocation

Our proposed design deals with revocation of user keys and of hosting server keys in different ways. The users are never given long-term public keys, instead they are only ever provided with proxy keys. As with proxy certificates [48] and Kerberos tickets [36], these proxy credentials in our PECF-GSI setting have a short lifetime, typically less than 12 hours. As the window of exposure to compromise is minimised, there is no need for an explicit revocation mechanism for user keys. In this case, the hosting servers are trusting R to only distribute fresh keys to the users if the users' privileges are still valid.

Conversely, hosting servers are issued with long term public keys. In this case, there is a requirement for an explicit revocation mechanism. To allow for the revocation of servers' public keys, we introduce the notion of an Identity Revocation List (IRL), where an IRL is analogous to a CRL in a certificate-based environment. The IRL includes the identity of any server whose key has been revoked. This allows users to verify the validity of a particular hosting server's public key prior to submitting a job to that server. IRLs are distributed to users by R via the secure channel established at authentication time. From the user's perspective, this "push" method of distribution simplifies the process of verifying whether a hosting server has had its public key revoked. We discuss this issue in more detail in Section 3.3.1.

3.3 Protocols

We now describe the protocols that we employ in PECF-GSI to provide single sign-on, mutual authentication and delegation for grid applications.

3.3.1 Single Sign-on

The MyProxy system makes use of the existing standard TLS protocol [15] to provide mutual authentication between a user and a MyProxy server. This is typically based on the MyProxy server's public key certificate and a password shared by the server and the user. However, in such a set-up, where the user enters his password only after the secure channel is established, the authentication of the user (through his password) is not directly tied to the secure channel [46]. This may give a false sense of security if management of certificates of the relevant parties (e.g. the server and its CA) are not handled properly. This prompted the study of password-based TLS protocols [1, 46].

We use a modified version of the protocol described in Section 2.1.3 for mutual authentication between a user and the authentication server in our PECF-GSI setting. This is because the protocol is provably secure and it can be im-

Table 1. Credentials and keys in PECF-GSI.

Scheme	Entity	Long-term Credential		Proxy Credential	
		Public Key	Private Key	Public Key	Private Key
Gentry–Silverberg (HIBE/HIBS)	R	$P_R = H_1(\mathrm{ID}_R)$	$S_R = s_0 P_R$	–	–
	A	–	–	$P_{\bar{A}} = H_1(\mathrm{ID}_R, \mathrm{ID}_A \Vert \mathrm{LT}_A)$	$S_{\bar{A}} = S_R + s_R P_{\bar{A}}$
	X	$P_X = H_1(\mathrm{ID}_X)$	$S_X = s_0 P_X$	$P_{\bar{X}} = H_1(\mathrm{ID}_X, \mathrm{ID}_X \Vert \mathrm{LT}_X)$	$S_{\bar{X}} = S_X + s_X P_{\bar{X}}$
Al-Riyami–Paterson (HCLE/HCLS)	R	$\langle s_R P_0, s_R Q_0 \rangle$	$S_R = s_R s_0 P_R$	–	–
	A	–	–	$\langle s_{\bar{A}} P_0, s_{\bar{A}} Q_0 \rangle$	$S_{\bar{A}} = s_{\bar{A}}(S_R + s_R P_{\bar{A}})$
	X	$\langle s_X P_0, s_X Q_0 \rangle$	$S_X = s_X s_0 P_X$	$\langle s_{\bar{X}} P_0, s_{\bar{X}} Q_0 \rangle$	$S_{\bar{X}} = s_{\bar{X}}(S_X + s_X P_{\bar{X}})$

plemented by modifying existing implementations of the widespread standard TLS protocol.

In PECF-GSI, we translate the discrete logarithm based approach of Abdalla *et al.* to the elliptic curve setting. We make use of the hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ from the HIBE scheme. A's password PW_A is mapped to $\pi_A = H_1(PW_A) \in \mathbb{G}_1^*$. The Diffie-Hellman component g^a is replaced by aP_0, where P_0 generates $\mathbb{G}_1$, and $\{aP_0\}_{\pi_A}$ is defined to be $aP_0 + \pi_A$. To recover aP_0, we simply subtract π_A from $\{aP_0\}_{\pi_A}$. Based on this mask generation function, which makes use of the system parameters in our PECF-GSI setting, we can derive a password-based TLS protocol analogous to the protocol of [1].[7] Further details of this protocol are given in Appendix A.

Steiner *et al.* and Abdalla *et al.* suggest that parameters such as $\mathbb{G}_1$, P_0 and H_1 should be fixed or form part of a standardised ciphersuite. This obviates the need for the user A to verify the number-theoretic appropriateness of these parameters.

Once A and R have been mutually authenticated and established a secure session, R extracts a short-lived public/private key pair $(P_{\bar{A}}, S_{\bar{A}})$, shown in Table 1. Subsequently, R sends the following information to A through the secure channel:

1. the newly created proxy credential $(P_{\bar{A}}, S_{\bar{A}})$;
2. an authenticated copy of the TA system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$;[8]
3. an up-to-date IRL.

Upon receiving the proxy credential, A stores the private key $S_{\bar{A}}$ in a local file system accessible by her proxy $\bar{A}$ when necessary. This completes the process of single sign-on by A. The system parameters that A receives from R are needed to run the Gentry–Silverberg HIBE and HIBS schemes when performing mutual authentication and delegation (see Sections 3.3.2 and 3.3.3).

The IRL is used by $\bar{A}$ to check the continuing validity of the identifier of the hosting server to which $\bar{A}$ is going to submit her job. It is worth noting that this approach "forces" the user into receiving an up-to-date IRL. Additionally, the user does not have to check the authenticity of the IRL, assuming R behaves in an honest manner. Upon expiry of the proxy credential, all the information that $\bar{A}$ obtained from R can be destroyed.

We remark that the whole process of single sign-on does not involve any kind of certificate or parameter verification from the user's perspective (recall that users are in a password-based zone in our PECF-GSI setting). Here, we regard verification of parameters as checking the authenticity of the parameters and not validating their number-theoretic structure.

3.3.2 Mutual Authentication and Key Agreement

While a password-based TLS protocol is a convenient mechanism for the authentication server and its (known) users, the standard PKI-based TLS protocol is clearly more suitable for mutual authentication and key agreement between two entities who have not previously communicated. We now leave the password-based zone and enter the certificate-free PKI zone, where we explain how two entities within a grid environment can authenticate each other and share a session key.

Figure 3 shows a certificate-free authenticated key agreement protocol, adapted from Lim and Paterson's identity-based TLS protocol [32], with minor modifications to the `ServerIdentifier` and `ClientIdentifier` messages. The protocol employs the Gentry–Silverberg HIBE and HIBS schemes.

In the first message of the protocol, $n_{\bar{A}}$ denotes a nonce chosen by $\bar{A}$, `session_id` is self-explanatory, and `cipher_suite` contains a cipher specification that handles the HIBE and HIBS schemes, e.g. `TLS_HIBE_HIBS_WITH_DES_CBC_SHA`. Here, $Enc_{\bar{X}}(.)$ denotes an encryption using the HIBE scheme

[7] In order to optimise the efficiency of our proposal, we re-use some of the components of the system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2, H_3, H_4, H_5 \rangle$ that R obtained from the TA.

[8] We note that the parameters $\mathbb{G}_1$, P_0 and H_1 are already in use by A to run the password-based TLS protocol. Hence, in an actual implementation, R only needs to transmit the remaining system parameters to A.

(1)	$\bar{A} \rightarrow \bar{X}:$	`ClientHello` = $n_{\bar{A}}$, `session_id, cipher_suite`
(2)	$\bar{X} \rightarrow \bar{A}:$	`ServerHello` = $n_{\bar{X}}$, `session_id, cipher_suite`,
		`ServerIdentifier` = ID_X, $\mathrm{ID}_X \| \mathrm{LT}_X$,
		`ServerHelloDone`
(3)	$\bar{A} \rightarrow \bar{X}:$	`ClientIdentifier` = ID_R, $\mathrm{ID}_A \| \mathrm{LT}_A$,
		`ClientKeyExchange` = $Enc_{\bar{X}}$(`pre_master_secret`),
		`IdentityVerify` = $Sig_{\bar{A}}$(`handshake_messages`),
		`ClientFinished`
(4)	$\bar{X} \rightarrow \bar{A}:$	`ServerFinished`

Figure 3. A certificate-free authenticated key agreement protocol

with X's proxy public key $P_{\bar{X}}$, while $Sig_{\bar{A}}(.)$ represents a signing operation in the HIBS scheme using A's proxy private key $S_{\bar{A}}$.

When $\bar{A}$ (playing the role of client) performs mutual authentication with $\bar{X}$ (playing the role of server), she needs to forward her public key information, i.e. $\mathrm{ID}_R, \mathrm{ID}_A \| \mathrm{LT}_A$ to $\bar{X}$ as part of the protocol handshake, and vice versa. Note that since $\bar{X}$ includes its long-term identifier in the `ServerIdentifier` message, $\bar{A}$ must check if the identifier is still valid using the IRL that she received from R. Similarly, since $\bar{A}$ uses R's long-term public key information, $\bar{X}$ must validate R's public key before using it.

Space constraints preclude a more detailed description of the TLS protocol and the above protocol. The interested reader is referred to the literature for further details [15, 32]. We also note that our certificate-free authenticated key agreement protocol can be adapted straightforwardly to support user-to-user authentication.

3.3.3 Delegation

The current delegation technique used in the GSI requires a round-trip interaction between a delegator (typically a grid user) and a delegatee (typically a hosting server or resource provider), as described in Section 2.2. Also, verification of a delegatee's credential requires validating both long-term and proxy certificates of all the parties involved along the delegation chain.

Lim and Paterson proposed an identity-based one-pass delegation protocol [32], in which the delegator signs a delegation token and forwards it to the delegatee. One advantage of this approach is that the delegator can bind the delegatee's public key information to the delegation token without acquiring the delegatee's proxy public key, thus requiring only one-pass protocol message. However, verification of a delegatee's status as the delegation target requires validation of all the signed delegation tokens (analogous to validation of certificates in the GSI) issued by all the delegators along the delegation chain.

Here, we propose a delegation protocol which is not only certificate-free and is a one-pass protocol message, but also has a very efficient verification mechanism in the sense that a delegatee's delegated credential can be checked by performing only one signature verification, regardless of the length of the delegation chain. This is a significant improvement on the two aforementioned delegation methods.

We now explain the details of the delegation technique that we employ in PECF-GSI by using an example between the delegator A (through her proxy $\bar{A}$) and the delegatee $\bar{X}$. In the delegation process, $\bar{A}$ performs the following steps:

1. compute a proxy public key $P_{\bar{A}/\bar{X}}$ of the form
$$H_1(\mathrm{ID}_R, \mathrm{ID}_A \| \mathrm{LT}_A, \mathrm{ID}_X \| \mathrm{LT}_{\bar{X}} \| \mathrm{Job}_{\bar{X}} \| \mathrm{Policy}_{\bar{X}}),$$
where $\mathrm{LT}_{\bar{X}}$ is the lifetime that $\bar{A}$ decides for $\bar{X}$, $\mathrm{Job}_{\bar{X}}$ describes A's job request, and $\mathrm{Policy}_{\bar{X}}$ indicates the policy that A wishes to enforce on $\bar{X}$;
2. extract a proxy private key $S_{\bar{A}/\bar{X}} = S_{\bar{A}} + s_{\bar{A}} P_{\bar{A}/\bar{X}}$ with her secret value $s_{\bar{A}}$;
3. transmit $\langle \mathrm{ID}_X \| \mathrm{LT}_{\bar{X}} \| \mathrm{Job}_{\bar{X}} \| \mathrm{Policy}_{\bar{X}}, S_{\bar{A}/\bar{X}} \rangle$ to $\bar{X}$ through a secrecy and integrity protected TLS channel.[9]

In this case, $\bar{A}$ actually acts as a PKG and issues a private key to $\bar{X}$, which becomes the entity below $\bar{A}$ at level 3 in Figure 2. This can be seen in the ID-tuple used to construct $P_{\bar{A}/\bar{X}}$, in which the first two parts are identifiers of $\bar{X}$'s ancestors, i.e. R and $\bar{A}$.

It is worth noting that here, $\bar{X}$'s delegated proxy credential is different from $\bar{X}$'s proxy credential shown in Table 1. The latter is usually used when $\bar{X}$ performs mutual authentication with users.

If a third party, for example $\bar{Y}$, wants to verify that $\bar{X}$ indeed is acting on $\bar{A}$'s behalf, then $\bar{Y}$ must: (i) authenticate

[9]It might be thought that the need for the secure channel to transport the proxy private key from $\bar{A}$ to $\bar{X}$ is a limitation of this approach. In fact, the secure channel between these two parties will exist anyway; the parties have to authenticate each other using the TLS handshake protocol, before the delegation can take place. This is to ensure that the delegation is targeted at the right entity and that the delegation target is convinced of the identity of the delegator.

