2nd Annual PKI Research Workshop
Pre-Proceedings

http://middleware.internet2.edu/pki03/

Gaithersburg, Maryland, USA
April 28-29, 2003

In cooperation with USENIX, the PKI Forum, and IFIP TC8

Organizers

General Chair: Ken Klingenstein, University of Colorado.

Program Chair: Carl Ellison, Intel Corporation

Steering Committee Chair: Neal McBurnett, Internet2.

Local Arrangements Chair: Nelson Hastings, NIST.

Program Committee:

Peter Alterman, NIH.
Matt Blaze, AT&T Labs Research.
Bill Burr, NIST.
Yassir Elley, Sun Microsystems.
Carl Ellison, Intel.
Stephen Farrell, Baltimore Technologies.
Richard Guida, Johnson and Johnson.
Peter Honeyman, University of Michigan.
Ken Klingenstein, University of Colorado.
Neal McBurnett, Internet2.
Clifford Neuman, USC.
Eric Norman, University of Wisconsin.
Tim Polk, NIST.
Ravi Sandhi, George Mason University.
Krishna Sankar, Cisco Systems.
Frank Siebenlist, Argonne National Laboratory.
Sean Smith, Dartmouth College.
Michael Weiner, Independent.

1st Annual PKI Research Workshop:

Archival materials live at `www.cs.dartmouth.edu/~pki02/`

Refereed Papers

An Overview of Public Key Certificate Support for Canada's Government On-Line (GOL) Initiative

Mike Just
Treasury Board of Canada, Secretariat (TBS)[1]
Just.Mike@tbs-sct.gc.ca

Abstract: The Canadian Federal Government is delivering on-line services to its citizens. A critical feature for ensuring the acceptance of these services is to ensure that security and privacy requirements are met. To this end, Canadian citizens may obtain an epass allowing them to securely obtain services through a government program web site. Technically, an epass is composed of a pseudonymous public key certificate. In this paper, we analyze the system in which this epass is managed and used, with particular emphasis as to how it supports the security and privacy of Canadian citizens.

1 Introduction

Governments have an obligation to protect the concerns of their *citizens* (hereafter referred to as *individuals*), above and beyond what might typically be provided for a more focused or specialized user base. In recent years, privacy is often cited as a primary concern. In Canada, this concern prompted the development and passing of the Personal Information Protection and Electronic Documents Act (PIPEDA) [PIPE00].[2]

As the Canadian Federal Government continues to offer online services, it is of the utmost importance that security and privacy requirements are met so as to lessen any potential concerns, and ensure user uptake. Though one of many service delivery channels (others include by phone and in-person), on-line service delivery can offer much efficiency and thus cost savings, but these can only be realized if on-line services are used. Canada has stated that government departments will have a complete online presence by 2005. With a population of greater than 31 million, and more than 1000 programs and services, the Canadian project is by no means small. This is especially true considering the collaborative potential with provincial and municipal governments.

In the remainder of this paper, we focus on the PK-based solution for the authentication of individuals using an epass. Technically, the epass is composed of a pseudonymous public key verification certificate along with its corresponding private key. The main results of this paper are twofold:

- To describe Canada's epass system, supporting secure access to online government services;
- To discuss how this system satisfies privacy and security requirements of individuals.

[1] The Treasury Board of Canada, Secretariat (TBS - http://www.tbs-sct.gc.ca/) is a central government agency whose mission is to manage the Government of Canada's human, financial and information resources. Within TBS, the Chief Information Officer Branch (CIOB - http://www.cio-dpi.gc.ca/) is responsible for coordinating Information Technology (IT) and IT Security activities.

[2] Whereas the Privacy Act [Priv85] applies to federal government institutions, PIPEDA applies to organizations that collect information for commercial purposes. Currently, PIPEDA applies to federally regulated organizations, but as of January 2004, it will also apply to provinces and territories that have not enacted similar legislation.

The epass system has undergone legal, privacy impact, and threat and risk assessments (see Lazarus [Laza02] for further information). In this paper, we focus primarily on the technical (as opposed to physical, procedural, legal, etc.) security and privacy measures. Whereas security requirements are met using familiar techniques and controls, some novel techniques are used to enhance privacy. In particular,

- Individuals are enrolled and identified only to government programs with which they already have a relationship. No identifying information is shared with the central certificate issuer.
- A Meaningless But Unique Number (MBUN) is used as the Distinguished Name (DN) in an individual's public verification certificate. On its own, this certificate contains no identifying information. Within a government program, the MBUN is tied to a Program Identifier (PID) corresponding to the individual.
- Individuals have the flexibility to use more than one epass, allowing them to fine-tune their protection based on their level of privacy concern. A sufficiently private baseline solution is offered for individuals that use a single epass.

As this system and its deployment continue to evolve, this paper captures many features that are part of the current system, though also highlights several research considerations for future applications (which are worth considering as more and more programs offer secure online services). Throughout, features that are not part of the current GOL system are explicitly noted as such.

In Section 2, we describe the management of the epass, from registration to renewal, recovery and revocation. Section 3 examines the use of epass for the Address Change Online (ACO) application provided by the Canada Customs and Revenue Agency (CCRA). We discuss the privacy and security issues regarding the management and use of the epass in Section 4. In Section 5, we provide some concluding remarks.

2 System Design

There are numerous statutory and regulatory requirements upon which a governmental service delivery system must be built. For the purposes of this paper and its security analysis, these requirements can be safely abstracted as the traditional security requirements of confidentiality, integrity, and non-repudiation, as well as ensuring that individual privacy is met. Matched against security requirements is often a requirement for ensuring system usability. Indeed, a more usable system is often used in a more secure manner.

In this section, we describe the design of the PK-based epass[3] system supporting individual authentication. Though the resultant security solution is a combination of legal, policy and technical controls, for the purpose of this paper we focus primarily on technical safeguards and in particular the use and management of public key certificates issued to individuals.

We begin with a high-level system overview, where more detail is given in the subsections below. Individuals register and obtain a single[4], pseudonymous public key certificate (an epass), issued by a central Certification Authority (CA). The identifier within this certificate is a Meaningless But Unique Number (MBUN). The property of uniqueness ensures that no two

[3] The term epass is an abstraction used to identify a generic credential used by individuals to authenticate for government services. Though generally referring to the doubly-encrypted object containing private and public keys and obtained by an individual at login, the term is sometimes used to identify only the public key certificate portion of this object.

[4] At their discretion, an individual may obtain more than one epass.

individuals possess the same MBUN, while lack of meaning ensures that given only an MBUN, no information as to the corresponding individual identity can be gleaned.

Individuals must separately enrol to each government program (one-time at first use) for which they desire electronic government services. This enrolment requires that the individual properly identify him or herself to the program (e.g. based on program-specific shared secret information), thereafter allowing the program to associate an MBUN to the individual's Program Identifier (PID) within that program.[5] The program will continue to use the PID as an index for the user, rather than the MBUN, at least since the MBUN may change. More specifically, the MBUN will be managed by the epass Management System, whereas the government program manages the PID. Hence, the MBUN may change, independently from any actions by the government program. For example, an MBUN associated with a user may change as a result of re-registration after certificate expiry or certificate revocation. Also, an individual may choose to associate a different epass with a government program at any time (and hence, associate a different MBUN with the PID).

The resulting picture is one in which pseudonymous user credentials are securely stored centrally.[6] Within a program, a translation from the MBUN to the PID is maintained in the form of a translation table. Once enrolled, a user can thereafter authenticate with their epass (using public key authentication techniques), after which a translation to the appropriate PID identifies the individual within the context of the government program.

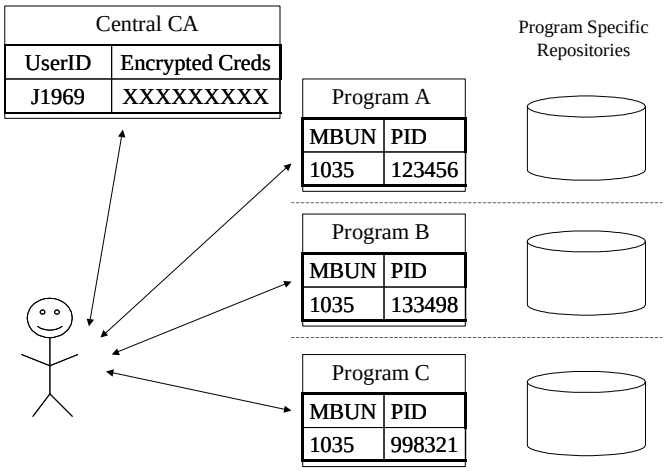

Figure 1: Distinction between MBUN and PID amongst system entities.

Recall that an individual may choose to have multiple credentials, in which case distinct Username and Encrypted Credentials would be maintained, and a separate MBUN may be used for access to each program.

The current system implementation relies on the individual having access to a web browser. A COTS product that implements a downloaded (FIPS 140-1 certified) Java applet provides the necessary additional functionality.

[5] The PID (sometimes called a legacy identifier) is chosen and managed by the government program, independent of the epass.

[6] As described below, the epass solution offers support for roaming users whereby an individual's private and public keys are stored, doubly encrypted in a central repository.

In the following subsections, the detail regarding the registration for an epass and program enrolment, in addition to the epass management, is discussed. The use of an epass for obtaining government services is discussed in Section 3.

2.1 Registration and Enrolment

Several variations are conceivable for the initial registration and enrolment of an individual for access to government services (see Section 2.1.1 for a brief description of some enrolment variations). Below, we expand on a common scenario, and in particular one that is currently offered by the Canada Customs and Revenue Agency (CCRA, formerly Revenue Canada)[7] in support of their Address Change Online (ACO) application.

Before continuing, we highlight an important terminology distinction. An individual will *register* with the central Certification Authority (CA) for an epass. The CA is not aware of any personal information regarding this user; hence the user is not required to identify for the purpose of obtaining an epass. However, when an individual *enrols* with a government program, they will identify themselves to that program. It is with a program that further services may be obtained. To facilitate use of the epass with the government program, at the time of enrolment the program will create an association between the MBUN (from the epass) to the Program ID (PID), where the latter is the index for the user within the realm or context of the program. To avoid unnecessary epass issuance, in the current system only individuals that have properly identified to a program will be redirected to the CA at the epass Management System (MS). And only once they have an epass can they complete program enrolment. Therefore, the process proceeds as three main steps:

1. Individual identification to a government program.
2. Individual registration for an epass.
3. Individual program enrolment (mapping the epass MBUN to the PID).

This process is described in more detail below.