$\bar{X}$ and (ii) check that $\bar{X}$ is in possession of $S_{\bar{A}/\bar{X}}$. These two checks will be carried out as part of the TLS handshake that takes place between $\bar{X}$ and $\bar{Y}$.

When $\bar{X}$ further delegates $\bar{A}$'s credential to another hosting server Y, $\bar{X}$ can construct a new proxy public key

$$P_{\bar{A}/\bar{X}/\bar{Y}} = H_1(\mathrm{ID}_R, \mathrm{ID}_A\|\mathrm{LT}_A, \mathrm{ID}_X\|\mathrm{LT}_{\bar{X}}\|\mathrm{Job}_{\bar{X}}\|\mathrm{Policy}_{\bar{X}}, \mathrm{ID}_Y\|\mathrm{LT}_{\bar{Y}}\|\mathrm{Job}_{\bar{Y}}\|\mathrm{Policy}_{\bar{Y}}),$$

where $\mathrm{Job}_{\bar{Y}}$ refers to the job (potentially sub-tasks of $\mathrm{Job}_{\bar{X}}$) that $\bar{X}$ wants $\bar{Y}$ to execute and $\mathrm{Policy}_{\bar{Y}}$ refers to the policy that $\bar{X}$ imposes on $\bar{Y}$, respectively. The matching private key is $S_{\bar{A}/\bar{X}/\bar{Y}} = S_{\bar{A}/\bar{X}} + s_{\bar{X}} P_{\bar{A}/\bar{X}/\bar{Y}}$. This private key and the relevant information can then be forwarded to $\bar{Y}$, which subsequently becomes subordinate to $\bar{X}$ at level 4 of the hierarchy.

To verify $\bar{Y}$'s delegated proxy credential, the verifier only needs to authenticate $\bar{Y}$ and check whether $\bar{Y}$ knows the private key corresponding to $P_{\bar{A}/\bar{X}/\bar{Y}}$, even though the delegation chain now has two delegatees ($\bar{X}$ and $\bar{Y}$). This can be done, in principle, by verifying a signature produced by $\bar{Y}$ using $S_{\bar{A}/\bar{X}/\bar{Y}}$.

We remark that the use of the Gentry–Silverberg HIBS scheme for this purpose would result in the size of the signature increasing as the delegation chain grows and verification of the signature becoming slower. Nevertheless, there exist improved HIBE schemes in the literature, from which we can derive a more efficient HIBS scheme. Boneh *et al.*, for example, recently proposed a HIBE scheme in which the size of the ciphertext is constant and decryption requires only two pairing computations, regardless of the hierarchy depth [13].

3.4 Security Considerations

In our single sign-on approach, we assume that R is a party trusted to issue the correct system parameters, most importantly Q_0, and up-to-date IRLs to its users through secure channels. Therefore, no additional infrastructure is required to verify the authenticity of the parameters and IRLs. Note that most of the components of the system parameters discussed in Section 3.2 can be fixed and made public, except $Q_0 = s_0 P_0$, where s_0 is the TA's master secret. A failure to obtain Q_0 from a trusted source would allow a trivial man-in-the-middle attack. Our single sign-on protocol is secure against such an attack, assuming R behaves honestly. Also, we assume that hosting servers always trust R in issuing proxy credentials to the correct users. These assumptions are essential for the protocol in Figure 3 and our delegation protocol to work as intended.

We now consider the possibility of an adversary attacking the delegation protocol. Using our example from the previous section, we need to consider the possibility that an adversary E, who is also a valid user under the same TA, intercepts the proxy private key $S_{\bar{A}/\bar{X}}$ that $\bar{A}$ created for $\bar{X}$, and replaces it with E's self-computed private key $S'_{\bar{A}/\bar{X}} = S_{\bar{E}} + s_{\bar{E}} P_{\bar{A}/\bar{X}}$. Superficially, this appears to be a feasible attack, since $P_{\bar{A}/\bar{X}}$ is public and E knows the system parameters. However, the HIBE and HIBS schemes have the property that the private key corresponding to $P_{\bar{A}/\bar{X}}$ can only be computed by the owner of the identity "$\mathrm{ID}_A\|\mathrm{LT}_A$", which is the immediate ancestor of the next level identity "$\mathrm{ID}_X\|\mathrm{LT}_{\bar{X}}\|\mathrm{Job}_{\bar{X}}\|\mathrm{Policy}_{\bar{X}}$" in the same hierarchy. Gentry and Silverberg's security model does model an adversary that obtains identifiers to which it is not entitled, and their HIBE and HIBS schemes are provably secure in such an attack model [24].

4 Removing Key Escrow

Key escrow is a feature of the HIBE and HIBS schemes used in PECF-GSI, as with other standard identity-based cryptographic schemes. This may not be acceptable for certain grid applications. One way of solving the key escrow problem is to apply the Al-Riyami–Paterson HCLE and HCLS schemes to PECF-GSI.

In order to use the Al-Riyami–Paterson HCLE and HCLS schemes (see Section 2.1.2), we need to modify the keys used in the HIBE/HIBS schemes. An entity's private key in the HCLE and HCLS schemes is the product of the entity's private key and its chosen secret value for the Gentry–Silverberg HIBE and HIBS schemes. For instance, R's long-term private key in the HIBE/HIBS schemes is $s_0 P_R$; hence R's new private key for the HCLE/HCLS schemes would be $s_R s_0 P_R$, where s_R is a secret value that R uses to extract private keys for its immediate lower-level entities. The public key of R, which comprises two components, is now $\langle s_R P_0, s_R Q_0 \rangle$. The lower section of Table 1 summarises the new key sets required for the Al-Riyami–Paterson HCLE and HCLS schemes.

4.1 Single Sign-on Without Key Escrow

The changes that we have to make to PECF-GSI in order to remove the key escrow issue are rather trivial. When A performs a single sign-on, she and her authentication server R first perform mutual authentication using a shared password, and then establish a secure channel (as before). Subsequently, R runs the PARTIAL-PRIVATE-KEY EXTRACT algorithm and issues a partial private key $(S_R + s_R P_{\bar{A}})$ to A, along with other information such as the system parameters and an updated IRL.

When A receives the partial private key, she runs the LOWER-LEVEL SETUP algorithm to randomly pick a secret value $s_{\bar{A}}$. This secret value, in turn, is used to compute her proxy public/private key pair. $\bar{A}$ can then use the proxy credential to perform mutual authentication and delegation with a hosting server. Since the value $s_{\bar{A}}$ is unknown to R, A's new proxy private key is kept secret from R, which is not the case in the protocols described in Section 3.3.

4.2 Security Concerns

The cryptographic set-up in CL-PKC allows users to create more than one public key for the same partial private key [3]. For example, A can randomly select two different secret values $s_{\bar{A}}$ and $s'_{\bar{A}}$, and compute two sets of distinct public/private key pairs. However, we believe that this property would not cause any major issues within a grid environment. This is because partial private keys produced by the authentication server are short-lived. In fact, the users can take advantage of this property by extracting different proxy key pairs for different job submissions to increase key freshness, before the expiry of their respective partial private keys.

In the context of CL-PKC, we must trust the authentication server not to mount active impersonation attacks against its users. Such attacks are possible because the authentication server can always select a secret value (for some "victim") and calculate a private key based on the victim's partial private key to which it necessarily has access. However, these attacks would leave behind cryptographic evidence which may reveal the authentication server's actions. We also note that traditional PKIs have an analogous problem: we have to trust a CA not to illegally sign user certificates, enabling the CA to impersonate these users to other parties.

5 Performance

In this section, we compare the communication costs of the protocols used in GSI and PECF-GSI for key agreement and delegation. We then compare the computational costs of long-term and proxy key generation, key agreement and delegation.

In the GSI, we assume the size of a 1024-bit RSA public key certificate is 1.5 kilobytes (ignoring small fields, such as subject and validity period). Similarly, a 512-bit RSA proxy certificate is 0.8 kilobytes.[10] Ciphertexts and signatures generated using a short-term RSA key are 512 bits.

For PECF-GSI, we work with a supersingular elliptic curve of embedding degree 4 over $\mathbb{F}_{2^{271}}$ [20, 22] to obtain the system parameters described in Section 3.2.[11] This choice results in a corresponding group of prime order q approximately equal to 2^{252}, and gives roughly the same security level as 1024-bit RSA. Using the point compression technique, elements of this group can be represented using 272 bits. Since all arithmetic is carried out in fields of characteristic 2, group operations and pairing computations can be implemented efficiently [7].

In addition to the curve and group selections, we require hash functions for the Gentry–Silverberg HIBE and HIBS schemes. The outputs of H_1 and H_3 are elements of $\mathbb{G}_1$, while H_4 gives an output with approximately 252 bits. Note that the size of outputs of H_2 and H_5 are dependent on n, the bit length of plaintexts. We assume that $n = 256$, since this is sufficient for our protocol messages (see Section 3.3.2). Hence, the size of ciphertexts and signatures produced by the Gentry–Silverberg HIBE and HIBS schemes (or the Al-Riyami–Paterson HCLE and HCLS schemes) can be computed, and are 1056 bits and 816 bits, respectively.

The estimated communication costs for the protocols that underpin the GSI and PECF-GSI are summarised in Table 2. The architecture based on the Gentry–Silverberg schemes is denoted by PECF-GSI-I, while the architecture based on the Al-Riyami–Paterson schemes is denoted by PECF-GSI-II. Actual computational costs in milliseconds are also summarized in this table. These timings were obtained by implementing the key generation algorithms in RSA and the Gentry–Silverberg HIBE/HIBS schemes using the MIRACL library [45]. The experiments were performed on a Pentium IV 2.4 GHz processor. For simplicity, we limit the length of the delegation chain to one. Computational costs are not currently available for PECF-GSI-II.

5.1 Communication Costs

In the GSI, the communication cost of the key agreement protocol through the standard TLS handshake is approximately 37.8 kilobits; the corresponding cost in PECF-GSI-I (with key escrow) is approximately 1.9 kilobits. Note that for simplicity, we ignore small components in both the TLS protocols, such as the `ClientHello` and `ClientFinished` messages. Key agreement in PECF-GSI-II, which does not have key escrow and so makes use of additional public key components, has a slightly higher communication cost compared to key agreement in PECF-GSI-I.

The communication costs for delegation in the GSI can be estimated straightforwardly from the protocol described in Section 2.2. Delegation in PECF-GSI-I is very lightweight because it only involves issuance of a private key. In PECF-GSI-II, additional public key components are included as well, and hence extra bandwidth is required. It is obvious that our certificate-free approach suits wire-

[10] It is worth mentioning that RSA keys can be replaced by much shorter keys, which are based on elliptic curve cryptography (ECC) [12]. However, this does not eliminate the fact that certificates will still be in use, and hence, the associated limitations of certificated-based architectures.

[11] We note that this curve is only chosen so that concrete timings and bit counts can be given. A wide variety of other choice of curves and their associated parameters are available.

Table 2. A comparison of performance characteristics.

Type of Cost (units)	Operation	GSI	PECF-GSI-I	PECF-GSI-II
Communication (KB)	**Key agreement**	37.8	1.9	2.4
	Delegation	7.8	0.3	0.8
Computation time (ms)	**Long-term key generation**	149.90	1.69	
	Proxy key generation	34.85	1.74	
	Key agreement	5.34	28.95	
	Delegation	38.33	10.16	

less environments well, in which transmission of data using battery-powered mobile devices is a relatively expensive operation.

5.2 Computational Costs

We can see from Table 2 that key generation in PECF-GSI is significantly more efficient than RSA key generation in the GSI.[12] This is, to some extent, an unfair comparison, because of the completely different mathematical properties behind these two approaches, but it does suggest that the single sign-on protocol in our proposal is less computationally expensive than in the GSI incorporating MyProxy.

The figures for key agreement (including mutual authentication) are obtained by summing the times taken for the user and authentication server to perform their respective parts of the protocol. A similar method is used to obtain a single figure for the computational costs of delegation [31, Table 4.3]. Key agreement in the GSI (using the standard TLS protocol) is computationally less expensive than the corresponding operations in PECF-GSI (using the modified identity-based TLS protocol). In contrast, delegation in PECF-GSI is almost four times faster than in the GSI.

It is unfortunate that key agreement is slower in PECF-GSI, but we note that the cumulative time for key agreement and delegation is still lower in PECF-GSI. Overall, it can be seen that the computational costs of PECF-GSI are comparable to those of the GSI. Our approach allows a different trade-off to be struck between the computational costs at the user side and at the server side.