An individual ultimately enrols in a government program by identifying him or herself to the program. This step assumes that the individual already has a relationship with the program, and in particular, has an assigned Program Identifier (PID) (see possible variations to this assumption in Section 2.1.1). As part of the identification process, the individual would typically provide information requested by the program. As a concrete example, individual enrolment within CCRA requires presentment of four pieces of information, namely

1. The individual's date-of-birth,
2. The dollar amount entered by the individual at line 150 of their 2000 or 2001 tax return,
3. A numeric identifier returned to the individual as part of their 2000 or 2001 tax assessment, and
4. The individual's social insurance number (SIN) (roughly equivalent to the US Social Security Number).

After identifying, if the individual doesn't currently have a public key certificate (or perhaps has a certificate but chooses to register for another for use with a program), the individual is transparently redirected to register for the epass. As part of registration, a random, unique MBUN is generated and placed in the certificate for the epass. The transfer to the CA for the purpose of certificate registration does not include transmission of any information that would identify the individual to the CA.

[7] http://www.ccra-adrc.gc.ca/

The steps in the process of program enrolment and certificate registration are depicted and described below.

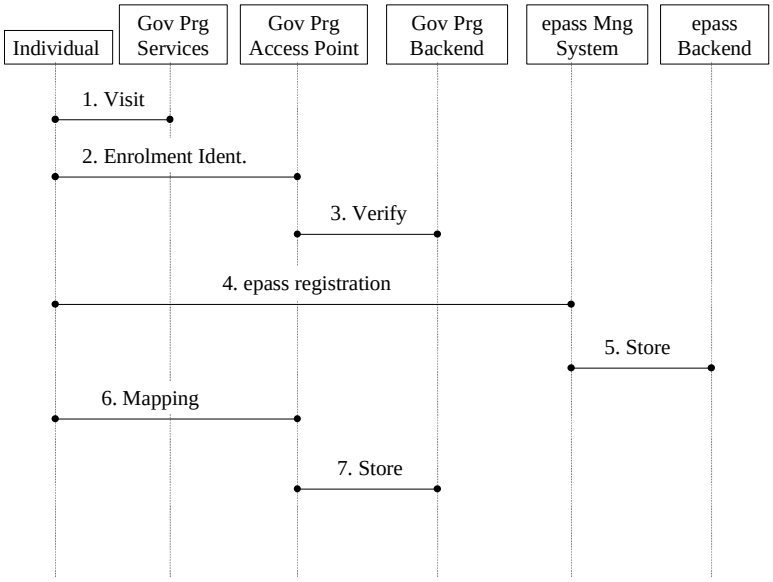

Figure 2: epass registration and government program enrolment

1. An individual visits a Federal Government program web site using their web-browsing client for the purpose of obtaining some secure services. The individual is redirected to the program Access Point (AP).
2. At the AP, there are several options, depending upon whether the user has previously registered for an epass or enrolled with the program:
    a. If the user already has an epass, they either
        i. Login (presuming the user has already enrolled with the program) and securely access program services (see Section 3), or
        ii. Enrol with the program.
    b. If the user does not have an epass, or wish to obtain a new pass for use with this program, they will proceed with enrolment.

    As part of enrolment, the registrant identifies himself or herself by answering questions posed by the program, derived from shared secret information shared between the individual and the program.
3. The registrant-provided answers are validated against the information stored in the program's legacy database.
4. If the registrant has successfully identified them self, a signed cookie is returned to their browser, and they are redirected to the epass Management System (MS) for registration (only if they wish to obtain an epass to use with this program, i.e. they don't yet have an epass, or have one but wish to use a new epass with this program). As part of this process
    a. The individual chooses a user ID and password;
    b. The individual selects recovery challenge questions and answers; and
    c. A private key is generated, with an MBUN similarly generated and assigned by the MS.
    d. The MS, given the public key and MBUN, then creates a public key certificate.
5. The individual's password-protected profile (containing their private and public keys) and recovery questions are further encrypted and stored in the MS repository. Note that no personal information is stored at this repository.

6. The information used to identify the individual as part of program enrolment is signed and encrypted. The MBUN, used as the certificate identifier for the user, is mapped by the program to the individual's Program ID (PID).
7. The program stores the signed registration information and MBUN-PID mapping.

The system supports a roaming client, as the individual's credentials may be stored in a management system repository. This so-called profile for the individual is doubly encrypted, including publicly encrypted for the management system and also encrypted using a key derived from the individual's password.

2.1.1 Variations

As suggested above, there are variations possible for the registration and enrolment process described above. In particular, the differing needs of each department will require that some flexibility exists, though from the point of view of usability, a consistent look-and-feel is maintained as much as possible for the individual. As more and more programs go online, some of these variations may be considered.

No shared secrets with program. There are situations in which direct, on-line program enrolment may not be possible, as the individual does not have a sufficient relationship with the program (e.g. may not have a PID):

1. Individuals that do not have a relationship with any program, e.g. newborn child;
2. Individuals that have a relationship with some programs, but not others, e.g. individuals that have a record of employment but no passport;
3. New citizens (e.g. landed immigrants); or
4. Programs that don't have sufficient shared information with which an individual may be identified as part of enrolment.

For these situations, there are a few options that could be considered to aid in secure enrolment (the second and third can be thought of as special cases of the first):

1. *Guarantor*. A trusted third-party (perhaps another individual, or a dedicated organization) can be used to attest as to the identity of an individual. In particular, the individual will identify to the guarantor (based possibly on some shared secret information, acceptable documentation, or based on some other form of relationship) whereby the guarantor will provide evidence attesting to some attribute for the user. A similar process is currently used for Canadian passport applications.
2. *Shared program enrolment services*. In cases where an individual is able to enrol within program A, but not program B (due to a lack of shared information with program B), the individual might choose to allow program B to request identification information from program A. Such a service would require that the individual's privacy be respected, e.g. that consent is obtained prior to the sharing of such information. The overall privacy implications for designing such a service would also have to be carefully considered.
3. *In-person registration*. Whereas an online registration relies on shared and trustworthy digital information, an in-person registration would allow a registrant to present physical identification (e.g. passport) in order to properly identify them. The cost-benefit of such a system would have to be carefully considered.

Multiple certificates. As mentioned earlier, the registration system is flexible so as to satisfy a wide spectrum of privacy requirements. In the case that an individual uses a single epass, the MBUN is not used as an index by government programs and programs cannot match data based on the MBUN (without the explicit approval of the individual) [Priv85, Priv93]. Alternatively, an

individual may obtain any number of certificates, e.g. a different epass is used for access to each program. This would not require any additional enrolment by the individual, as they must enrol one-time with each program already, though would require they register for and track each epass separately.

Business registration. In addition to individual registration and enrolment, Canada's GOL initiative will support business enrolment and registration. The first program and department to participate is expected to be the Employment Insurance (EI) program, as managed by Human Resources Development Canada (HRDC).[8] This online registration project will support the submission of Records of Employment (ROEs) by Canadian businesses. Business representatives will similarly enrol within the program, whereas a business-designated representative manages control regarding who is able to enrol.

2.2 Certificate Lifecycle Management

In the previous section, epass registration and program enrolment were discussed. In this section, we focus on the certificate lifecycle management operations of renewal, recovery and revocation.

2.2.1 epass Renewal

Presuming that an individual is able recall their password, periodic (and automated) update of an epass represents the dominant certificate management activity. This update would be attempted when a prescribed portion of their certificate lifetime has been reached, as certificate update requests would be generated (transparent to the individual) at each occasion when they login to access government services. Currently, a citizen epass is issued with a five-year lifetime, and updates are attempted once 50% of this lifetime has elapsed.

2.2.2 epass Recovery

In the case that an individual loses control of their password, additional measures must be in place. It is widely recognized that automated recovery processes are key to offering a cost-effective system; else help-desk costs can dominate (e.g. based on lost passwords). As such, as part of the epass registration process, individuals provide a list of challenge questions and corresponding answers. As a result of a successful recovery, a new verification certificate is generated, containing the same MBUN.[9]

At a later time (such as when they no longer remember their password), the user will be prompted with the question and provide the appropriate answer for each. The current question and answer model involves providing a fixed list of questions to the user, with the user selecting five and selecting from a list of fixed answers. (See [Just03] for alternative recovery question models.)

A *graduated lockout* mechanism could be used to mitigate against exhaustive attacks attempting to maliciously recover a user. In such a scheme, an individual could be temporarily locked out for several rounds, each round triggered by a number of consecutive recovery failures. Looking ahead, other options for recovery are possible, including free-form answer submission and possibly voice biometrics. In the latter case, a user could be prompted to record a short statement when registering. At recovery, the individual could be asked to repeat the statement. Of course, this would unfortunately require a change in service channel (from computer to phone).

[8] http://www.hrdc-drhc.gc.ca/

[9] Hence, this is more properly viewed as *account recovery*, rather than *epass recovery*.

If an individual is unable to recall their username, then the recovery questions cannot be retrieved and posed to the user. Since the CA does not have any other context in which they could re-register the user, this could require re-registration and further, re-enrolment in each program.[10] However, one could envision making use of the reverse PID-MBUN mapping (stored at a program) to aid recovery. Subject to the user's consent, it would be technically possible to allow the user to subsequently identify themselves to a program (that both has a record of the PIN-MBUN mapping and has a suitably secure set of shared secrets that may be used to identify the user) in which case the MBUN could then be used to proceed with the aforementioned recovery process through the CA.

2.2.3 epass Revocation

Given the distinction between registration and enrolment, we can consider two instances of "revocation." As related to public keys, an individual's epass can certainly be revoked, in which case the individual would no longer be able to use the revoked certificate to authenticate to any government program. There seem to be few reasons for which such a revocation would occur (e.g. death). Currently, the certificate owner is able to revoke their certificate by correctly responding to their recovery questions.

In addition, per-program "revocation" can occur by "de-activating" the PID-MBUN mapping so that for that program a user cannot be successfully authenticated. Depending on the reason for revocation in both cases, programs may still want to be able to verify digital signatures so that a history of the PIN-MBUN mapping is maintained.

3 System Use

The first and only application that currently makes use of the epass system is the Address Change Online (ACO) application provided by the Canadian Customs and Revenue Agency (CCRA – formerly Revenue Canada). The steps performed during the execution of this application are depicted and described below.

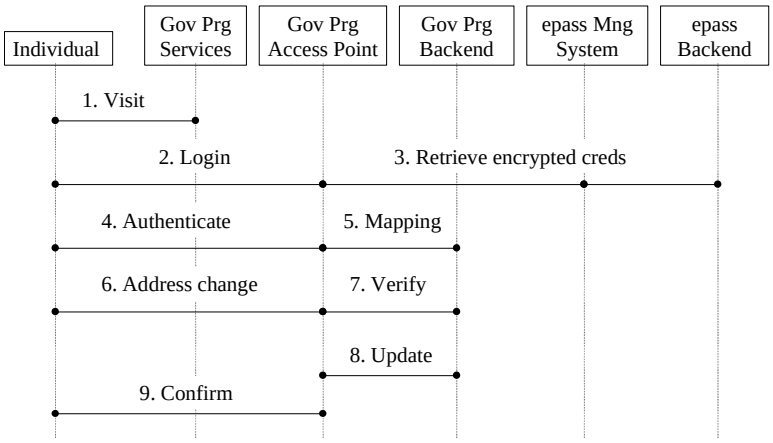

Figure 3: Program login and execution of address change application

[10] Other systems, such as Yahoo!, support the recovery of the username by matching to the user's email and sending an email, containing the username, to the user's email address-of-record. However, with the GOL solution, the CA is not privy to any personal information that might identify the user (such as an email address). Hence this solution cannot be used.