6 Conclusions and Future Work

We have proposed a grid security infrastructure which is password-enabled and certificate-free. Our infrastructure offers three distinct advantages.

- The only long-term secret required by users is a password. This is likely to improve usability and accessibility of grid applications considerably. Moreover, we do not need to worry about revocation of users' public keys, a considerable problem in certificate-based architectures.
- Key agreement and delegation in our approach requires much less bandwidth than the GSI. In addition, our delegation technique requires only a single verification.
- The computational effort required by the key generation algorithms that we employ is considerably lower than in the GSI.

Our lightweight security architecture is more suitable than the GSI for use by devices with limited resources, thereby significantly extending the number of devices that can interact with computational grids and going some way to realising the potential of wireless grids. That said, identity-based and certificateless public key cryptography are relatively new, and thus lack support from standardisation bodies. This may hinder early adoption of our proposal.

To meet the requirement of grid applications which do not tolerate key escrow, we proposed the use of certificateless public key cryptography, which enables users to select their own private components. This only resulted in minor changes, in terms of key set-up, to our original architecture.

An important security aspect of grid applications which we have not considered in this paper is authorization. A natural extension of this work is to develop novel authorization techniques using properties of identity-based cryptography.

We are also aware that there are other aspects of performance that ought to be considered, such as fault tolerance and availability of our architecture in comparison with MyProxy. Since MyProxy still continues to evolve, it will be appropriate to make such comparisons in the near future.

Acknowledgements The authors would like to thank Jim Basney for helpful discussions and the anonymous referees for useful comments.

[12] Note that we only compare private key extraction in PECF-GSI to RSA public/private key pair generation in the GSI, because the time taken to compute a public key using the hash function H_1 in PECF-GSI is negligible given the parameters we have chosen. Furthermore, construction of a public key by hashing an identifier occurs as part of the associated encryption/decryption scheme.

References

[1] M. Abdalla, E. Bresson, O. Chevassut, B. Möller, and D. Pointcheval. Provably secure password-based authentication in TLS. In *Proceedings of the 1st ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS 2006)*, pages 35–45. ACM Press, March 2006.

[2] S.P. Ahuja and J.R. Myers. A survey on wireless grid computing. *The Journal of Supercomputing*, 37(1):3–21, 2006.

[3] S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. In C.S. Laih, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2003*, pages 452–473. Springer-Verlag LNCS 2894, November 2003.

[4] S.S. Al-Riyami and K.G. Paterson. *Certificateless Public Key Cryptography*. Cryptology ePrint Archive, Report 2003/126, October 2003. Available at http://eprint.iacr.org/2003/126.

[5] A. Alsaid and C.J. Mitchell. Installing fake root keys in a PC. In D. Chadwick and G. Zhao, editors, *Proceedings of the 2nd European Public Key Infrastructure Workshop (EuroPKI 2005)*, pages 227–239. Springer-Verlag LNCS 3545, 2005.

[6] K. Barr and K. Asanović. Energy aware lossless data compression. *ACM Transactions on Computer Systems*, 24(3):250–291, August 2006.

[7] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *Advances in Cryptology - Proceedings of CRYPTO 2002*, pages 354–368. Springer-Verlag LNCS 2442, 2002.

[8] J. Basney, M. Humphrey, and V. Welch. The MyProxy online credential repository. *Journal of Software: Practice and Experience*, 35(9):817–826, July 2005.

[9] B. Beckles, V. Welch, and J. Basney. Mechanisms for increasing the usability of grid security. *International Journal of Human-Computer Studies*, 63(1-2):74–101, July 2005.

[10] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2000*, pages 139–155. Springer-Verlag LNCS 1807, 2000.

[11] M. Bellare and P. Rogaway. *The AuthA Protocol for Password-Based Authenticated Key Exchange*. Contribution to IEEE P1363, March 2000.

[12] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Möller. Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS). *The Internet Engineering Task Force (IETF)*, RFC 4492, May 2006.

[13] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2005*, pages 440–456. Springer-Verlag LNCS 3494, 2005.

[14] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology - Proceedings of CRYPTO 2001*, pages 213–229. Springer-Verlag LNCS 2139, August 2001.

[15] T. Dierks and C. Allen. The TLS protocol version 1.0. *The Internet Engineering Task Force (IETF)*, RFC 2246, January 1999.

[16] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputing Applications*, 11(2):115–128, 1997.

[17] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Elsevier, San Francisco, 2004.

[18] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational Grids. In *Proceedings of the 5th ACM Computer and Communications Security Conference (CCS '98)*, pages 83–92. ACM Press, November 1998.

[19] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[20] S.D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2001*, pages 495–513. Springer-Verlag LNCS 2248, 2001.

[21] S.D. Galbraith. Pairings. In I.F. Blake, G. Seroussi, and N.P. Smart, editors, *Chapter 9 of Advances in Elliptic Curve Cryptography*, pages 183–213, Cambridge, 2005. Cambridge University Press, LMS 317.

[22] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D.R. Kohel, editors, *Proceedings of the 5th International Symposium on Algorithmic Number Theory (ANTS-V)*, pages 324–337. Springer-Verlag LNCS 2369, 2002.

[23] C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *Advances in Cryptology - Proceedings of EUROCRYPT 2003*, pages 272–293. Springer-Verlag LNCS 2656, May 2003.

[24] C. Gentry and A. Silverberg. Hierarchical ID-Based cryptography. In Y. Zheng, editor, *Advances in Cryptology - Proceedings of ASIACRYPT 2002*, pages 548–566. Springer-Verlag LNCS 2501, December 2002.

[25] GridCafé. *Grid Projects in the World*. Available at http://gridcafe.web.cern.ch/, last accessed in January 2007.

[26] P. Gutmann. Plug-and-play PKI: A PKI your mother can use. In *Proceedings of 12th USENIX Security Symposium*, pages 45–58, 2003.

[27] J.M. Hayes. The problem with multiple roots in web browsers - certificate masquerading. In *Proceedings of the IEEE 7th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 306–311. IEEE Computer Society Press, 1998.

[28] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *The Internet Engineering Task Force (IETF)*, RFC 3280, April 2002.

[29] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of 4th Algorithmic Number Theory Symposium (ANTS-IV)*, pages 385–394. Springer-Verlag LNCS 1838, 2000.

[30] O. Kornievskaia, P. Honeyman, B. Doster, and K. Coffman. Kerberized credential translation: A solution to web access control. In *Proceedings of 10th USENIX Security Symposium*, pages 235–250, August 2001.

[31] H.W. Lim. *On the Application of Identity-Based Cryptography in Grid Security*. Ph.D thesis, University of London, 2006.

[32] H.W. Lim and K.G. Paterson. Identity-based cryptography for grid security. In H. Stockinger, R. Buyya, and R. Perrott, editors, *Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing (e-Science 2005)*, pages 395–404. IEEE Computer Society Press, 2005.

[33] J. Linn. Generic security service application program interface version 2, update1. *The Internet Engineering Task Force (IETF)*, RFC 2743, January 2000.

[34] L.W. McKnight, J. Howison, and S. Bradner. Wireless grids: Distributed resource sharing by mobile, nomadic, and fixed devices. *IEEE Internet Computing*, 8(4):24–31, July/August 2004.

[35] P.C. Moore, W.R. Johnson, and R.J. Detry. Adapting Globus and Kerberos for a secure ASCI Grid. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (SC2001), CD-ROM*, page 21. ACM Press, November 2001.

[36] B.C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, September 1994.

[37] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the Grid: MyProxy. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 2001)*, pages 104 –111. IEEE Computer Society Press, August 2001.

[38] The OpenSSL Project. *OpenSSL: The Open Source Toolkit for SSL/TLS*, 2006. Available at http://www.openssl.org/, last accessed in September 2006.

[39] K.G. Paterson. Cryptography from pairings. In I.F. Blake, G. Seroussi, and N.P. Smart, editors, *Chapter 10 of Advances in Elliptic Curve Cryptography*, pages 215–251, Cambridge, 2005. Cambridge University Press, LMS 317.

[40] K.G. Paterson and G. Price. A comparison between traditional public key infrastructures and identity-based cryptography. *Information Security Technical Report*, 8(3):57–72, 2003.

[41] T. Phan, L. Huang, and C. Dulan. Challenge: Integrating mobile wireless devices into the computational grid. In *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MOBICOM 2002)*, pages 271–278. ACM Press, 2002.

[42] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the 2000 Symposium on Cryptography and Information Security (SCIS 2000)*, January 2000.

[43] R.S. Sandhu, M. Bellare, and R. Ganesan. Password-enabled PKI: Virtual smartcards versus virtual soft tokens. In *Proceedings of the 1st Annual PKI R&D Workshop*, pages 89–96, 2002.

[44] A. Shamir. Identity-based cryptosystems and signature schemes. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology - Proceedings of CRYPTO '84*, pages 47–53. Springer-Verlag LNCS 196, August 1985.

[45] Shamus Software Ltd. *MIRACL*. Available at http://indigo.ie/∼mscott/, last accessed in January 2007.

[46] M. Steiner, P. Buhler, T. Eirich, and M. Waidner. Secure password-based cipher suite for TLS. *ACM Transactions on Information and System Security*, 4(2):134–157, May 2001.

[47] The TeraGrid Project. *TeraGrid*. Available at http://www.teragrid.org/, last accessed in January 2007.

[48] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M.R. Thompson. Internet X.509 public key infrastructure proxy certificate profile. *The Internet Engineering Task Force (IETF)*, RFC 3820, June 2004.

[49] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 proxy certificates for dynamic delegation. In *Proceedings of the 3rd Annual PKI R&D Workshop*, pages 42–58, April 2004.

[50] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke. Security for Grid services. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12 2003)*, pages 48–61. IEEE Computer Society Press, June 2003.

A A Password-Based TLS Protocol

Figure 4 shows the EC-SOKE-TLS protocol, an adaptation of the Simple Open Key Exchange for TLS (SOKE-TLS) [1] to the elliptic curve setting.

(1)	$A \to R:$	`ClientHello` = n_A, `session_id`, `cipher_suite`
(2)	$R \to A:$	`ServerHello` = n_R, `session_id`, `cipher_suite`, `ServerKeyExchange` = ID_R, rP_0, `ServerHelloDone`
(3)	$A \to R:$	`ClientKeyExchange` = ID_A, $\{aP_0\}_{\pi_A}$
(4)	$R \to A:$	`ServerFinished`
(5)	$A \to R:$	`ClientFinished`

Figure 4. EC-SOKE-TLS protocol

We assume user A (playing the role of client) and authentication server R (playing the role of server) share a password π_A and some parameters required for the protocol, such as $\mathbb{G}_1, P_0$ and H_1. We use $\{\cdot\}_\pi$ to denote a mask generation function which maps an element of $\mathbb{G}_1$ into another element of $\mathbb{G}_1$ by using the password π. In our case, the mask generation function is defined to be the addition of a group element and a password (as described in Section 3.3.1).

In step (1), A sends R a standard `ClientHello` message as in the standard TLS protocol. Here, n_A is a random number generated by A.

In step (2), R responds with a `ServerHello` message which contains a different random number n_R and other associated information. R also randomly picks $r \in \mathbb{Z}_q^*$, calculates its Diffie-Hellman value as rP_0 and forwards the `ServerKeyExchange` message to A. The `ServerHelloDone` message is sent to indicate the end of step (2).

In step (3), A randomly selects $a \in \mathbb{Z}_q^*$ and computes aP_0. This Diffie-Hellman value is then encrypted (or masked) using A's password π_A and forwarded to R, along with her identity.

A and R compute a pre-master secret

$$pms = H_0(\mathrm{ID}_A, \mathrm{ID}_R, \pi_A, \{aP_0\}_{\pi_A}, yP_0, ayP_0),$$

where H_0 is a secure hash function; the associated master secret is

$$ms = \mathrm{PRF}(pms, \text{“}\texttt{master secret}\text{”}, n_A, n_X),$$

where PRF is a pseudo-random function specified for the standard TLS protocol [15].

Note that R can compute pms if and only if it can recover aP from A and compute the composite Diffie-Hellman value ayP_0. On the other hand, A must know the value a of aP that R would recover from her password π_A. This is to prevent an adversary from impersonating A to R using a guessed password π'_A by computing $\{a'P_0\}_{\pi'_A}$, where a' is a value which is known to the adversary. This impersonation would fail unless the adversary could predict the value a'' of $a''P_0$ that R recovers using the correct password π_A from $\{a'P_0\}_{\pi'_A}$. The difficulty of finding a'' is believed to be as hard as solving the ECDL problem.

In step (4), R produces the `ServerFinished` message, which contains the verification value:

$$\mathrm{PRF}(ms, \text{“}\texttt{server finished}\text{”}, h_1, h_2)$$

where h_1 and h_2 represent hash values of all handshake messages up to but not including this message, using different hash functions.

Note that A must verify if R has computed the correct value in the `ServerFinished` message before calculating her corresponding verification value in the `ClientFinished` message in the last step. This is to prevent a bogus server from impersonating R to A and mounting an offline dictionary attack. If A sends the `ClientFinished` message immediately after sending the `ClientKeyExchange` message, the bogus server R' can test if a candidate password π'_A is correct by performing the following steps:

1. decrypt $\{aP_0\}_{\pi_A}$ from A using its guessed password π'_A and obtain $a'P$;
2. compute the corresponding pre-master secret pms' and master secret ms' using $\{aP_0\}_{\pi_A}, yP_0, a'yP$, assuming R' knows the value y;
3. compute the verification value of the `ClientFinished` message using pms' and ms';
4. compare the verification value obtained in step (3) with the value that R' received from A. A match between these two values indicates a correct guess. Otherwise, R' repeats steps (1) to (4) using a different candidate password π''_A.

A is authenticated to R if the last message from A contains the correct verification value. The `master_secret` is subsequently used to derive further keys for the TLS record layer.

Enabling the Provisioning and Management of a Federated Grid Trust Fabric

Stephen Langella[1], Scott Oster[1], Shannon Hastings[1], Frank Siebenlist[2], Tahsin Kurc[1], Joel Saltz[1]

[1]*Department of Biomedical Informatics*
Ohio State University
Columbus, OH 43210
{langella,oster,hastings,kurc,jsaltz}@bmi.osu.edu

[2]*Mathematics and Computer Science Division*
Argonne National Laboratory
Argonne, IL 60439
franks@mcs.anl.gov

Abstract

In order to authenticate and authorize users and other peer-services, Grid services need to maintain a list of authorities that they trust as a source for issuing credentials. Grids inherently span multiple institutional administration domains and aim to support the sharing of applications, data, and computational resources in a collaborative environment. In this environment there may exist hundreds of certificate authorities, each issuing hundreds if not thousands of certificates. In such a dynamic multi-institutional environment with tens of thousands of users, credentials will be issued and revoked frequently, and new authorities will be added regularly. Clearly a Grid-wide mechanism is needed for maintaining and provisioning trusted certificate authorities, such that Grid services and users may make authentication and authorizations decisions against the most up-to-date trust information. In this paper we present the design and implementation of the Grid Trust Service (GTS), a federated framework for creating and managing a Grid trust fabric, enabling the provisioning of certificate authority information.

1. Introduction

As Grid computing technologies gain acceptance and adoption, the transition from highly specialized Grids with only a few institutional participants to a Grid environment with hundreds of institutions is becoming a reality. Security is of primary importance in the Grid and the support for secure communication, authentication, and authorization is a critical requirement, specifically in settings where sensitive data (e.g., patient medical information) must be accessed and exchanged. Also needed are mechanisms to establish and manage "trust" in the Grid so that asserted identities and privileges can be verified and validated with the required level of confidence. Within a collaboration, it is clear that the different institutions will have tiered levels of confidence in the users and service management policies of the various other institutions. While generally all institutions want to collaborate in some fashion, they will have services with varying security policy enforcement requirements. The interconnections, between clients and services that are able to securely communicate in the larger Grid, form conceptual overlays of trust, which we herein refer to as the "trust fabric" of the Grid. *Figure 1* shows an example trust fabric composed of four trust groups (Trust Groups A-D), over a worldwide Grid. The establishment, provisioning, and management of the trust fabric are critical to the scalability, maintenance and security of the Grid and other web service environments. This paper is concerned with the design and implementation of the Grid Trust Service (GTS) framework to facilitate the provisioning and management of a Grid trust fabric.

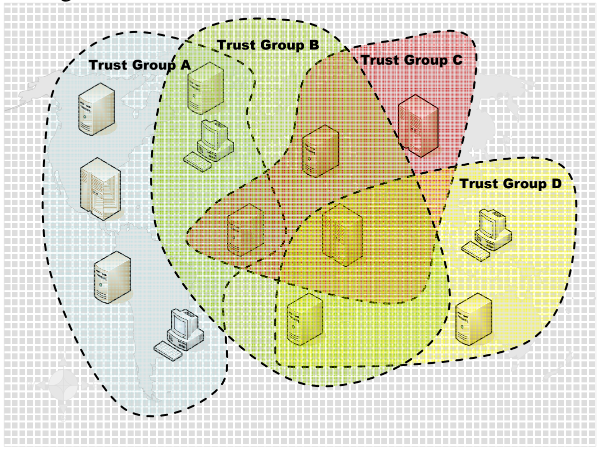


Figure 1 Conceptual Trust Fabric example with different trust groups

Many components of the Grid rely on having trust agreements in place. For example, when a user wants to access a service, she is authenticated based on an identity assigned to her. In the Grid, clients and services authenticate with one another using X.509 identity certificates. Grid Identities are assigned to users by authorities. When a grid-identity is asserted by an authority in the form of an X.509 identity certificate, it is digitally signed by that authority. One of the criteria relying parties use in making authentication decisions is determining whether or not the certificate presented is signed by a trusted certificate authority.

Thus, authentication requires a trust agreement between the consumers of X.509 identity certificates and the certificate authorities that issue them.

In a Grid environment, there may exist tens or even hundreds of certificate authorities, each issuing hundreds if not thousands of certificates. To make matters more complex, in a dynamic multi-institutional environment, the status of identities may be updated frequently. Identities and credentials can be revoked; suspended, reinstated, or new identities can be created. In addition, the list of trusted authorities may change. In such settings, certificate authorities will frequently publish Certificate Revocation Lists (CRL), which specify "black listed" certificates that the authority once issued but no longer accredits. For the security and integrity of the Grid, it is critical to be able to perform authentication and validate a given identity against the most up-to-date information about the list of trusted certificate authorities and their corresponding CRLs.

Each institution normally manages its own security infrastructure with its own CAs, and all client and services within such an administrative domain needs to be configured to trust the local trust roots. If collaborations span administrative domains, then participating entities have to be configured to trust the trust roots defined in the different organizations within the limits of their own local policies. The required trust root configurations to participate in such Virtual Organizations (VO) are complex, error prone and security policy sensitive. By centralizing the configuration management and provisioning collaborating clients and services "on demand", one can ensure that the correct and up-to-date trust-root information is made available. In this scenario, the central provisioning server becomes a trusted entity itself, and clients need to be configured to trust its provisioning information. In order to facilitate the trust in the provisioning servers, they should be locally known to the clients, which requires local provision servers to aggregate and to front-end remote ones.

The Grid Trust Service (GTS) is a Web Services Resource Framework (WSRF) [9] compliant federated infrastructure enabling the provisioning and management of a grid trust fabric. The salient features of the GTS can be summarized as follows:

- It provides a complete Grid enabled federated solution for registering and managing certificate authority certificates and CRLs, facilitating the enforcement of the most recent trust agreements.
- It allows the definition and management of trust levels, such that certificate authorities may be grouped and discovered by the level of trust that is acceptable to the consumer.
- The federated nature of the GTS, coupled with its ability to create and manage arbitrary arrangements of authorities into trust levels, allows it to facilitate the curation of numerous independent trust overlays across the same physical Grid.
- The GTS can also perform validation for a client, allowing a client to submit a certificate and trust requirements in exchange for a validation decision, which allows for a centralized certificate verification and validation.

2. Motivation

Our work is driven mainly by the requirements that are derived from the Grid security usage scenarios gathered from the Cancer Biomedical Informatics Grid (caBIG^{TM}) [1] program. This program, funded by the National Cancer Institute (NCI), was launched to provide a coordinated approach to the informatics requirements of basic and clinical cancer research and multi-institutional studies. The goal is to accelerate the delivery of innovative approaches for the prevention and treatment of cancer by facilitating sharing, discovery, and integration of distributed information and analytic resources. Although the caBIG effort started relatively recently, it is expected that the caBIG community will grow to comprise hundreds of organizations and many thousands of cancer-research participants from geographically dispersed medical centers, universities, government agencies, and commercial companies.

Given the sensitivity of the medically related data and the number of institutions involved, security has quickly become a high priority issue in caBIG. In order to articulate the security requirements of caBIG and evaluate existing technologies, a Security Technology Evaluation White Paper [8] has been developed. This white paper serves as one of the motivating influences for the GTS work. A key security issue is to implement an effective mechanism of managing the Grid-wide identities and privileges of large numbers of users across multiple organizations. Another key issue is to be able to authenticate and authorize users to caBIG services based on this information, such that access to resources can be restricted to individual users or a group of users. Considering the scale of caBIG, these issues become challenging.

The GTS framework has been implemented in the context of the caGrid [2, 30], which is the Grid software infrastructure of caBIG. caGrid is a service-oriented architecture and implementation that provides core services, toolkits and wizards for the development and deployment of community provided services. One of the primary design principles of caGrid is to leverage the open Grid standards [3, 4]. The caGrid infrastructure is built on top of the Globus Toolkit [5], the most widely used reference implementation of the Grid standards. The Globus Toolkit implements support for security via its Grid Security Infrastructure (GSI) [6, 7]. GSI requires the use of X.509 Identity Certificates for identifying a user. An X.509 certificate with its corresponding private key constitutes a unique credential or so-called "Grid credential" that is used to authenticate both users and services within the Grid. Under the current Globus release (4.0.3), the authentication process ensures that the X.509 Identity Certificate provided by the peer was issued by a trusted certificate authority. However, one limiting issue with the current mechanisms is that trusted certificate authorities (CAs) and their CRLs are maintained locally on the file system of each Globus installation. When a client authenticates with a service, Globus locates the root CA and CRL of the client's Identity Certificate on the local file system. Once located, the Globus runtime validates the Identity Certificate against the CA certificate and CRLs. Although this approach is effective, it is difficult to provision CA certificates and CRLs in a large multi-institutional environment, as one has to ensure that all CA and CRL information must be copied to every installation and kept current with the dynamically changing environment. Given the sensitivity of the data in caBIG, it is critical that services authenticate clients against the most up-to-date list of CA certificates and CRLs. This requirement is one of the primary motivations behind the design and development of the Grid Trust Service (GTS).

3. Background and Trust Fabric Profiles

The XML Key Management Specification (XKMS) [10] specifies models and protocols for distributing and managing public key credentials. XKMS specifies a tiered service model allowing applications to leverage the level of functionality that meets their requirements. The design of the GTS mimics this tiered service model, and for the purpose of this paper we will describe the relevant tiers in terms of the locations of CA certificates and where certificate validation occurs.

Tier 0, also referred to here as the *Locally Stored, Locally Validated* (LSLV) profile, specifies that CA certificates are stored and accessed locally; certificate validation is done against the locally stored CA certificates. Although the deployment and maintenance of the LSLV profile seems simple, the realization of such a profile in a large and dynamic Grid environment faces several problems. First, the LSLV profile makes it difficult to provision CA certificates and their corresponding CRLs. Under the LSLV approach, every local environment is required to update its local trust store each time a new CA certificate becomes available or each time a CA publishes a new CRL. Moreover, the LSLV profile could pose a potentially serious security risk because it requires users to maintain their own trust fabric. A simple error by an inexperienced user could easily introduce a security hole in the environment.

Tier 1, also referred to here as the *Remotely Retrieved, Locally Validated* (RRLV) profile, specifies that CA certificates are retrieved remotely from a Trust Service; certificate validation is done locally against the remotely retrieved CA certificates. When deployed in a large Grid environment, the RRLV profile significantly improves upon the LSLV profile. Retrieving the latest CA certificates and corresponding CRLs remotely allows Grid services to perform validation against the latest trust fabric. It also moves the management of the trust fabric from the hands of users to the hands of administrators, who have more expertise and experience in managing trust. Performing the validation locally also allows services and applications to enforce local validation policies that may go beyond those specified in X.509 certificate validation. At the same time local validation can be a potential security risk, if the local validation process is not effectively enforced.

Tier 2, also referred to as the *Remotely Stored, Remotely Validated* (RSRV) profile, specifies that CA certificates are stored remotely; all certificate validation with exception to validating the certificate of the remote validation service is done remotely against the remotely stored CA certificates. The RRLV profile and the RSRV profile are similar in that validation is done against the latest trust fabric. They differ in where the validation is done. Under the RSRV profile validation is done remotely, which removes the validation from the hands of the local providers minimizing potential security exposure when validation is not strongly enforced locally. However, it introduces a potential performance problem in that a local service is required to contact a remote validation service every time validation is required.

4. Grid Trust Service (GTS)

The Grid Trust Service (GTS) is a WSRF compliant Grid Service framework for creating, managing, and provisioning of a federated Grid trust fabric. Establishing trust in the Grid is rooted in the problem of determining whether or not to trust a given certificate authority. Through its service interface, the GTS provides the ability to register and manage certificate authorities. Using the GTS, Grid entities (services and clients) can discover the certificate authorities in the environment, decide whether or not to trust a certificate authority, and determine the levels of trust assigned to a certificate authority.