1. The individual visits a Federal Government program web site using their web-browsing client for the purpose of obtaining some secure services. The individual is redirected to the program Access Point (AP).
2. At the AP, there are several options, depending upon whether the user has previously registered for an epass or enrolled with the program:
    a. If the user already has an epass, they either
        i. Login (presuming the user has already enrolled with the program) and securely access program services (see Section 3), or
        ii. Enrol with the program.
    b. If the user does not have an epass, or wish to obtain a new pass for use with this program, they will proceed with enrolment.

    Assuming that the individual already has an epass and has enrolled with the program, at this point the individual will enter their user ID and password for retrieval of the epass.
3. Through a secure session, the individual's encrypted credentials are retrieved and returned to the individual.
4. The individual's password is used to decrypt their encrypted credentials at their browser and the individual uses their epass to authenticate to the government program. A signed cookie is returned to the browser if the authentication is successful.
5. Upon verification of the user's authentication attempt, the MBUN from the epass is used to map to the appropriate program ID (PID).
6. As part of the ACO application the user may browse their address information. When they choose to update address information, the appropriate form is returned to the user. The user completes the form then it is digitally signed and encrypted and returned to the web server.
7. The server validates the signed message, including revocation checking, and confirms that the user is authorized to make the change for the address information corresponding to the given PID.
8. The address change is made in the government program database and the signed and encrypted confirmation form is also stored.

The ACO application is particularly interesting from a privacy point of view as it represents an application whereby an individual is able to both view and correct their personal information – in this case, their address information. As with epass registration, the retrieval of the epass is achieved in an anonymous way, so that the central epass system does not know any personal information regarding the holder of the epass. Authentication and signing with the epass credentials are performed so that only the program in question is able to successfully identify the individual (through the MBUN-PID mapping).

4 Discussion

The resultant epass system is a combination of technical, legal, policy and procedural controls. As in the previous sections, in the discussion below we focus on the technical aspects, though will cite other controls as appropriate.

It is likely that a variety of other technical solutions would satisfy the security and privacy requirements for individuals, and likely more or less proficient in different areas. The primary advantage of a public key-based system is that it allows storage of persistently authenticated data with a digital signature. The management support (though not necessarily unique to PKI) is also advantageous, supporting relatively convenient automated renewal, recovery and revocation.

The advantageous properties, relating to the use of pseudonymous certificates, are discussed primarily in Section 4.1 below. The alternative of veronymous[11] certificates seems unnecessary, and would likely result in either a set of sufficiently identifying information being stored in the certificate (in order to satisfy the differing identification requirements within the context of each program) or a smaller set of information being used similar to a universal identifier. Alternative solutions, using attribute certificates for example, might also produce a sufficient solution. With the current epass solution, programs maintain their current individual identifiers.

4.1 Privacy

The familiar set of privacy principles (e.g. FIPs [FIPs00], OECD [OECD80]) cover a wide range of issues and a broader set of potential services than for the epass functionality discussed in this paper. In the discussion below, we enumerate some of these principles as relevant, as well as some additional principles that seem appropriate to analyzing an authentication system (some of which were stated in the results of the Privacy Impact Assessment performed on the epass system [Priv02]).

We focus on the attributes relevant to the epass as a pseudonymous certificate below. Note that signed messages are also encrypted so that information about an individual, as contained in a signed message, is only visible to the intended recipient.

1. *Anonymity/Pseudonymity*. Outside of a program in which an epass-holder has enrolled, the public key certificate reveals no information about an individual; to an observer, the individual is essentially anonymous. Within a program that the individual has enrolled, the MBUN acts as a pseudonym, allowing proper identification of the individual once linked to the PID.
2. *Choice*. Individuals can choose to register for a single epass, or alternatively, a separate epass for use with each program. With a single epass, a comfortable level of privacy is provided since (i) programs will continue to use the PID as their primary index (recall, the MBUN can change, outside of the control of the program), and (ii) legislation [Priv85] and policy [Priv93] restrict the sharing of information between government programs. While more than one epass might allow a more comfortable separation of program information for some, it requires registration for and tracking of more than one epass (e.g. to determine which epass is used for access the services of a particular program). More generally, further choice is offered by the maintenance of multiple channel support (e.g. telephone), in which case individuals are not obliged to obtain an epass.
3. *Inference*. Compared to an alternative solution in which the name of a particular issuer and subject are cited in a certificate (e.g. if "John Doe" were issued a certificate from the "Department of Corrections"), no similar inference may be drawn with an epass as the Government of Canada issues certificates.
4. *De-centralized information.* The only centralized entity involved in this system (the CA) does not possess, nor is ever given, personal information.
5. *Data separation*. With a single epass, programs index on a program ID (PID). Privacy legislation and policy ensure that new use of information is not performed without user consent [Priv85, Priv93]. As a technical option, users may choose to have more than one epass.

[11] A term coined by Carlisle Adams. Distinguishing from anonym ("no name") and pseudonym ("false name"), a veronym refers to a "true name." A veronymous certificate refers to a certificate that contains a veronym.

6. *Access to personal information*. As a particular example, specific to the ACO application, users may view and correct their address information within a program.

An additional concern related to privacy is that of identity theft. Such theft could allow fraudulent program enrolment. For this reason, each program ensures that a suitably high level of assurance is supported when identifying individuals. In addition, the design philosophy, whereby individuals must enrol separately to each program, mitigates the scope for potential fraudulent enrolments.

4.2 Security

Beyond system privacy, system security can be qualified. We do so by discussing the confidentiality, integrity and non-repudiation as achieved by use of an epass. In general, these properties are achieved using a combination of familiar technical security controls, in addition to other physical and procedural controls; we highlight some relevant technical controls below.

The individual interacts with programs through secure sessions. Therefore, we achieve confidentiality and integrity by reliance on this secure channel (i.e. SSL). In addition, persistent encryption and signing beyond the session may also be achieved depending the needs of a particular government program. For persistent encryption, a client is able to encrypt using the encryption certificate for a back-end server. Similarly, a client may persistently sign information, in support of authentication and non-repudiation.

Elaborating on identification, certificates are pseudonymous whereby a link is maintained at each program that matches the MBUN to the corresponding PID. In addition, at enrolment, the individual signs the evidence of enrolment. This supports proper identification (as would be necessary for further authorization requirements) and allows evidence to be contributed for non-repudiation. For an observer outside of a government program, the verification certificate attached to the signed (and encrypted) data is not readily attributable to any particular individual.

5 Concluding Remarks

In this paper, we've described and analyzed the system in which individuals may obtain an epass for accessing secure services online. Currently, this system is demonstrated through a single address change application. However, many more applications will be added as more and more departments similarly offer secure services.

This use of public keys for an epass is similar to modifications as suggested by Ellison [Elli02]. In our case, the use of some certificate identifier (other than the public key or some other value dependent upon the public key, such as its hash) allows for the same identifier to be used even as a result of epass renewal or recovery (in which case, a new public key is generated).

Looking ahead, there will be numerous opportunities for individuals to operate within multiple jurisdictions with their epass. Already, cross-certification has begun between the federal and provincial levels. And as this paper is written, the Canadian Federal Government is working towards cross-certification with the US Federal Bridge CA.

Acknowledgements

Thanks to several Treasury Board colleagues for their helpful comments, including Rick Brouzes, Michael de Rosenroll, Rhonda Lazarus, Wendy Stewart, Brenda Watkins and John Weigelt. Thanks also to the workshop referees for their comments.

6 References

[Elli02] Carl Ellison, "Improvements on Conventional PKI Wisdom," in *Proceedings of the 1st Annual PKI Research Workshop*, April 2002. http://www.cs.dartmouth.edu/~pki02/

[Laza02] Rhonda Lazarus, "Government of Canada's Legal and Policy Framework for Government On-Line," presented at *Canadian IT Law Association Conference*, October 2002. http://www.cio-dpi.gc.ca/pki-icp/issuesactiv/frame/frametb_e.asp

[FIPs00] Federal Trade Commission, "Privacy Online: Fair Information Practices in the Electronic Marketplace," May 2000 http://www.ftc.gov/reports/privacy2000/privacy2000.pdf

[Just03] Mike Just, "Designing Secure Yet Usable Credential Recovery Systems Using Challenge Questions", *Workshop on Human-Computer Interaction and Security Systems*, 6 April 2003. http://www.andrewpatrick.ca/CHI2003/HCISEC/index.html

[OECD80] Organization for Economic Co-operation and Development (OECD), "Guidelines on the Protection of Privacy and Transborder Flows of Personal Data", 1980.

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

[Priv85] Department of Justice - Canada, *Privacy Act*, 1985. http://laws.justice.gc.ca/en/P-21/index.html

[Priv93] Treasury Board of Canada, Secretariat, *Privacy and Data Protection Policy, Chapter 2-5 – Data Matching*, Dec 1, 1993. http://www.tbs-sct.gc.ca/pubs_pol/gospubs/TBM_128/CHAP2_5_e.asp

[Priv02] Treasury Board of Canada, Secretariat, *Privacy Impact Assessment Policy and Guidelines*, May 2002. http://www.tbs-sct.gc.ca/pubs_pol/ciopubs/pia-pefr/siglist_e.asp

FREEICP.ORG: FREE TRUSTED CERTIFICATES BY COMBINING THE X.509 HIERARCHY AND THE PGP WEB OF TRUST THROUGH A COLLABORATIVE TRUST SCORING SYSTEM

Marco Antônio Carnut (kiko@tempest.com.br)
Tempest Security Technologies
Centro de Estudos e Sistemas Avançados do Recife - CESAR
Universidade Federal de Pernambuco – CIn/UFPE

Evandro Curvelo Hora (evandro@tempest.com.br)
Tempest Security Technologies
Centro de Estudos e Sistemas Avançados do Recife - CESAR
Universidade Federal de Pernambuco – CIn/UFPE
Universidade Federal de Sergipe – DCCE/UFS

Cristiano Lincoln Mattos (lincoln@tempest.com.br)
Tempest Security Technologies
Centro de Estudos e Sistemas Avançados do Recife - CESAR
Universidade Federal de Pernambuco – CIn/UFPE

Fabio Q. B. da Silva (fabio@cin.ufpe.br)
Universidade Federal de Pernambuco – CIn/UFPE

ABSTRACT

This paper describes a CA hierarchy that mimicks PGP's web-of-trust model using a collaborative web-based trust scoring system to provide free client digital certifcates with strong identity guarantees. Entry-Level CAs that approve temporary short-lived certificates immediately may replace traditional password-based web registration systems; identity guarantees may be added later by passing several qualification rounds in a trust manager web application. When the user exceeds the minimum qualification criteria, he is granted a Verified Identity-class certificate. The system encourages users to tie their digital IDs with their real world IDs, making them more institutionally acceptable and sometimes automatically verifiable; it is argued how this can also provide means of mitigating and managing identity disputes. Experiences gathered from implementing both a CA hierarchy and a relying party web application based on these principles are also presented.