In implementing a trust fabric in a Grid environment, we envision that the trust fabric will consist of Grid users and administrators, Grid services, multiple CAs, and multiple GTS instances. The flexibility of GTS allows many possible deployment scenarios. For example, an institution can set up a local CA and GTS instance. Alternately, a group of organizations may all share a common CA for certificates and a GTS to maintain the list of trusted external CAs. In any deployment, each Grid user will be given a certificate, signed by a CA that can be used by services to authenticate the user. Similarly, each Grid service will be given a certificate, also signed by a CA, so that a client application, user, or other service can check the integrity of the service. Since GTS instances are Grid Services, they should be assigned certificates as well.

As deployments leveraging the GTS to maintain the trust fabric are effectively delegating this responsibility to the GTS, it is imperative the GTS instance(s) can be trusted. Traditionally a trust "bootstrapping" approach is adopted wherein clients and services communicating with the GTS are manually configured to trust its CA. Additionally, by default, the GTS clients perform host authorization against the specific GTS with which they are communicating. This ensures the service providing the information about the trust fabric (the GTS) has a certificate signed by a trust authority, and that it is provably the specific instance the client intended to communicate with. There are multiple possible deployment options for assigning certificates to GTS instances. A possible way is that each GTS instance has a self-signed certificate (i.e., serving as its own CA). In such a deployment, clients and services are manually configured to trust the self-signed certificates of the GTS instances they intend to interact with. Alternatively, there can be one (or a few) *trusted root-CA*, which will be used to assign the certificates to each GTS instance. Installations in the Grid are then bootstrapped to trust this authority or small set of authorities. Note that even if the clients are pre-configured with the trusted CAs, the GTS infrastructure can be used as a distribution mechanism of the CA's CRLs.

An advantage of the deployment with self-signed GTS certificates is that it does not require a root-CA to exist. If clients and services will only interact with a few GTS instance, this could be a preferred way of deployment. However, GTS certificates cannot be revoked in such a deployment. An advantage of the deployment with root-CA is that if the root-CA monitors integrity of GTSs, it can revoke GTS certificates and publish CRLs, which will include the revoked GTS certificates, in case of a security breach (with the acknowledged issue of communicating the revoked GTS-certificate to the GTS-clients when the distribution mechanism relies on the GTS).

While GTS facilitates management of certificate authority lists, the trust establishment with a CA and setting its trust level is a manual process. That is, the administrator of a GTS instance is expected to exchange correspondence with the owner of the CA to be added to the list of CAs managed by the GTS instance. Once a trust level, or set of trust levels, has been established, the CA can be added to the list of CAs so that it can be discovered by users and services. The trust levels of the CA can be used by a client or service to determine whether or not to trust the CA. (We will describe the management of trust levels in greater detail in Section 4.2.) In the trust fabric setting, a CA is responsible for publishing its CRL so that local CRLs and CRLs maintained by GTS instances can be updated. In addition to the level of trust, each GTS maintains a status value for each CA in its list of CAs. By changing the status of a CA (e.g. from "trusted" to "suspended"), the GTS can report to a relying parties that any certificates from the CA in question should not be trusted at this time, this can be valuable if there is a security breach.

In a large Grid environment, it is desirable to have a federated trust fabric for redundancy and scalability, and for the integration of multiple trust overlays. A possible way of federating GTS instances is to create a hierarchical structure, in which there are *authority GTS instances* and *subordinate GTS instances.* The authority GTS instances maintain lists of trusted CAs and CRLs and synchronize with CAs for updates. The subordinate GTS instances can be designed to synchronize with one or more authority GTS instances. In this way, when the state of the trust fabric changes (e.g., because of publishing a new CRL), the updates need not be broadcast to all GTS instances individually. The approach for federation of GTS instances is discussed in Section 4.5.

4.1. Profiles Supported by GTS

Out of the box, the Globus Toolkit supports the LSLV profile, and the GTS adds the support for both the RSRV profile and the RRLV profile. Our use cases for enabling trust between identity providers and consumers of identities and between attribute authorities and consumers of attributes call for the use of Remotely Stored Remotely Validated (RSRV) profile. It is also anticipated that future releases of Globus will support callouts to remote validation services. The GTS supports this profile by providing a validation operation through its service interface. In this manner, the GTS is a validation service allowing clients to submit validation criteria and an X.509 certificate chain for validation.

Figure 2 illustrates an example usage scenario of the RSRV profile. In this example, a GTS instance is used to manage and validate client certificates in caGrid. This example has a Dorian [11] instance serving as a CA and proxy certificate generator. Dorian is a Grid service infrastructure for the management of Grid user accounts. Dorian provides an integration point between external security domains and the Grid security domain, enabling users to obtain Grid credentials using their locally provided authentication mechanism. In *Figure 2*, an Ohio State University (OSU) user authenticates to the OSU authentication system using her local user name and password. Upon successfully authenticating, the user is given a signed SAML assertion, which can be given to Dorian in exchange for a Grid proxy or Grid credentials. In the example, a trust agreement has been established between the GTS instance and the Dorian instance wherein the Dorian instance's CA is listed as a trusted authority in the GTS instance. Trust and the trust level(s) between the GTS and Dorian instances can be established by the administrator of Dorian and the administrator of GTS through the exchange of information such as certificate policy and certification process statements. When the user presents the Grid proxy certificate to a Grid service, the grid service first performs a check to ensure that the user is the holder of the private key associated with the proxy certificate, the proxy certificate is then sent to the GTS instance for validation (Step 6: "Is Proxy Trusted?" in *Figure 2*). The GTS instance can respond back to the service with a "yes" or "no" answer. If the response is "no", the service prevents the user from accessing its resources. If the response is "yes", the service can take additional steps to authorize the user and control her access to the service's functionality based on the authenticated user's privileges.

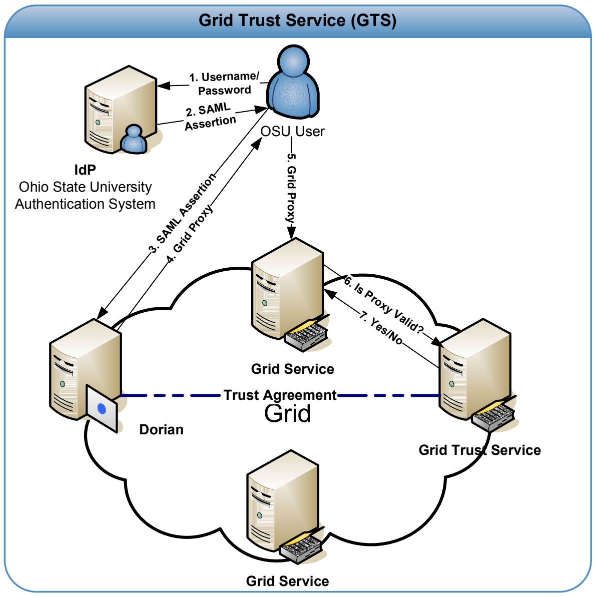


Figure 2 Trust Service (GTS) RSRV Profile

The GTS also supports the RRLV profile for accessing the trust fabric. This is because the current release of Globus Toolkit (Globus 4.0.3) does not provide a means of supporting remote validation of credentials; rather validation must be done against a locally maintained trust fabric. To enable Globus to authenticate users by validating their credentials against the trust fabric maintained in the GTS, the RRLV profile is employed. The GTS provides a framework called SyncGTS, which is embedded in the Globus runtime to automatically synchronize the local trust certificate store with the latest trust fabric maintained in the GTS. The SyncGTS functionality is presented in more detail in Section 5.

Figure 3 illustrates how authentication and certificate validation can be performed with the RRLV profile in GTS. When a Grid service is invoked, Globus authenticates the client by validating that the Grid proxy provided is signed by a trusted certificate authority. The certificate is validated against the local trusted certificates directory as is seen in the figure. In the example in *Figure 3*, the Dorian certificate authority has been registered with the GTS as a trusted certificate authority and Globus has been configured to synchronize its local trusted certificate store with the GTS. Thus when the OSU user invokes a Grid service using her Dorian-obtained proxy, she will be successfully authenticated by Globus. Future versions of Globus will support a credential-validation callout, at which time the RSRV GTS profile, illustrated in *Figure 2* can be employed.

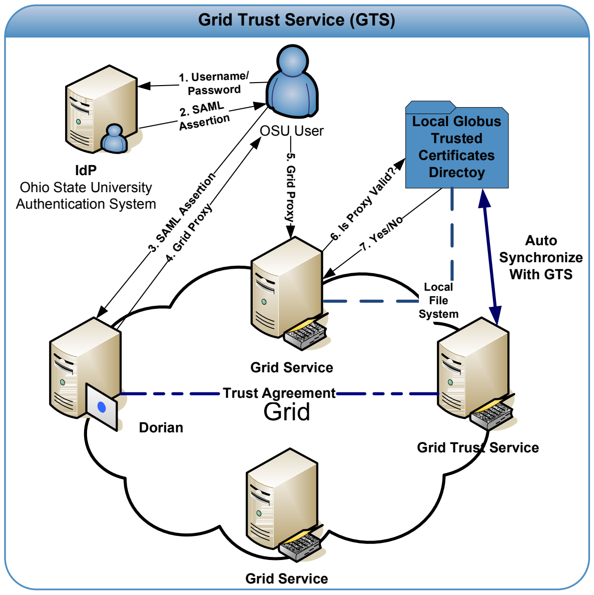


Figure 3 Grid Trust Service (GTS) RRLV Profile

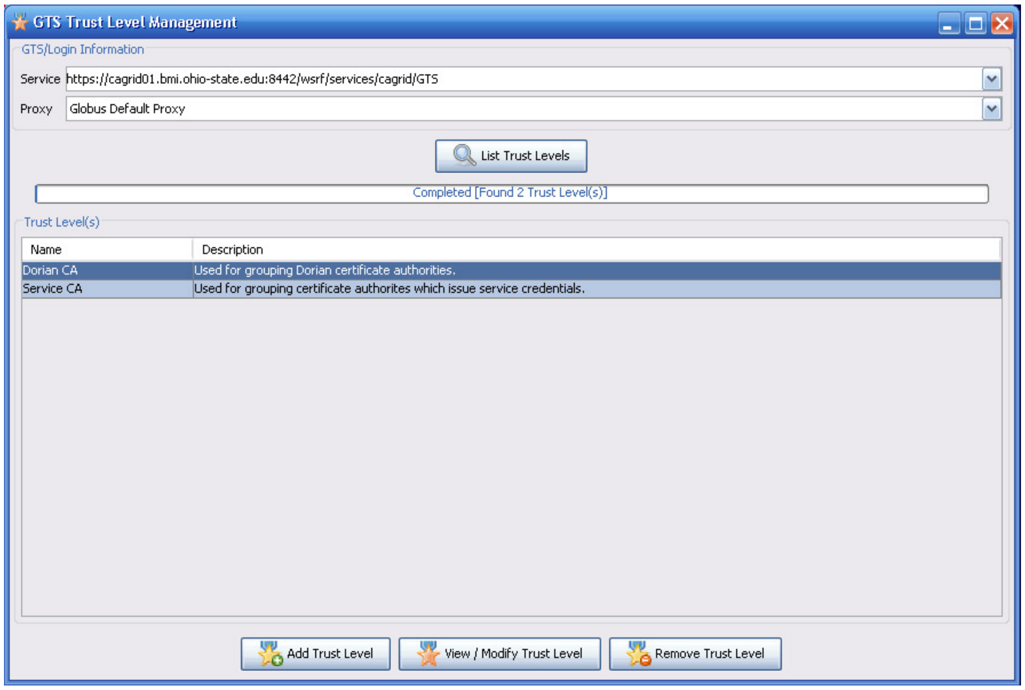
4.2. Managing Trust Levels in GTS

The trust level specifies the level of confidence with which a given certificate authority is trusted in the fabric in which it is deployed. In the Grid, one can assume that certificate authorities will be trusted with different levels of confidence.[1] There will be multiple types and instances of certificate authorities. Some authorities may be used to assert identities; other authorities may be used to assert digitally signed documents. Even certificate authorities asserting the same thing may have differing levels of trust associated with them, as they may employ different policies for issuing and validating identities. For example, a certificate authority may require that anyone applying for a certificate present official documentation about their real identity. The CA issues a certificate to the applicant after these documents are reviewed by the CA staff. Another certificate authority may automatically issue certificates based on an online application submitted by the applicant -- the applicant may have been requested to log on to the system using a user id and password. In these cases, the first certificate authority has a stricter policy for issuing certificates; thus, it is reasonable to expect that the first certificate authority should be trusted more than the second certificate authority . More specific examples of trust levels are CAs that are accredited by the International Grid Trust Federation (IGTF) [31] or CAs whom use a Hardware Security Module (HSM) for storing their private key.