.

1 INTRODUCTION

The hierarchical X.509 PKI [13, 12] and the PGP web of trust [25, 6, 3] have historically been presented as inherently antagonic approaches [2, 8] and extensive discussion has been published about their limitations and unsuitability for global e-commerce [24, 19, 10, 9]. Notwithstanding, these were the only ones to achieve a reasonable level of popularity, as measured by the widespread availability of implementations (PGP [27], GPG [31] and numerous email client plugins), toolkits (OpenSSL [28], Jonah [33], Cryptlib [34], etc.), web server software (Apache [30] + mod_ssl [29]) and clients (IE, Netscape, Mozilla, Opera, etc).

This paper proposes a way to take the best of both worlds, showing one possible way to endow an X.509 hierarchy with a collaborative trust system somewhat like the PGP's web of trust model, but with considerable advantages. In fact, we wanted that the two PKIs user bases could reinforce each other, making our solution also a kind of bridge-CA.

Specifically, we wanted to provide a way for individuals to be identified by means of SSL/TLS client certificates for authentication purposes in web-based applications, but without having to pay for their identities to be verified like in commercial CAs. The solution has been present in PGP since its inception: users vouch for other users' identities by signing their keys, building a distributed, collaborative web of trust [15]. X.509 was not quite designed to support this, so we built a web-based CA application that allows users to introduce each other, assigning numeric scores to the amount of certainty users grant each other – in fact, a generalization of PGP's own trust scores, but with a few novelties: a mechanism for tying their virtual identities to real world identifiers (so as to make them more "institutionally acceptable"); a way to perform many simple identity validation checks automatically; and a method to detect and manage identity disputes, either malicious or not.

We also tried to maintain a few key design principles: the whole CA infrastructure should be implementable with common open-source software (Apache, mod_ssl) and should be usable with the standard popular web browsers (IE, Nescape, etc.). Moreover, they should be made as simple as possible, up to the point of rivalling with common "email & password" web registration schemes.

The rest of this paper is organized as follows: section 2 describes the CA infrastructure at considerable length: the Entry-Level and Verified Identity family of Certificate Authorities, the Trust Manager and the trust scores, along with many experiences from the actual reference implementation. Section 3 details the combination of automatic and human-assisted identity validation procedures used by the Trust Manager to ascertain the users and hosts identities. Particular attention is given to the resilience to misbehavior with a description of the identity contention management scheme. It is also argued that these metrics adhere to good design principles proposed in the literature. We could not judge how well our system would perform without trying it in a real application; section 4 describes our initial experience in adapting an existing application to support our mixed-PKI infrastructure. Section 5 presents conclusions and future work directions.

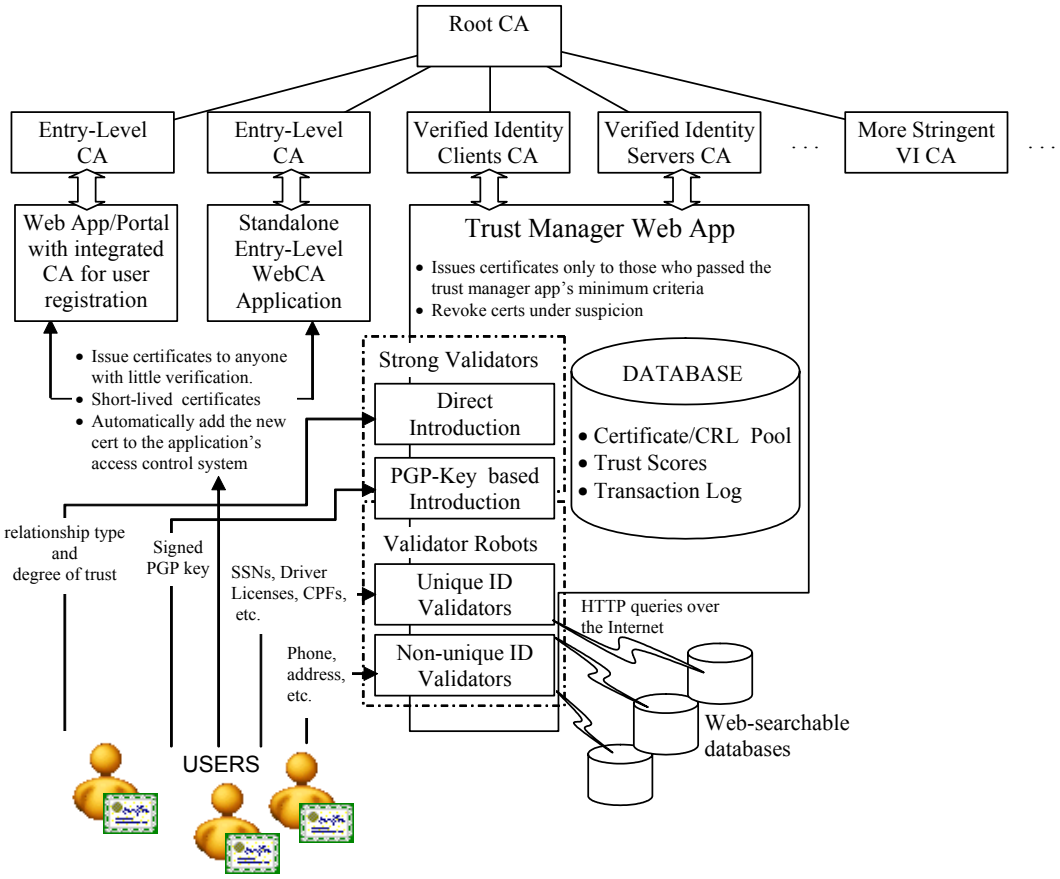

Figure 1: Overall system architecture: above, the X.509 treelike root CA → intermediate CAs key heirarchy. The two main CAs are the Entry-Level and the Verified Identity, associated to their respective web sites. The first issues certificates to nearly any user, just to allow them to log on the trust manager web application. Web portals might have an integrated EL CA as its user registration system. Users are allowed to issue certificates under the Verified Identity CA only when they meet a minimum set of scores. The way to increase them is by passing through several kinds of validators: the validators robots check the users' personal information on web databases; and the strong validators are based on user-to-user introduction. The VI CA also has a PGP keypair and adds his signature to users with PGP keys (when they qualify); and accepts signed PGP keys as strong-validation introductions. This leverages the userbases of both PKIs, helping them to reinforce each other.

2 SYSTEM ARCHITECTURE

2.1 CA and Key Hierarchy

We start by proposing a fairly standard CA hierarchy: a root CA which certifies two families of intermediate CAs:

- **The Entry-Level (EL) CA family:** these CA applications generate certificates online to any user that requests one, with just minimal validation, such as complying with a simple naming policy, avoiding duplicates and challenging the validity of the email address by replying to it. Its sole purpose is to put a valid, working, fully functional digital certificate into the users' client applications – most likely, their web browsers – immediately and for free. These certificates have short validity periods compared with the more trusted ones – two or three months seem sensible. They should be accepted only for testing or initial enrollment in web applications.
- **The Verified Identity (VI) CA:** this CA issues digital certificates when users meet some specific credibility and trustability scoring. These certificates have a larger validity period, something like six to twelve months. Actually, there could be several such CAs, each with successively more stringent scoring requirements. Large-scale production applications should require these certificates for the bulk of their functionality; the applications should "insist" that the users "upgrade" to a VI certificate as soon as possible.

The VI CAs would have both X.509 certificates/private keys and PGP keypairs, so they could act as cross-certifiers. The idea is to leverage each PKI's user base to reinforce each other and foster wider adoption.

2.2 The Entry-Level Certification Authorities

We wanted to make users able to generate a new SSL client certificate just as easily as PGP users can

generate their key pairs. However, most web browsers don't have provisions for properly signing Certificate Signing Requets; and even if they did, we would have to be part of a hierarchy anyway. So, we need a CA; we call it an Entry-Level CA.

We use the term "Entry-Level" to suggest a certificate with no identity guarantees – just like PGP keypairs –, in analogy with "temporary" membership or airline frequent flyer cards. The user should expect that he will be required to change it for a "definitive" one. Applications should grant minimum "guest-like" privileges to accesses made with this certificate.

To maintain parity with PGP, users are identified by their email addresses and a nickname (possibly, but not necessarily, their real names). In our prototype implementation we followed a Google-like UI simplicity principle: the user types in his name and email in a *single form* field and gets the certificate installed in the very next screen.

This ideal has been implemented with reasonable success on Netscape and similar browsers (Mozilla, Opera), as shown in figure 2: from the moment the user hits the submit button to the point the certificate gets installed, those browsers add just a few simple steps to ask or set the private key container's passphrase. Internet Explorer, however, proved much more intimidating: figure 3 shows that even when it's not necessary to update the ActiveX control that handles key/CSR generation, the process may take up to 12 steps with scary messages about scripting violations; and the user interface almost compels the user to store his certificate with no passphrase.

Another catch is that the user has to have already installed our root CA's certificate. In our intranet setting, this is part of our customized OS installation procedure, so our users don't have to perform this step. In other environments, however, this will be needed; although the process is not complex, some browsers present a multi-step wizard with many choices that non-technical users often misunderstand.

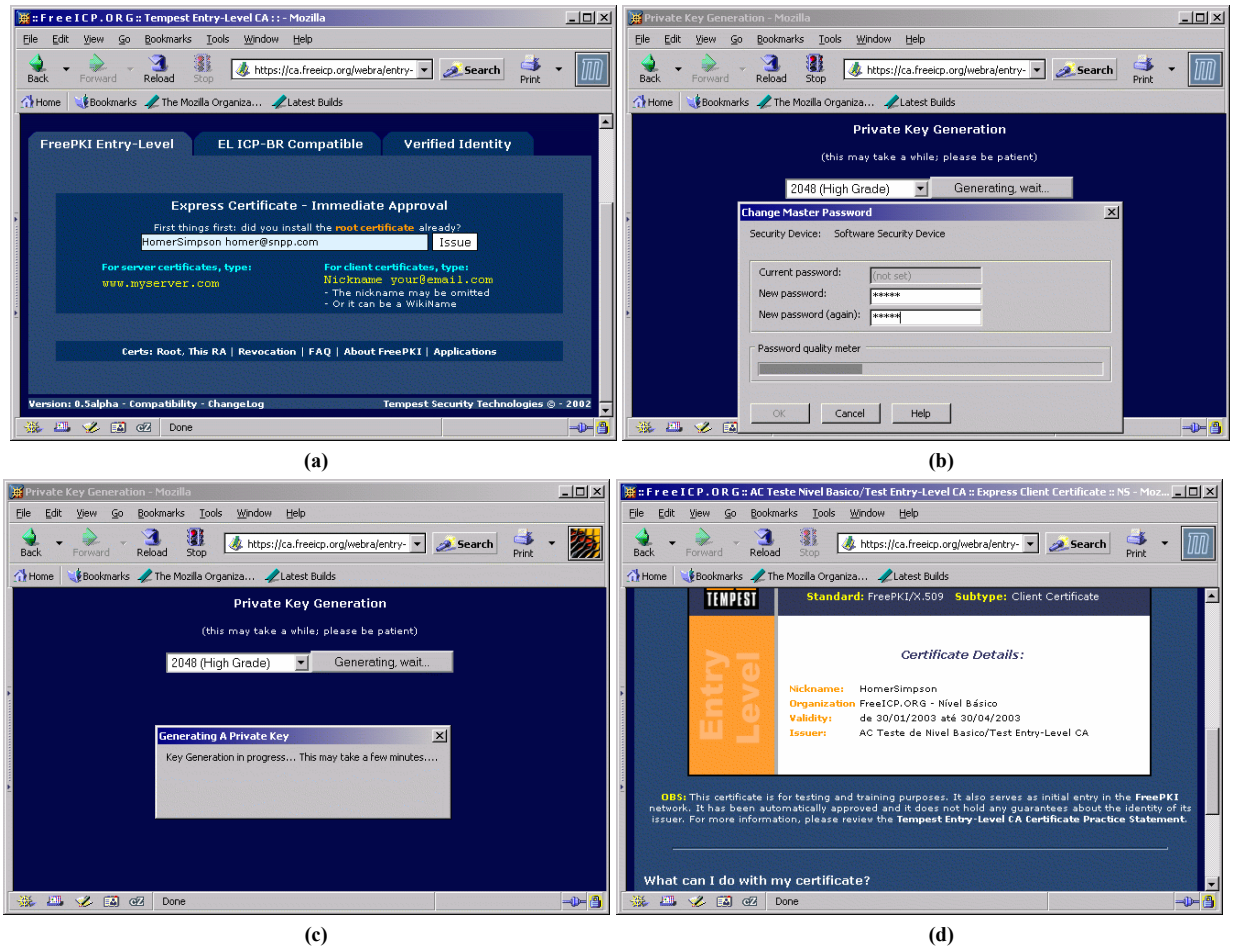

Figure 2: Mozilla & Netscape-derived browsers are more amenable to the express certificate concept: the whole process can be done in 4 steps. In **(a)**, the user types his name, email address and hits the Issue button. In **(b)**, the user is asked its container passphrase, or, in this case, to set one up. After that the user needs to perform no further action but to wait the key generation to finish **(c)** and the installation to complete **(d)**. In Opera, the dialogs look different but the process is essentially the same. All that supposes that the user has already installed the root certificate, which is conducted by a 1-6 step "wizard-like" sequence of dialog boxes, depending on the exact browser used. Mozilla, shown here, is the simplest.

Users also often don't get the point of the fingerprint verification and in many cases proceed without actually performing it rigorously. We can only hope to gain enough popularity in the future to be able to include our final root CA certificate in upcoming versions of common web browsers.

Figure 3: Express certificate generation in IE takes at least 11 steps, six of which shown here: in **(a)** the user types his username and email. In **(b)**, IE warns the novice with a somewhat needless obvious question. In **(c)** and **(d)**, the uncommonly well informed and disciplined user sets the security level to high; if he didn't explicitly ask for the high security setting (which the API doesn't allow the CA to set), the certificate would be stored without a passphrase. In **(e)**, the user finally gets to set his passphrase. In **(f)**, after confirming two levels of dialog boxes, there user receives another scary message before getting his certificate installed. All this supposes that the user has already installed the root certificate (a 9-step wizard) and has a recently patched (pos-Q323172) version of Internet Explorer (2 more steps which may fail silently if the computer is configured with policies restricting software installation); these were omitted for sake of brevity. All this ends up making certificate generation a frustrating process that fails in more than 60% of the attempts.

After having the certificate installed, a confirmation email is sent to the address the user specified. It clearly informs:

- The website name and the exact URL he accessed to perform the enrollment;
- An URL to revoke this certificate in a single click – for instance, in case the user feels the certificate was issued by someone other than himself;
- A succint description of the certificate's purpose – often, its sole purpose is to access some web application; in some EL CAs, we redirect the user to a directory of services that require these certificates;
- The fact that EL certificates are to be understood as "temporary, limited access", having a somewhat brief validity period (a few days or weeks); and the fact that the user can apply for a Verified Identity Certificate which grants greater validity and, possibly, more access privileges; an URL where the user can learn more about the certificate classes, CPSs, etc., is also given.

The loose identity tie, based mostly on the email address, makes the EL certificate somewhat like Verisign's Class 1 certificates [22]. Their process, however, require confirmation that the user controls that email address: they send an email with an URL that the user must access to pick up his/her certificate. Since it's very easy to create a valid email address in one of the many free webmail services, this doesn't add much security. Since commercial CAs usually take several hours to issue the certificate and send the email inviting the user to pick it up, this also sets the stage for a very common mistake: trying to pick it up in a different browser or computer from where the user requested it.

Every now and then some users lose the email with the revocation URL. In these cases, we tell them to go to the EL CA enrollment page just as if they wanted a new certificate. It detects that the supplied email addresses already have a valid certificate associated with them and offers the users three choices:

- **Revocation**: the EL Web CA application resends the email with the revocation URL; if and when the user wants to revoke the certificate, he accesses the URL.
- **Reissuing:** the EL Web CA sends a email with a special URL that revokes the previous certificate and issues a new one in a single step.
- **Do nothing:** leave things as they are.

There is considerable debate about whether revocation is a good idea or even needed at all in PKI systems [18, 16]. In light of this "revoke if it wasn't you who requested it" philosophy, along with the need to reissue certificates often due to the short certificate validity, revocation seems well suited, even though most relying-party applications neither correctly process CRLs nor support OCSP [26] or the like. We decided that all our "FreeICP.ORG-compliant" applications should include full support to a policy-based revocation verification system.

Another important point is the naming policy. It follows the following principles:

- **Globally Unique DNs**: The certificate holder's Distinguished Name in the Subject field should identify only his email address (with the Email OID), his name (in the Common Name OID) and the name of the Entry-Level CA who issued the certificate in a OU field. This makes DNs globally unique, preventing name clashes in case some user tries to issue certificates under more than one EL CA. Thus, the EL CA does not need to check elsewhere to see if this DN has already been taken. This also simplifies building associated directory services, like a global LDAP database.
- **Only one certificate per email address:** otherwise, the identity guarantee would be even slacker and the reissuing/revocation detection wouldn't work.
- **Server Certificates:** If a user supplies a valid DNS name as his name, the EL CA may issue a server certificate instead. It does need to do any kind of checks to see if the address exists. Several server certificates may be issued for the same contact email address (presumably, the servers' administrator). These certificates, however, are to be used for testing purposes only, since they bear no identity guarantees, and, as we shall see, the process for generating Verified Identity server certificates does not need them. The EL CA has the option, according to its own policies, of *not* issuing server certificates at all.
- **Identity privacy:** nickname and email offer little to correlate the user with his real world persona and sound very familiar to oldtime PGP users. This is in stark contrast with several other certification services, which require lots of personal data *in advance* to perform identity validation – an extreme example being the Brazilian National PKI, which not only demands the user's ID in the four most proeminent national registries [4], but includes them in the certificate, making them easy prey for spammers and identity thieves. In our system, personal data is required only when the user wants to get his identity validated, as shown in section 3.1 .

The deliberate bias towards user friendliness instead of "security" (as represented by identity guarantees) may be regarded as distateful by PKI purists. In fact, the scheme proposed above provides only slightly more

features than the PGP PKI (because of the much clearer revocation process) and the same level of identity validation – nearly none at all.

We argue, however, that all these usability trade-offs are fundamental to get user acceptance – both the end-user and the web application developers and administrators. To the best of our knowledge, there are no comprehensive studies on how usability problems affect the X.509 PKI – despite profuse folkloric horror user support stories within CA managers and PKI practitioners communities. However, [23] explains why PGP, widely thought as being "user friendly" because its Windows versions have a decent GUI, is much more non-intuitive and less usable than many of its enthusiasts would like to admit. Many of its results are very well applicable to the X.509 arena and have inspired our design for extreme simplicity.

Admittedly, even this simplicity cannot solve many *compliance defects* [5] inherent in PKI systems, like the impossibility to enforce good passphrases to protect the private key (since its generated by the client software; as shown in Figure 3, Internet Explorer, in particular, makes it upsettingly easy to have a private key with no passphrase at all; both Netscape and IE don't provide a way to require a minimum passphrase complexity) or to securely distribute the root CA's certificate (all of our EL CA's pages invite the user to reinstall the root CA certificate and check their fingerprints). However, yet again we are trusting the client software and user to do the "Right Thing". It is hoped that future versions of these clients may rectify these deficiencies.

Notwithstanding, the EL CA's Certificate Practice Statement must make it very clear what "Entry-Level certificate" means: no identity guarantees, good for testing, learning and initial entry in the trust system; and that the user's ultimate goal should be to upgrade the Entry-Level certificate to a Verified Identity one.

2.3 The Trust Manager Application and the Verified Identity CA

The PGP PKI adds in-band identity guarantess by allowing public keys to bear (possibly many) signatures from other users. In the X.509 PKI we can't to that because certficates can't have more than one signature; and end entities can't sign certificates – only CAs can. Thus, the natural solution is to make a CA that issues the user another certificate, which we call a "Verified Identity" certificate, when he passes some set of identity verification criteria.