In order to model different levels of trust in the trust fabric, the GTS provides a mechanism for its administrators to define and manage trust levels. When certificate authorities are registered into the trust fabric they are assigned one or more trust levels. Clients can specify the level of trust that they require when discovering trusted CAs or when requesting validation. Trust levels in the GTS each consist of a unique name (value) and description. The unique name is used to implicitly bind a certificate authority to a trust level. The description is used as a human readable method of understanding what a specific trust level represents. Through its Grid service interface, the GTS provides several operations for accessing and administrating trust levels. The *getTrustLevels()* operation is a publicly accessible operation which provides a list of all the trust levels existing in a GTS. The *addTrustLevel(), updateTrustLevel(),* and *removeTrustLevel()* operations are accessible to GTS administrators, providing them with a method of administering trust levels. *Figure 4* illustrates the trust level management interface in the GTS administrative user interface (admin UI).

On initial deployment, the GTS does not come pre-configured with a set of trust levels, at the time of development it was determined that it was desirable to allow Grid administrators to define trust levels based on the security policy defined by their Grid. The GTS provides the tools for administrators to assign and manage trust levels to CAs. The system is flexible in that each GTS instance can have its own trust levels. It

[1] The trust level concept in the Grid is similar to obtaining an identity card from different institutions. Obtaining a passport will require that the application provide more documentation and a more thorough background check be performed. On the other hand, getting a library card will require a less strict background check and less documentation


GTS Trust Level Management
GTS/Login Information
Service https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Proxy Globus Default Proxy
List Trust Levels
Completed [Found 2 Trust Level(s)]
Trust Level(s)
Add Trust Level
View / Modify Trust Level
Remove Trust Level

Name	Description
Dorian CA	Used for grouping Dorian certificate authorities.
Service CA	Used for grouping certificate authorites which issue service credentials.

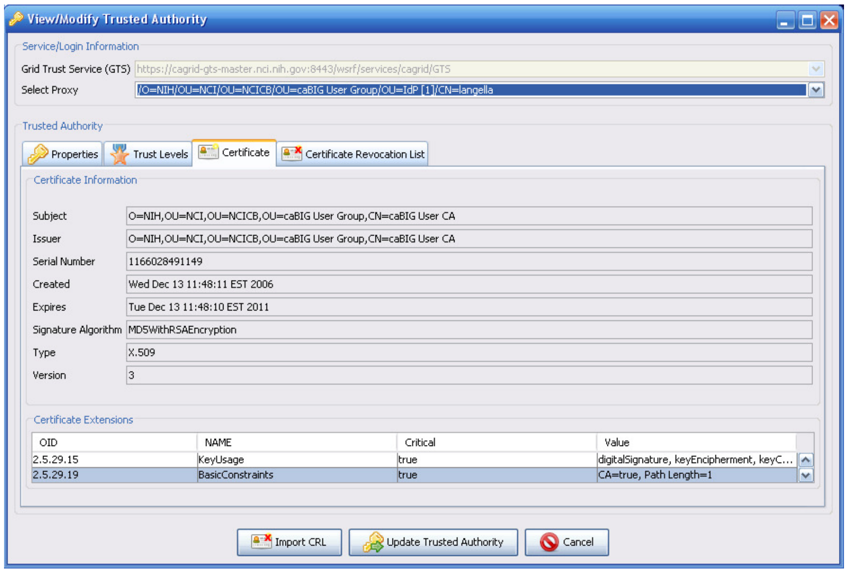
Figure 4 Admin UI for trust level management

may be desirable to have a common set of trust levels across multiple institutions, however, so that a client or a service can interpret the trust level associated with a CA correctly. In that case, a standard for trust levels should be accepted by the community and made available in a common repository or distributed to each GTS instance. Note that having a common set of trust levels does not require each GTS instance to assign the same trust level to the same CA. A GTS instance may assign a higher level of trust to a CA than another GTS instance. However, a common set of trust levels allows a client to interpret the trust level assigned by each GTS instance correctly and determine whether or not to trust the corresponding CA.

4.3. Managing Certificate Authorities

The GTS service interface provides several operations for registering and managing trusted certificate authorities. Registration of a certificate authority requires the specification of the CA's root certificate, a set of trust levels, a status, and an optional CRL. The CA's root certificate is required for validating certificates. The set of trust levels specifies the level of trust associated with the CA. The status specifies the current state of the certificate authority; the status can be set to "trusted" or "suspended". Setting the status of a certificate authority allows it to be temporarily added and removed from the trust fabric. For instance, if the security of a CA has been compromised, its status can be set to "suspended" to quickly inform the relying parties not to trust certificates issued and signed by the CA. For each trusted certificate authority, the GTS maintains a Certificate Revocation List (CRL). The CRL contains a list of certificates that have been revoked by the CA.

The GTS makes several operations available to administrators for managing trusted certificate authorities, these include *addTrustedAuthority()*, *updateTrustedAuthority()*, *removeTrustedAuthority()*, and *updateCRL()*. The *updateCRL()* operation provides a mechanism for CAs to publish their CRL immediately after it is locally updated. For example, Dorian revokes a user's certificate and adds an entry to its CRL, every time a user's account is suspended. Upon updating its CRL, Dorian immediate publishes it to the GTS. The *updateCRL()* operation can be invoked by GTS administrators and individual trusted certificate authority administrators. The GTS provides a mechanism of assigning and maintaining a list of individual trusted certificate authority administrators. *Figure 5* illustrates the trusted certificate authority management interface in the GTS admin UI.



Grid Trust Service (GTS) https://cagrid-gts-master.nci.nih.gov:8443/wsrf/services/cagrid/GTS
Select Proxy /O=NIH/OU=NCI/OU=NCICB/OU=caBIG User Group/OU=IdP [1]/CN=langella

Subject	O=NIH,OU=NCI,OU=NCICB,OU=caBIG User Group,CN=caBIG User CA
Issuer	O=NIH,OU=NCI,OU=NCICB,OU=caBIG User Group,CN=caBIG User CA
Serial Number	1166028491149
Created	Wed Dec 13 11:48:11 EST 2006
Expires	Tue Dec 13 11:48:10 EST 2011
Signature Algorithm	MD5WithRSAEncryption
Type	X.509
Version	3

OID	NAME	Critical	Value
2.5.29.15	KeyUsage	true	digitalSignature, keyEncipherment, keyC…
2.5.29.19	BasicConstraints	true	CA=true, Path Length=1

Figure 5 Admin UI for managing a trusted CA

4.4. Managing Administrators

Many of the operations provided by the GTS provide a means of administrating the trust fabric and are therefore restricted to GTS administrators. The GTS provides three operations for managing the assignment of administrative roles to users: *addPermission(), revokePermission(),* and *findPermissions().* These operations are, themselves, obviously restricted to GTS administrators.

The GTS allows for the assignment of two types of permissions; GTS Administrators, and Trusted CA Administrators. GTS Administrators are "super users" and can perform any operation on a GTS (i.e. manage certificate authorities, manage trust levels, manage permissions, etc). A Trusted CA Administrator permission corresponds to a specific CA, giving a user the ability to update the CRL for the corresponding CA.

4.5. Federation

Redundancy and scalability are critical properties of a federated trust fabric. Serious performance implications will occur if all entities in the Grid are discovering and performing validation against a trust fabric maintained in a central GTS. In order to enable a federated trust fabric, each GTS can be administered to synchronize with a set of authoritative GTSs. GTSs can inherit both trust levels and trusted certificate authorities from its authority GTSs. Registering an authority GTS requires the specification of the following properties: service's uniform resource identifier (URI), priority, whether or not to synchronize the trust levels, time to live, whether or not to perform authorization, and the authority service's identity. The priority property is used for resolving conflicts between authority GTSs, for example if two authority GTSs have a listing for the same certificate authority, the authority GTS with the highest priority will be used for obtaining that certificate authority, and its corresponding information (e.g. its CRL). If contact to an authoritative GTS is lost for a significant amount of time, the trust fabric within the subordinate GTS may become significantly out of date; this could be a potential security risk. The time to live property specifies how long certificate authorities obtained from authoritative GTSs will be valid for in the subordinate GTS. The time to live on a given certificate authority record is reset after each synchronization with the authority GTS. If contact with an authority GTS is lost, the time to live will expire and the certificate authority will be removed from the subordinate's trust fabric.

Figure 6 illustrates an example of how multiple GTSs can be deployed to create and manage a federated trust fabric. In the example there are five GTSs: caGrid GTS, TeraGrid GTS, OSU GTS, caGrid/TeraGrid GTS, and UT GTS. The caGrid GTS has no authority GTSs, it manages the certificate authorities A and S. The TeraGrid GTS has no authority GTSs, and it manages the certificate authorities X and S. The OSU GTS has one authority GTS, the caGrid GTS. The OSU GTS inherits the certificate authorities A and S from its authority the caGrid GTS. The OSU GTS manages an additional certificate authority B. The OSU GTS is an example of how the global trust fabric can be extended to include local trusted certificate authorities, in this case and the additional certificate authority CA B, which is trusted by OSU. The caGrid/TeraGrid GTS has two authority GTSs, the caGrid GTS and the TeraGrid GTS. The TeraGrid GTS inherits CA A from the caGrid GTS and CA X from the TeraGrid GTS, since the caGrid GTS has a higher priority then the TeraGrid GTS, it inherits CA S from the caGrid GTS. The caGrid/TeraGrid GTS is an example of how two existing trust fabrics from two different Grids can be joined together. Finally the UT GTS has one authority GTS, the TeraGrid GTS. The UT GTS inherits CA X and CA S from the TeraGrid GTS. The UT GTS is an example of standing up a GTS for better redundancy and scalability.

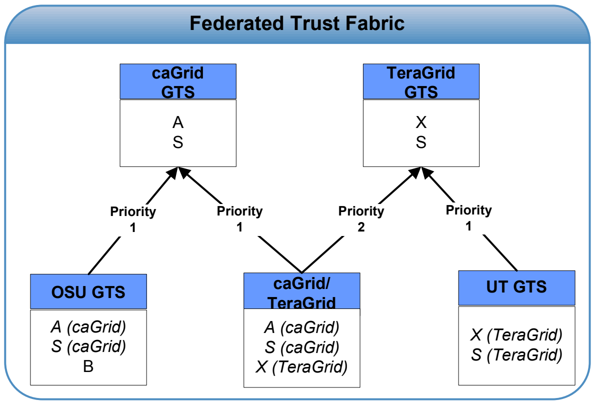
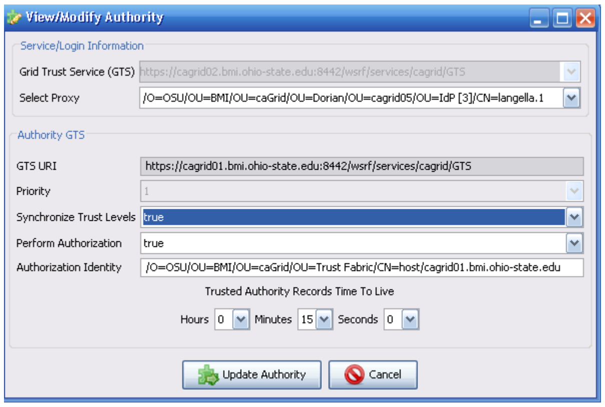


Figure 6 Example federated GTS deployment

Supporting a federated trust fabric across GTSs introduces additional metadata to be associated with trust levels and certificate authorities. This metadata includes the "Source GTS", "Authority GTS", and "Time to Live". The "Source GTS" specifies the service URI of the GTS in which the trust level or certificate authority was inherited from. The "Authority GTS" specifies the service URI of the GTS that is the authority of trust level or certificate authority. "Time to Live" specifies the date until which the certificate authority entry is valid.

Through its service interface, the GTS provides administrative operations for managing the federation. The trust fabric is managed by specifying authority GTSs for a given GTS. To support this, the GTS service interface provides the following operations: *addAuthority()*, *updateAuthority()*, *removeAuthority()*, and *updateAuthorityPriorities()*. *Figure 7* illustrates the authority GTS management interface in the GTS admin UI.