Along with this Verified Identity CA there is the *trust manager* web application (TMWA, for short). It requires SSL client certificate authentication, accepting any user whose certificate was issued by both the EL and VI CAs. For each of them, the application would store their certificates, personal and contact data that the user voluntarily made available for purposes of identity checking and three *trust scores*:

- **Credibility score**: measures how certain we are that this individual is who s/he claims to be. It will be calculated as a weighted average of several *validators*, described below. It is analogous to PGP Key's "validity" rating, but much more granular – PGP's validity can be only "valid" or "invalid", while our credibility score is an integer number.
- **Introducer score:** indicates how trustable this user is when attesting or repudiating other users' identities. EL-certified users cannot have introducer points; only VI-class users may introduce other users. It is akin to PGP's "trust" rating, but, again, much more granular.
- **Suspicion score:** keeps track of how much this user is involved in identity contention with someone else. Users under suspicion (i.e., with non-zero suspicion scores) cannot have certificates issued or reissued under the VI CAs; besides, their introducer power is suspended. Notwithstanding, they can accumulate credibility points normally. If his credibility score exceeds his suspicion points, his privileges will be granted back and his suspicion points will be reset to zero. If a user spends too much time (say, a month) under suspicion, his account is deleted ("garbage collected", in our jargon) after being sent an email warning a few days before.

It is instructive to compare this scheme with other proposals like Thawte's Freemail Web-of-Trust program [21]: in their system, there is only one score that handles both the user's credibility and its experience/reliability as an introducer (which are called "notaries"). There is no suspicion management, since each notary is required to meet in person with any individual he introduces and it is seems to be thought that this makes the system immune to disputes.

Each VI CA would have an "eligibility criteria", based on the trust scores, metrics from the trust graph and, possibly, other criteria (e.g., requiring a specialized client, more secure than the mainstream web browsers). When some EL certificate user meets or exceeds these criteria, the VI CA would send an email inviting him to issue a VI certificate (this most likely requires generating a new private key, since most client software requires a one-to-one mapping between a certificate and a private key).

3 VALIDATORS

Validators are procedures executed by the TMWA for verifying the identity a certificate holder. We propose the following kinds of validators:

- **Automatic validators:** scripts/robots that verify some of the users personal data through automated

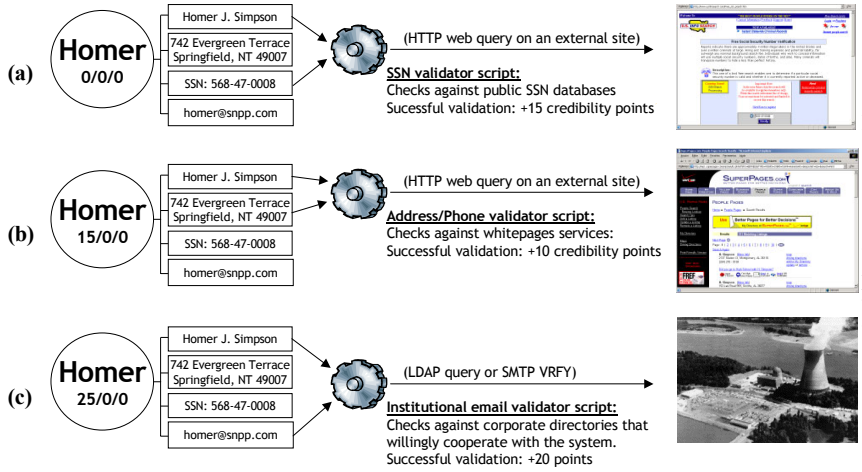

Figure 4: Automatic ("weak") validators in action: Homer Simpson enrolls in the TMWA and starts with no credibility, introduction or suspicion points. After having posted some verifiable personal information in the TMWA database, the system runs several scripts to confirm his claims: in (a), the TMWA queries an external web site (say, usinfosearch.com) to validate his name and SSN, earning him 15 points. In (b), the TMWA queries another website (in the example, superpages.com) to verify his address, earning 10 more points. In (c), it queries his empolyer's LDAP database and/or mail server. Homer gets out of the weak validator process with 45 points (not shown in the picture), with shouldn't be enough to grant him a VI certificate.

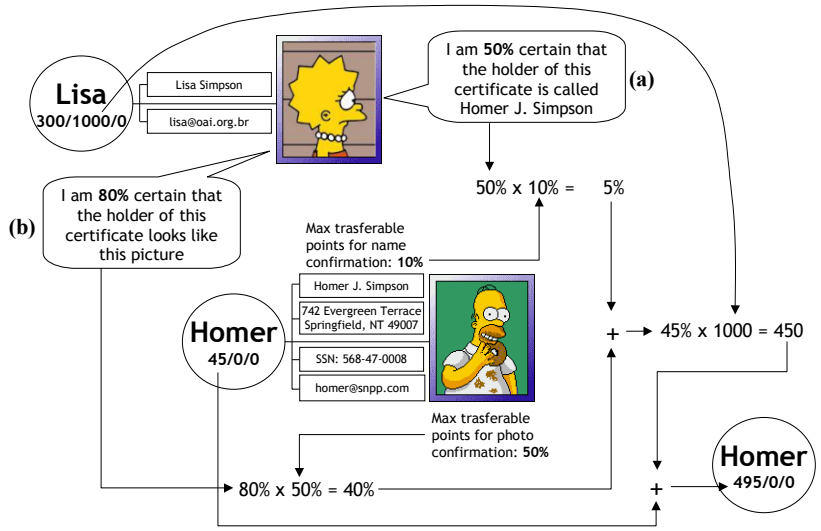

Figure 5: Strong validator process: A case of strong validation through direct introduction: Lisa is a highly trusted introducer, with 1000 introducer points. In (a), she transfers 50% of certainty that the certificate's owner name is Homer J. Simpson. The TMWA presets this validaton as yielding at most 10% of the introducer's trust score, so it transfers only 5% of Lisa's 1000 intro points to Homer. In (b), Lisa also attests with 80% certainty that this is Homer's picture, which, multiplied by the TMWA built-in limit of 50% for photo validations, grant him 40% of Lisa's 1000 points. He finishes this accredidation session with 45 points from the weak validators and 450 points from the strong validators. If this exceeds some VI CA minimum thresholds, it will grant him a VI certificate.

queries on public websites. For instance, checking names and addresses in whitepage services such as knowx.com or public government services (section 3.1 presents more specific examples). Users passing on these validators would receive a small amount of credibility points. It has to be small because, since it is based on public data, it's rather easily spoofed – because of that, the automatic validators are also called "weak" validators. However, they fulfill an important role: tying the certificate holder with a verifiable identity in the real world, as maintained by other independent sources. If anyone wants to spoof anyone else, they would spoof someone who probably exists and may eventually expose the spoof and/or dispute with the spoofer.

An important design principle is that they should not, insofar as possible, require on-site CA operators; they should be performed automatically, either by querying an online public Web database or being driven by remote users' input. They are to be triggered by the client users themselves, by accessing the proper web pages in the TMWA. One of their main functions is to allow users who already possess an entry-level certificate to increase their scores, up to the point for qualifying to get a VI certificate issued – without having to go in person or send paper credentials over snail mail to the CA.

- **User-driven Introduction:** the traditional way of cross-certification through trusted introducers – the user gets someone else *already with a high introducer trust rating* to vouch for his identity. The introducer would access his personal account in the Web CA and fill in a form saying that he has *x* percent certainty that the newcomer is who he says he is. This number would be multiplied by his introducer trust score and an attenuation factor

dependent on the exact identifier being validated. For instance, photographs are harder to fake than names or email addresses, so they should grant more points. The final result is added to the newcomer's credibility score.

All that means that the web-of-trust exists on the Web CA application's database as a set of trust scores; a graph of introducer-introductee relationships; and a log of validation procedures followed by each user. This last item is especially interesting for debugging and auditing, for it allows us to reconstruct the user's history and justify why the system has given him the score he has.

Any given user is capable of, at any time, check his scores and be informed of what steps to take in order to increase them. When the scores of a particular user grow beyond a specified threshold, he should be issued a certificate under the Verified Identity CA. That would mean that the user passed enough challenges and validations for the VI CA to be sure enough of his identity to issue him a certificate.

3.1 Automatic Validators ("Validator Robots")

Automatic validators provide new users a way to gain a small but significant initial credibility quickly, online, without having to ask other people to vouch for them.

It works like this: a new user would log on the TMWA using his EL certificate/private key pair and supply certain kinds of online-verifiable personal data, such as postal address, phone numbers, IDs in public services – Social Security Numbers, driver license numbers, etc.

The user would not be *required* to enroll his personal data in the TMWA; however, as he earns credibility points for each successful validation, he has an incentive to voluntarily do so. The Web CA/TMWA has a strict and clearly published privacy policy about keeping this data.

Also notice that the newcomer's personal data will be seen only by introducers (and possibly external auditors), which are expected to be much less than all Verified Identity users, and even less that the public at large. The TMWA may also offer to show the newcomer's personal data only to introducers he explicitly allows or invites, such as close friends, business associates, etc.

Groupings of the user's personal data could be validated by performing a HTTP web query on widely known and respected services. (This query would be performed by an automated script; no CA operator or human assistance should be necessary.) For instance:

- **Addresses and phone numbers:** these could be validated by checking them on whitepages directories such as knowx.com, whitepages.com and the like. It is considered valid only if the phone/address is registered with the user's name. Other people living in the same address would not pass this validation, but they have other alternatives.
- **Country-specific identifiers in public national databases:** Unique identifiers would be especially desired. For instance, several Brazilian governmental agencies' web sites provide web interfaces for querying their databases. Our prototype implementation has robots to check users' driver licenses, elector IDs, and others. The sites usually return the users full name and other status information when given the numeric IDs the user entered in the TMWA. If the name they give match (with some fuzziness factor to account for slight misspellings and truncations) with what the user provided, he is granted a few points.
- **PGP Key-based validation/introduction:** if the newcomer has a PGP key, he could post it to the TMWA. The PGP automatic validator then sends him an email encrypted with his PGP key containing an URL with a random validation code. If the TMWA receives the hit in this URL (which, remember, requires SSL client authentication), we take it as proof that the owner of the PGP key is the same person that owns the SSL client certificate. For that, we grant him a few credibility points.

  Since we are sure the user controls the PGP private key, we can take it a step further: if the user's PGP public key is signed by some trusted introducer, then it will be regarded as a direct user introduction, as described in section 3.2 – but performed in an entirely automatic manner. This is a special case where a "weak" validation may become a "strong" introduction.
- **Photographs and other human-verifiable data:** Certain personal data, such as headshot photographs, could also be accepted. Since they cannot be validated automatically, they would just "sit there" waiting for a human introducer to validate (as a means of saying "I attest that I checked that the individual who owns the private key corresponding to this certificate looks like this photo") or repudiate ("This is the picture of a slug and this certificate holder is fooling around with the system"). More about that in section 3.2 .

It is important to remind that all this personal data is kept in the TMWA database only. It is not included in the digital certificate when it is finally granted to the user. The VI certificate's DN is nearly the same of the EL certificate (except for the name of the VI CA in the OU field).

As each successful validation is achieved, the user's credibility points should be increased by the validator's trust weight multiplied by a measure of the

success of the validation. Figure 4 illustrates the process schematically, while figure 6b shows some snapshots of the process being conducted in our prototype implementation.

These kinds of validations are said to be "weak" because they are based on public data. They don't really prove the user is who he says he is. Thus, the amount of credibility points a user receive by these validations should be small compared with other validators, given that anyone can get personal data from some random individual in the very same services the TMWA uses to validate them and claim to be someone else.

The primary security function of the weak validators is to make it harder for a spoofer to get a certificate issued to an entirely fictious individual whose existence would be unlikely to be challenged. By having to assign a verifiable identity to the certificate, a spoofer incurs the risk of being challenged by the spoofed individual, as detailed in section 3.3 .

3.2 Strong Validators

The fastest way for a user to gain credibility points in the trust scoring system is by having other participants, especially highly trusted ones, to *voluntarily* verify his identity. This is particularly easy if the newcomer has a friend, supervisor, business associate or anyone within his acquaintance that holds a sizeable amount of introducer points.

The process is envisaged in the following ways:

- **TMWA introduction:** suppose Newton the newcomer asked (by email or though the TMWA community service) Ingus the introducer to vouch for him. Ingus logs on the TMWA, searches Newton in the database and fills a form specifying the amount of certainty he has that the individual he is introducing is who he says he is. This number, multiplied by his introducer score and an attenuation factor, is added to the Newton's credibility score. The attenuation is to prevent a single introducer from being able to escalate someone else's credibility too fast. Figure 5 sketches the situation schematically and figure 6c/d show the same situation happening in our prototype implementation.

  In order to encourage Ingus to perform the confidence level evaluation with the greatest care, the system informs him that if Newton is later determined to be a fraud, Ingus will have his introducer points reduced by the same percentual amount of confidence he deposited in Newton; and will receive as many suspicion points – which might put him directly in suspicion mode if it turns out to exceed his credibility. In other words, Ingus' evaluation is interpreted to be like an *insurance*: the amount of his own trust he would be willing to lose if Newton is found not to be who he says he is.

- **Cross-Certification**: A natural generalization of the PGP key-based "weak validator that may become an automatic strong introduction" is to accept certificates from other CAs or key hierarchies whose validation processes are known and that can be easly assigned a credibility rating. For instance, Verisign certificates could be accepted as another level of validation – Class 1 certificates, which validate only the email address, would add little extra credibility, while Class 2 and 3 certificates, which rely on institutional credentials and in-person enrollment, respectively, would grant much more points. Certificates from the CAs within our own hierarchy that employed traditional validation processes, as described in section 3 , could be likewise accepted.

It is worth reminding again that all these operations should be carefully logged, both for debugging and auditing purposes, so it becomes possible to reconstruct exactly why any particular user has got his scores.

3.3 Contentions

If a user Charlie the challenger supplies an unique identifier (say, his name, e-mail address, SSN, etc.) already claimed by someone else, he is to be put in *suspicious mode*: he earns as many suspicion points as the sum of the credibility scores of each user (himself included) having the same ID.

If Charlie's credibility reaches a certain fraction (say, half) of the credibility of some user he is contending with, the challenged user gets notified of this fact by email. This warning should give him time to take precautions against *takeover*: if Charlie's credibility exceeds the challenged user's, Charlie is awarded possession of the contended IDs. The challenged user is then put into suspicious mode: it's now his problem to prove his identity beyond Charlie's credibility.

These rules attempt to foil some avenue of identity theft attacks outlined below:

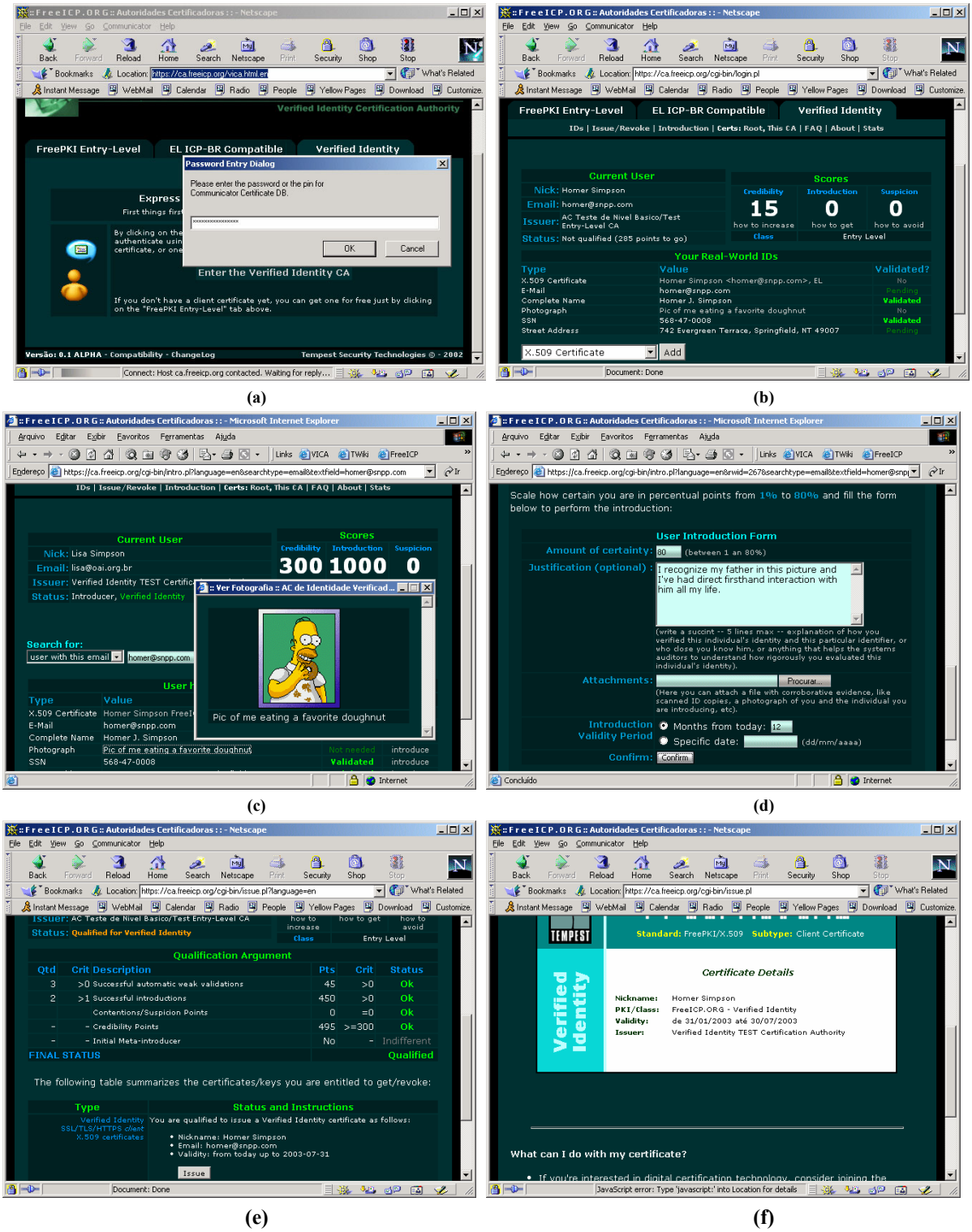

Figure 6: A simulation of Homer getting his VI cert: In **(a)**, he uses his EL client certificate to log on to the TMWA/VI CA; in **(b)**, we see him after he has already inserted some of his real-world IDs; as he was inserting them, some automatic validators have started their jobs: we see that his name/SSN tuple has already been validated, while the email and street address validators are still in the execution queue ("pending"). In **(c)**, Lisa logs on to the TMWA and searches for Homer. She checks his photo and in **(d)** vouches for it. In **(e)**, we go back to Homer and see him already qualified. After clicking the Issue button, he finally gets his VI certificate in **(f)**.

- **Post-takeover:** Suppose a legitimate user has already got his VI certificate issued without incident. Then, a persistant attacker issues several EL certificates with his name and uses them to log in the TMWA and generate contentions, supplying the legitimate user's public personal data to pass

through many weak validators and gain a modest amount of credibility points. As long as the legitimate user keeps his own credibility points high, the contenders won't be able to steal his identity. He is probably in the best position to do so, since he can convince introducers to attest his identity and strong validators yield so much more credibility points. The legitimate user gets early notification about contenders and their chances to take over his identity.

- **Pre-population:** The attacker enrolls in the TMWA before the legitimate user, supplying some of the spoofed user's public personal data to pass some of the weak validators. If the eligibility criteria for getting a VI certificate is set to near or more than the sum of what all possible weak validators could give, or requires a minimum number of introducers regardless of the credibility points, the attacker won't be able to assume the (unsuspecting) legitimate user's identity. Later, when the legitimate user enrolls, he will get into suspicion as soon as the starts contending with the spoofer; but since he is "the real one", he should be in the best position to convince the introducers to vouch for him and should win the credibility point fight easily.

All that relies, of course, in the trustworthiness of the introducers and the rigor with which they perform every single identity check. A rogue introducer can help attackers to bootstrap themselves through the credibility ranks or even create whole cliques of self-certifying fake communities (as long as the real users being spoofed don't enroll in the system and start contending with the fakes). The population base of our prototype implementation has not reached enough critical mass to allow these phenomena to be empirically observed, measured and statistically characterized, but it is natural to expect these issues will manifest themselves as the population base grows.

The fact that credibility points given from the introducer to the introductee act as insurance creates an incentive for caution: the introducer should know that if an identity validation error from his part is discovered (say, by other more graduated introducers or external audits), it will revert against himself, almost certainly quelling his privileges – a phenomenon we call "introducer demise".

In our prototype implementation, we didn't make introducer demises propagate through all of his introductees – this forces the whole trust scores to be recalculated, and, if not carefully calibrated, may make the entire trust web collapse. It was felt as undersirable in our small web, making it too fragile; but may be considered a minor local event in a large scale (say, millions of nodes) web, adding a self-correcting nature against introducer-aided fraud. Surely, this deserves deeper study.

At any rate, it is expected that contentions require much more human intervention than identity validations that go about without incidents. On the other hand, it should be possible to calibrate the system so that the former happens much more rarely than the latter.

Contentions also help to avoid "no way out" situations. For instance, it is not rare for users to lose their private keys. In these cases, we simply direct the user to get a new EL certificate and use it to reenroll in the VI CA. He will immediately start a contention with the "old copy" of himself – however, by simply reinserting his personal data and asking the same introducers he used last time to vouch for his identity, he should be able to surpass his old version's scores and get a new VI certificate. This will also cause his old VI certificate to be revoked, the old account to get in suspicion and eventually deleted by the garbage collector.

In short, contentions (and the whole score system) play an essential role in *managing* the users identities and real world IDs associations. It doesn't aim to be 100% fraud-proof; instead, it tries to be good enough for practical purposes and provide means of discovering and correcting errors, insofar as possible in an automatic manner; and appeal to the introducer community as a last resort.