View/Modify Authority

Service/Login Information

Grid Trust Service (GTS): https://cagrid02.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS

Select Proxy: /O=OSU/OU=BMI/OU=caGrid/OU=Dorian/OU=cagrid05/OU=IdP [3]/CN=langella.1

Authority GTS

GTS URI: https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS

Priority: 1

Synchronize Trust Levels: true

Perform Authorization: true

Authorization Identity: /O=OSU/OU=BMI/OU=caGrid/OU=Trust Fabric/CN=host/cagrid01.bmi.ohio-state.edu

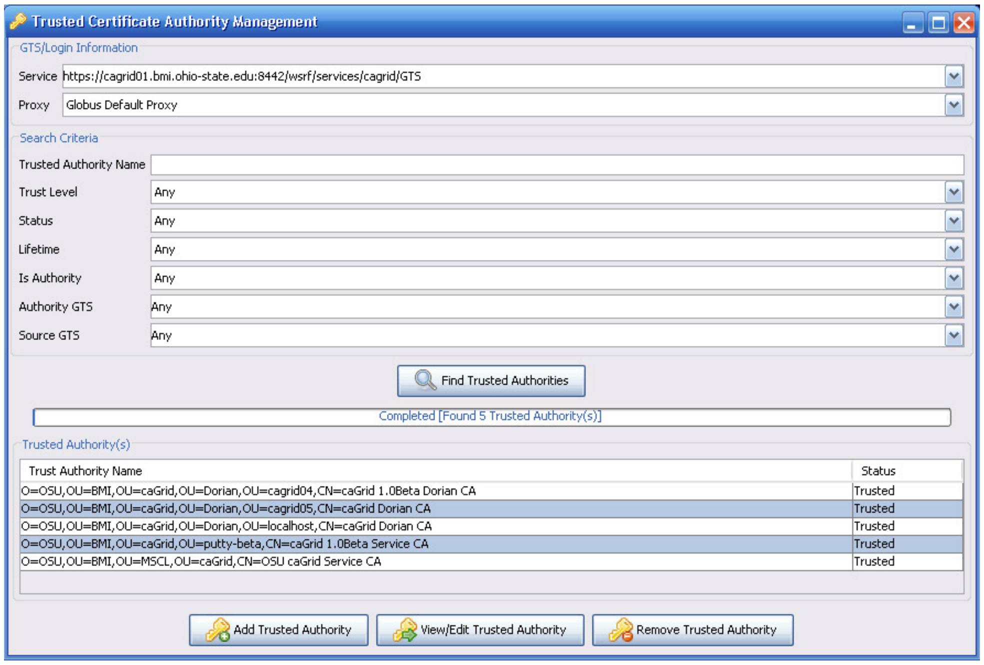
Trusted Authority Records Time To Live

Hours: 0 Minutes: 15 Seconds: 0

Figure 7 Admin UI for GTS Authority Management

4.6. Discovery and Remote Validation

In order to support the RRLV profile, local validation processes will need a method of discovering and obtaining trusted CAs from the pre-configured GTS. The GTS provides a publicly available operation, *findTrustedAuthorities()* through its service interface. In discovering trusted certificate authorities the GTS allows the specification of search criteria. The search criteria include the name of the certificate authority, the trust level, the status, the lifetime, the source GTS, and the authority GTS. Using this operation, validators implementing the RRLV profile can discover a list of trusted certificate authorities based on the trust level. Additionally, trust fabric administrators may leverage this operation for discovering the trust fabric such that they may administer it. *Figure 8* illustrates the discovery interface in the GTS admin UI.



Trusted Certificate Authority Management

GTS/Login Information

Service	https://cagrid01.bmi.ohio-state.edu:8442/wsrf/services/cagrid/GTS
Proxy	Globus Default Proxy

Search Criteria

Trusted Authority Name	
Trust Level	Any
Status	Any
Lifetime	Any
Is Authority	Any
Authority GTS	Any
Source GTS	Any

Completed [Found 5 Trusted Authority(s)]

Trusted Authority(s)

Trust Authority Name	Status
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid04,CN=caGrid 1.0Beta Dorian CA	Trusted
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=cagrid05,CN=caGrid Dorian CA	Trusted
O=OSU,OU=BMI,OU=caGrid,OU=Dorian,OU=localhost,CN=caGrid Dorian CA	Trusted
O=OSU,OU=BMI,OU=caGrid,OU=putty-beta,CN=caGrid 1.0Beta Service CA	Trusted
O=OSU,OU=BMI,OU=MSCL,OU=caGrid,CN=OSU caGrid Service CA	Trusted

Figure 8 Admin UI for Trust Fabric Discovery

To support the Remotely Stored, Remotely Validated (RSRV) model, the GTS takes on the role of a validation service. The GTS *validate()* operation enables clients to submit, for validation, a certificate chain and validation criteria. The GTS uses the X.509 validation specifications for validating the certificate chain. Additionally, it enforces the X.509 Proxy extensions for validating the certificate chain, if an X.509 proxy certificate exists in the chain. The GTS uses the validation criteria to identify a set of certificate authorities to validate the specified certificate chain against. The validation criteria are similar to discovery criteria, optionally allowing the set of certificate authorities to validate against to be limited based on the following: the name of the certificate authority, the trust level, the status, the lifetime, the source GTS, and the authority GTS.

5. SyncGTS

As mentioned, the current Globus Toolkit release (version 4.0.3) supports the LSLV profile for certificate validation. To meet the security requirements of caBIG, we needed to ensure that certificate validation is performed against the latest trust fabric. To this end, without having to modify the Globus Toolkit we created SyncGTS. SyncGTS is a plugin for the Globus Toolkit enabling it to support the RRLV profile. Globus performs authentication, or certificate validation, against a trusted certificates directory on the local file system. SyncGTS, shown in *Figure 9* maintains the local trusted certificates directory for Globus. When SyncGTS synchronizes with a set of GTS instances, it copies the CA certificates and CRLs to the local trusted certificates directory using proper Globus naming conventions, purging all certificate and CRLs that existed from the previous synchronizations. SyncGTS is configured with a synchronization description, which describes the synchronization criteria. Each synchronization criterion specifies a GTS to synchronize with and a set of search criteria for enforcing restrictions (trust levels, etc.) on the resulting certificate authority set. When SyncGTS is executed, it connects to all the GTS instances specified in the synchronization description and obtains a list of trusted certificate authorities and corresponding CRLs matching the specified search criteria.

For auditing purposes SyncGTS maintains a reporting model, *SyncReport*, describing each synchronization point. A report is generated and written to the local file system, each time SyncGTS synchronizes with a set of GTSs. The report describes when the synchronization occurred, the synchronization description executed, and the outcome of the synchronization. The outcome details which GTS a certificate authority and CRL came from and where on the file system the CA information and CRL were written. It also describes any certificate authorities and CRLs that were removed, because they were remnants of a previous synchronization.

SyncGTS can support several deployment options. It can be embedded directly into a Globus container as a Grid service, enabling all services running in that container to perform authentication against the GTS maintained trust fabric. SyncGTS can also be embedded directly in client-side applications by leveraging the SyncGTS Java API. In addition, SyncGTS provides command line utilities, which can be leveraged on both the client and service sides.

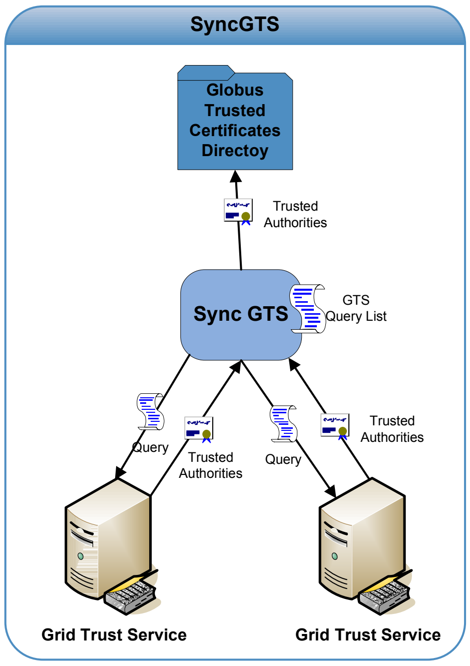


Figure 9 Conceptual View of SyncGTS

6. Related Work

Grid-wide management of security related information (such as certificates, credentials, user accounts, and authorization information) is a critical and challenging issue. The challenge stems from the fact that the Grid consists of autonomous end-points with their own security policies and infrastructure and it is a dynamic environment. A number of toolkits and service architectures have been developed, for example, to manage user credentials and user accounts. The MyProxy Credential Management Service [22, 23] is an open source project for managing X.509 Public Key Infrastructure. MyProxy provides the ability to manage private keys and certificates for Grid users, and also has an option to provision the trusted CAs and CRLs to the users. The current client implementation for the provisioning is C-based, and does not deploy webservice compliant protocols. Furthermore, it lacks automatic synchronization capabilities and possible aggregation mechanisms of provisioning information.

The Portal-based User Registration System (PURSe) [24] implements a user interface for users to register and access their Grid credentials. It uses SimpleCA and MyProxy as the backend system for management of Grid credentials. GAMA [25] is a GSI-based infrastructure that provides a backend server for creating and managing X.509 credentials for users and a suite of portals that serve as interfaces for users and administrators to access GAMA's functionality. Dorian [11] provides a Grid service infrastructure, based on the use of public key certificates and SAML assertions, for managing and federating user identities in the Grid. It facilitates combined use of SAML and Grid certificates to authenticate users to the Grid environment through their institution's authentication mechanism. The GTS differs from these systems in that it addresses the complementary problem of federating trusted identity authorities (i.e., Certificate Authorities) with different trust levels and validation of user certificates against this federated environment.

Management of trust is recognized as an important component of security in distributed environments [6, 12-21]. Manchala [12] describes trust models and metrics in e-commerce applications and discusses how risk can be analyzed under different models. Azzedin and Maheswaran [13] present a trust model for a Grid environment. Their approach models trust based on behavior and reputation of entities that interact with others. They describe techniques for computing this type of behavior trust, how it evolves in an environment, and how it can be managed in a Grid setting. GridAdmin, proposed by Quillinan, Clayton, Foley, [14] is a system that provides support for automatic handling of requests for administrative actions and resource allocations. The system incorporates trust metrics in responding to and ranking such requests. Weaver et. al. [17] discuss trust-sharing agreements and an IT infrastructure for federated security in distributed healthcare applications. Hwang and Tanachaiwiwat [18] propose a trust model and systems using this trust model for dynamic resource allocations. Grandison and Sloman [26] present a toolkit that provides support for specifying and monitoring trust relationships for Internet applications. Ahsant et. al. [27] discuss how business trust relationships can be propagated to the Grid environment and how these relationships can be federated dynamically. Li, Zhu, and Lam [28] propose a two-level trust model, in which the first level defines trust relationships between virtual organizations and the second level (lower level) specifies trust relationships within a domain. Park, Moon, and Sohn [20] propose an approach and a service for validation of certificates in the Grid using XKMS. Their system implements support for realizing trust relationships in a dynamic environment. Basney et. al. [29] describes extensions to the basic Grid security architecture in order to support negotiation and dynamic establishment trust relationships between entities in the Grid. Thompson et. al. [21] discusses trust and trust models based on CAs for the Grid environment. Our work complements the previous work on trust management in that earlier work focused on specification of trust and establishment and management of trust between entities. The GTS, on the other hand, enables Grid-wide management of trusted Certificate Authorities with different trust levels.

7. Conclusions and Future Work

We have shown the need for secure multi-institutional Grids and have demonstrated current approaches for enabling access to trusted CA certificates as well as CRLs. We have outlined and analyzed the problems that are incurred when attempting to scale Grid security across multiple institutions, which each has its own set of authorized users as well as local security constraints. We have presented GTS, a design and implementation for providing a Grid-wide trust fabric that solves issues in curating a Grid-wide trust network. The GTS' ability to store and manage large networks of trusted CA certificates and their CRLs will enable simple and safe scalability in leveraging secure and trusted certificates in a Grid. This CA certificate trust network and corresponding CRLs provided by GTS as a Grid service will enable validating users and services from any institution against a known set of trusted certificate authorities. In conjunction with SyncGTS, GTS will enable the local caching of the trust network so that local user validation can occur. The combination of these tools, along with other Grid security measures such as federated identity management, not only will enable Grid security to be less cumbersome to manage for administrators but also will increase the ability to create much larger, more secure, and disparate multi-institutional Grids.

Although we use the tiered approach of the XKMS, our current implementation does not employ the XKMS defined protocols and interfaces in the GTS infrastructure. We plan to incorporate this into GTS in a future release. Moreover, we plan to extend GTS such that it can provision On-line Certificate Status Protocol (OCSP) responder information. We expect the Globus Toolkit to support OCSP in its future release and will ensure that GTS can provide a mechanism for OCSP-enabled clients and services to contact and trust the right authorities. Similarly we also plan on exploring adding support for the Server-based Certificate Validation Protocol (SCVP) [32]. Another planned future extension to GTS is the support for the provisioning of attribute and authorization authorities. With the latter information, client and service will be configured to allow them to query the relevant and trusted attribute and authorization services.

In the future we would like to collaborate with the IGTF to define a standard set of trust levels. We would also like to work with the IGTF to investigate how the GTS can be used to distribute IGTF accredited trust roots and whether Grids using the IGTF would be interested in leveraging our framework.

Acknowledgements:

This work was supported in part by the National Cancer Institute's Center for Biomedical Informatics, under the caBIG^{TM} program (#79077CBS10), by the National Science Foundation Grant ANI-0330612, by the Ohio Board of Regents BRTTC program (#BRTT02-0003 and ODOD AGMT TECH 04-049), and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

References

[1] Cancer Biomedical Informatics Grid (caBIG^{TM}), https://cabig.nci.nih.gov.

[2] The caGrid infrastructure, https://cabig.nci.nih.gov/workspaces/Architecture/caGrid/

[3] I. Foster, C. Kesselman, and S. Tuecke., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International J. Supercomputer Applications, 15(3), 2001.

[4] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Technical Report,

Globus Project; http://www.globus.org/research/papers/ogsa.pdf, June 2002.

[5] The Globus Toolkit, http://www.globus.org.

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

[7] I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch, R. Butler, and D. Engert, "A National-Scale Authentication Infrastructure.", IEEE Computer, 33(12):60-66, 2000.

[8] Ken Lin and Gary Daemer "caBIG™ Security Technology Evaluation White Paper". https://cabig.nci.nih.gov/workspaces/Architecture/Security_Tech_Eval_White_Paper_Provisional, January 2006.

[9] Web Services Resource Framework (WSRF), http://www.oasis-open.org/committees/wsrf, 2006.

[10] XML Key Management Specification (XKMS) http://www.w3.org/TR/xkms/

[11] S. Langella, S. Oster, S. Hastings, F. Siebenlist, T. Kurc, and J. Saltz, "Dorian: Grid Service Infrastructure for Identity Management and Federation," The 19th IEEE Symposium on Computer-Based Medical Systems, Special Track: Grids for Biomedical Informatics, Salt Lake City, Utah., 2006.

[12] D.W. Manchala, "E-Commerce Trust Metrics and Models". IEEE Internet Computing 4(2): 36-44, 2000.

[13] F. Azzedin and M. Maheswaran, "Evolving and managing trust in grid computing systems," In the Proceedings of Canadian Conference on Electrical and Computer Engineering, vol.3, pp. 1424- 1429, 2002.

[14] T.B. Quillinan, B.C. Clayton, and S.N, Foley, "GridAdmin: Decentralising Grid administration using trust management". Third International Symposium on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks, pp. 184- 192, July 2004.

[15] M.H. Hanif Durad and C. Yuanda, "A vision for the trust managed grid". The Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, 2006.

[16] Y. Demchenko, C. de Laat, and V. Ciaschini, "VO-based Dynamic Security Associations in Collaborative Grid Environment," International Symposium on Collaborative Technologies and Systems (CTS 2006), pp. 38- 47, May 2006.

[17] A.C. Weaver, S.J. Dwyer, A.M. Snyder, J. Van Dyke, J. Hu, X. Chen, T. Mulholland, and A. Marshall, "Federated, secure trust networks for distributed healthcare IT services," In Proceedings of IEEE International Conference on Industrial Informatics (INDIN 2003), pp. 162- 169, Aug. 2003.

[18] K. Hwang and S. Tanachaiwiwat, "Trust Models and NetShield Architecture for Securing Grid Computing". Journal of Grid Computing, 2003.

[19] N. Nagaratnam et. al., "Security Architecture for Open Grid Services". Global Grid Forum Working Draft. 2003.

[20] N. Park, K. Moon, and S. Sohn, "Certificate validation service using XKMS for computational grid". In Proceedings of the 2003 ACM Workshop on XML Security (XMLSEC '03), ACM Press, New York, NY, Oct 2003.

[21] M.R. Thompson, D. Olson, R. Cowles, S. Mullen, and M. Helm. "CA-based Trust Model for Grid Authentication and Identity Delegation", Grid Certificate Policy Working Group, Global Grid Forum, Oct, 2002.

[22] J. Novotny, S. Tuecke, and V. Welch., "An Online Credential Repository for the Grid: MyProxy". Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), pp. 104-111, IEEE Press, Aug 2001.

[23] J. Basney, M. Humphrey, and V. Welch., "The MyProxy Online Credential Repository". Software: Practice and Experience, Volume 35, Issue 9, July 2005

[24] PURSe: Portal-Based User Registration Service, http://www.grids-center.org/solutions/purse/

[25] K. Bhatia, S. Chandra, and K. Mueller, "GAMA: Grid Account Management Architecture," E-Science '05, pp. 413-420, 2005.

[26] T. Grandison and M. Sloman, "Trust Management Tools for Internet Applications". The First International Conference on Trust Management. Springer Verlag, 2003.

[27] M. Ahsant, M. Surridge, T.A. Leonard, A. Krishna, and O. Mulmo, "Dynamic Trust Federation in Grids". In Proceedings of the 4th International Conference on Trust Management, Pisa, Tuscany, Italy, 2006.

[28] T.-Y. Li, H. Zhu, and K.-Y. Lam, "A Novel Two-Level Trust Model for Grid". International Conference on Information and Communications Security (ICICS 2003), pp. 214-225, 2003.

[29] J. Basney, W. Nejdl, D. Olmedilla, V. Welch, and M. Winslett, "Negotiating trust on the Grid". The 2nd Workshop on Semantics in P2P and Grid Computing, New York, 2004.

[30] Joel H. Saltz, Scott Oster, Shannon L. Hastings, Stephen Langella, William Sanchez, Manav Kher, Peter A. Covitz, "caGrid: design and implementation of the core architecture of the cancer biomedical informatics grid", Ed. Alvis

Brazma, Bioinformatics, Vol. 22, No. 15, 2006, pp. 1910-1916.

[31] International Grid Trust Federation (IGTF), http://www.gridpma.org/

[32] T. Freeman, R. Housley, A. Malpani, D. Cooper, W. Polk. “Server-based Certificate Validation Protocol (SCVP)”, IETF Draft

List of Acronyms

AA	Attribute Authority
AC	Attribute Certificate
ACL	Access Control List
ANL	Argonne National Laboratory
API	Application Programming Interface
ARP	Attribute Release Policies
ASN.1	Abstract Syntax Notation One
AUC	Apple University Consortium
B2B	Bridge-to-Bridge
B2C	Business to Consumer
BBWG	Bridge-to-Bridge Working Group
BDH	Bilinear Diffie-Hellman
CA	Certification Authority
caBIG	Cancer Biomedical Informatics Grid
CAD	Computer Aided Design
CAPI	Cryptographic Application Programming Interface
CAS	Community Authorization Service
CBE	Certificate Based Encryption
CERN	European Organization for Nuclear Research
CL	Certificateless
CMS	Cryptographic Message Syntax
CoG	Commodity Grid
CP	Certificate Policy
CRL	Certificate Revocation List
CSP	Cryptographic Service Provider
DCE	Distributed Computing Environment
DCE	Distributed Computing Environment
DER	Distinguished Encoding Rules
DKIM	Domain Keys Identified Mail
DLL	Dynamic Link Library
DN	Distinguished Name
DNS	Domain Name Systems
DOC	Electronic Document
DS	Digital Signature
DSA	Digital Signature Algorithm
DSL	Digital Subscriber Line
DSS	Digital Signature Services
DSSS	Data Security Systems Solutions
ECC	Elliptic-Curve Cryptography
ECDL	Elliptic Curve Discrete Logarithm
ECDSA	Elliptic Curve Digital Signature Algorithm
EE	End Entity
EEC	End Entity Certificates

EEMA	www.eema.org (pka European Electronic Messaging Association)
EM	Encryption Module
ESI	Electronic Signature Infrastructure
FBCA	Federal PKI Bridge Certificate Authority
FDA 21 CFR	Food and Drug Administration 21 Code of Federal Regulations
FICC	Federal Identity Credentialing Committee
FIPS	Federal Information Processing Standard
FNAL	Fermilab National Accelerator Laboratory
FPKIPA	Federal PKI Policy Authority
GAMA	Grid Account Management Architecture
GINA	Graphical Identification Authentication
GPS	Global Positioning System
GRAM	Grid Resource Allocation and Management
GSI	Global Security Interface
GSISSH	Global Security Interface Secure Shell
GT	Globus Toolkit
GTS	Grid Trust Service
GUI	Graphical User Interface
GxP	Good Pharmaceutical Practice
HCLE	Hierchical Certificateless Encryption
HCLS	Hierchical Certificateless Signature
HIBE	Hierchical Identity-Based Encryption
HIPAA	Health Insurance Portability and Accountability Act
HISS	Hierchical Identity-Based Signature
HPDC	High Performance Distributed Computing
HSM	Hardware Security Module
HSPD-12	Homeland Security Presidential Directive 12
IBE	Identifier-based Encryption
IdP	Identity Provider
IEEE	Institute for Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIS	Internet Information Server
IP	Internet Protocol
IPSEC	Internet Protocol Security
IRL	Identity Revocation List
ISO/ITU-T	International Organization for Standardization/International Telecommunication Union - Telecommunication Standardization Sector
JCA	Java Cryptography Architecture
JES	Java Enterprise System
JISC	Joint Information Systems Committee
KCA	Kerberos Certification Authority
KDC	Key Distribution Center
Kx509	Kerberized X.509
LDAP	Lightweight Directory Access Protocol

LHC	Large Hadron Collider
LSLV	Locally Stored, Locally Validated
MG	Module Manager
MIRACL	Multiprecision Integer and Rational Arithmetic C/C++ Library
NCI	National Cancer Institute
NGS	UK National Grid Service
NIST	National Institute of Standards and Technology
NSS	Network Security Services
NTP	Network Time Protocol
OASIS	Organization for the Advancement of Structured Information Standards
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OpenSC	Open Source Smart Card Project
OS	Open Systems
OSG	Open Science Grid
OSU	Ohio State University
OTP	One Time Passwords
PAM	Pluggable Authentication Modules
PCT	Patent Cooperation Treaty
PDA	Personal Digital Assistant
PECF	Password-Enabled and Certificate-Free
PERMIS	Privilege and Role Management Infrastructure Standards Validation
PIV	Personal Identity Verification
PKC	Public Key Cryptography
PK-CROSS	Public Key Cryptography for Cross-Realm Authentication in Kerberos
PKCS	Public-Key Cryptography Standard
PKG	Private Key Generator
PKI	Public Key Infrastructure
PK-INIT	Public Key Authentication in Kerberos
PLM	Product Lifecycle Management
PURSe	Portal-based User Registration System
RA	Route Attestations
RADIUS	Remote Authentication Dial In User Service
RAL	Rutherford Appleton Laboratory
RM	Request Managers
RO	Request Originators
RP	Relying Party
RRLV	Remotely Retrieved, Locally Validated
RSRV	Remotely Stored, Remotely Validated
S/MIME	Secure/Multipurpose Internet Mail Extensions
SAML	Security Assertion Markup Language
SAPI	Signature Application Programming Interface
SASL	Simple Authentication and Security Level

SDK	Software Developers Kit
SDSI-SPKI	Simple Distributed Security Infrastructure- Simple Public Key Infrastructure
SHA-1	Secure Hash Algorithm, as specified in FIPS 186-1 (also denoted SHA1)
SLCS	Short Lived Credentials Service
SMS	Systems Management Server
SODA	Signing and Storing Documents for Digital Accounting
SSH	Secure Shell
SSL	Secure Sockets Layer protocol
SSO	Single Sign-On
SWITCHaai	The Swiss Education & Research Network Authentication and Authorization Infrastructure
TA	Trusted Authority
TCA	Temporal Certification Authority
TCP	Transmission Control Protocol
TDC	Temporal Digital Certificate
TDCDP	Temporal Digital Certificate Declaration Practices
TDCIP	Temporal Digital Certificate Issuing Policy
TGT	Ticket Granting Ticket
TLS	Transport Layer Security
TPKI	Temporal Public Key Infrastructure
TSA	Time Stamp Authority
TTP	Trusted Third Party
UPN	User Principal Name
UPU-EPM	Universal Postal Union – Electronic Post Mark
URI	Unified Resource Identifier
USB	Universal Serial Bus
UUID	Universally Unique Identifier
VA	Validation Authority
VAT	Value Added Tax
VO	Virtual Organization
VOMS	Virtual Organization Membership Service
VSC	Virtual Smart Card
W3C	World Wide Web Consortium
WIP	Works-in-Progress
WSRF	Web Services Resource Framework
X.509	The ISO/ITU X.509 standard
XACML	Extensible Access Control Markup Language
XKMS	XML Key Management System
XML	Extensible Markup Language
XPCOM	Cross-Platform Component Object Model
XPIDC	Cross-Platform Interface Description Language