3.4 VI certificate eligibility criteria

Our prototype implementation has only one VI CA with a very simple acceptability criterion: if the user exceeds 300 credibility points given from at least two introducers, he is granted a "VI level 1" certificate. This simplistic approach was chosen because it's easy to explain, simple for users to know what to do and makes the process of getting the VI1 certificate very quick: the user enters as much personal verifiable data as he wants, gathering a small amount of credibility due to the weak validators; then he consults the public list of introducers in the TMWA community page, asking the ones he knows to vouch for him.

Typically a few hours later, when the introducers check their emails (the TMWA informs them that someone asked to be introduced), the newcomer is validated and he is invited to the VI certificate generation page (it is worth noting that all this is made with SSL client authentication, thus requiring his EL certificate). His VI certificate is then issued in the same single-step manner adopted by the EL CAs and, finally, the user is informed of the applications that accept/require his newly issued certificate, along with instructions about how to register with them.

We plan to have "level 2", "level 3", VI certificates with stricter validation requirements, such as requiring several introducers, allowing the introducers to specify the validity of their trust grant and allowing the

newcomer to attain VI status only if at least one introducer vouches for him for at least one year (the suggested validity period of the VI certificates), etc. Another idea is to have a VI CA that requires the introducers to be members of stricter PKIs, such as ICP-BR (the Brazilian National PKI).

This makes a good moment to remind that each certificate from each CA has a life cycle of its own; they are not necessarily coupled or associated in any way. There's no need, for instance, to revoke an EL or a lower level VI certificate because the user has been issued a higher-level VI certificate. The only tying association is that they're kept in the TMWA.

It is interesting to compare this authentication metric with others, such as the ones studied by [17]. It is worth repeating the eight authentication principles they laid out and comment how our system adheres to or deviates from them.

- *Principle 1: The model, to which a metric is applied, should not require the user to infer bindings between keys and their owners. In particular, when representing certificates in a model: entities don't sign certificates, keys do.*

  In our system, the TMWA clearly identifies the several identities associated with a particular keypair/certificate, leaving no room for guesswork.

- *Principle 2: The meaning of the model's parameters should be unambiguous. This especially applies to the meaning of probabilities and trust values in the models that use them.*

  The numeric trust scores provide quantitative estimates of each trust quality (credibility, introducer, suspicion, etc). The scale and calibration may be somewhat arbitrary, but, within itself, it's self-consistent.

- *Principle 3: A metric should take into account as much information as possible that is relevant to the authorization decision that the user is trying to make.*

  The user (or application) doesn't make much more authorization decisions than choosing what EL or VI CAs to trust. But their acceptability criteria can be very well specified. We have tree different scores, which seem already a great deal of relevant authorization information – our system even has suspicion detection and management, a feature not found in many other metrics. We feel that more than that would overcomplicate the system.

- *Principle 4: A metric should consult the user for any authentication relevant decisions that cannot be accurately automated. A decision that could affect authentication should be hidden from the user only if it can be reached using unambiguous, well-documented, and intuitive rules.*

  That's precisely what strong validators are for. Since it was felt that automated validations could be rather easily spoofed, we made them the weak validators.

  On the other hand, our concept of "trust insurance" doesn't mean "monetary insurance" that would be paid in case of system failure (although it may be conceivable that it may provided as a add-on commercial service); instead, it means only a guarantee that introducers will be penalized for errors or misbehavior.

- *Principle 5: The output of a metric should be intuitive. It should be possible to write down a straightforward natural language sentence describing what the output means.*

  It is easy to explain what the metrics measured: "you got *n* points from one introducer, *m* points from another one, *i* points from posting your SSN, *j* points from posting your email, *k* points from posting your Brazilian CPF number, which add up more than the *t* threshold needed to get you a VI certificate."

  This opens up an interesting possibility: the page containing the certificate's CPS could add, within the bulk of the CPS text, an automatically generated, natural language explanation of these metrics and the guarantees (technical and legal) they provide – much like the "Unabridged Certificate" proposed in [7].

  Although the implementation has to take into account a lot possible state transitions, it is surprisingly easy to explain the dynamics of the scores due to its close mapping to how we intuitively transfer trust in the real world: we believe someone is who he says he is when he shows credentials and our acquaintances confirm; the credibility points just put a numeric scale to it. When we are introduced by someone highly regarded, we "gain" his credibility – he doesn't lose it unless we are proven to be a fraud. When we catch two or more people claiming to be someone else, we try to gather more and more evidence that supports one of them and disproves the others. Consistently bad introducers tend to develop bad reputations and become no longer trusted.

- *Principle 6: A metric should be designed to be resilient to manipulations of its model by misbehaving entities, and its sensitivity to various forms of misbehavior should be made explicit.*

  Section 3.3 detailed some of the contention management and their resistance to misbehavior.

More field experience is needed, however, to ascertain their efficiency in practice.

- *Principle 7: A metric should be able to be computed efficiently.*

  Since the TMWA enforces only direct introductions, there is no need to construct the entire introduction graph to compute the trust scores nor run graph-theoretic algorithms with superlinear time complexities (it may be useful to build the graph for other purposes, though). The calculations can be done incrementally and even reconstructed from the transaction log in linear time.

- *Principle 8: A metric's output on partial information should be meaningful.*

  Any user registered in the TMWA has trust scores, even if they have passed no validators. So, the metric is meaninful (although not useful) even in the absence of information.

3.5 The Root CA

The root CA has a very simple website offering the following services:

- **Automatic Entry-Level CA certificate signing:** the Entry-Level CA reference implementation sports a semi-automatic installer. One of its chores is to request the name and administrative email address of the new EL CA use them to generate its private key and CSR. It then sends it to a special URL within the Root CA's site that enqueues CSRs for processing by the signing engine. The queue has some built-in intelligence to discard duplicate attempts within a certain timeframe and avoid some flooding attempts. After being signed, the resulting certificate is sent to its administrative address specified in the beginning of the process.

  It has been suggested that the signing process should demand that the EL CAs administrator should be VI users; this guarantees a contact person and helps minimize rogue EL CAs. Although not implemented at the moment, this will probably be done in the near future.

- **Manual CA certificate signing:** an alternative manual procedure in case the automatic fails; now seldom used.

- **Revocation and non-compliance denounce:** the EL CAs have only a few obligations: they must not generate certificates that diverge from the naming policy nor issue certificates with validity periods greater than three months. But since the EL CAs operators have the source code, they may very well cheat. It's not possible to avoid it preventively, but the root CA can "retaliate": if anyone submits a nonconformant certificate to this service, the root CA will revoke the EL CA's certificate. (Generating an invalid certificate on prupose with a special "self-destruct" string is the correct, although exotic, procedure that the EL CA administrator should follow when he wants to revoke it). Admittedly, this is a rather weak contermeasure, given that most relying parties may not check the root CA's CRLs regularly or at all.

3.6 Server Certificates

Our initial focus was to identify individuals. However, one of the biggest demands – which spawned many commercial CAs – is to provide server identification. A free, collaborative way to securely identify servers might be desirable. Many of the concepts we developed seem to apply equally well to this field.

- The weak validator concept can be, in principle, extended for Internet hosts (say, for IPSec using IKE) or SSL servers: the robot would "ping" the service to see if it is up and running in the DNS or IP address specified by certificate's DN. In the case of SSL or IKE, it could also check if it is returning a proper set of certificates, etc. It should be possible to validate many kinds of services: HTTPS (HTTP over SSL), POP3 and SMTP over SSL, and possibly other less popular services, such as TELNET, FTP, VNC or Jabber over SSL.

- Internet hosts could be introduced in a similar way, except that their administrators would act in their behalf, inviting introducers to vouch for the identity of their SSL servers or IPSec-enabled hosts.

These generalizations, however, may be suceptible to DNS forgeries. Besides, there seems to be some confusion about what kind of guarantee the system could provide: many users misunderstand the term "secure site" and unrealistically expect them to mean "unhackable", or that the institution running the server is trustworthy; among many other interpretations quite different from the correct one. We are still working on a sensible set of validation procedures more easily understood by introducers and final users alike.

4 EXPERIENCES WITH THE FIRST RELYING-PARTY APPLICATIONS

The VI CA itself is the zeroth relying party application, since it is a full-fledged Web aplication requiring SSL client authentication. However, its tight integration with the other CAs makes it too much of a special case; to really grasp what our infrastructure could do, we selected another application to add FreeICP support to. TWiki [32], a web based collaborative content management system, was the natural choice, since we already used to run a few Wiki sites and making a PKI-enabled version with stronger authentication and improved security was a longtime wish of ours.

A short description of TWiki's functionalities follows: it looks just like an ordinary web site but allows editing the web pages (called "topics") directly in the web browser, adding attachments and keeping everything under revision control, so it's possible to reconstruct any past version, know who changed what and when, or undo undesired changes. Topics about

Figure 7: FreeICP integration in applications: The simplicity of Entry-Level CAs allow them to be included as a small visual element (the "Quick Registration" box) in a web application. It is integrated with the application in the sense that it not only generates certificates, but also performs the application setup necessary to create the user's account.

the same subject of interest are grouped in "webs". It has a simple but effective access control system: each web can have an access control list defining which users may be granted or denied permission to read or change the information. Individual topics can also have these ACLs for further granularity.

The original TWiki identified its users by the traditional username/password pair through the standard HTTP BasicAuthentication mechanism. User names are internally mapped to WikiNames satisfying its special naming conventions. To use mod_ssl's FakeBasicAuthentication mechanism is a natural and simple way to "upgrade" the system to use client certificates instead. This, however, proved rather unsatisfying, so we quitted using it and decided to implement client certificate support directly in the core application code:

- The application parses the certificate and maps the DNs to usernames. If it is not found in the user list (which is itself a topic), it redirects the user to an error page (except in the VI listing upgrade case described below).
- If the user's certificate is not a Verified Identity one, it is only granted access to public webs; that is, ones with no ACLs – even if that user is explicitly included in some web's ACL. This implements the "low privilege, guest-like access" principle that Entry-Level certificates should have. In this mode, the user can read the tutorials, practice with the test/sandbox areas, but has no access to sensitive information.
- If the user logs with a VI cert but is still enrolled with *a corresponding* EL certificate (i.e., one with exactly the same name and email address), the user is not redirected to the error page; instead, it is sent to a page offering to automatically upgrade his registration data. After confirming, he can no longer log on with his EL certificate; from this point on, only his VI certificate will be accepted and he will be granted access to the private webs (i.e., ones with explicit ACLs). This implements the "higher level, privileged" access principle that VI certificates are entitled to.
- An integrated Entry-Level CA was added as the new user registration box, as shown in Figure 7. That way, the user gets his certificate issued and his initial setup in the application (creating his personal topic from a template, adding him to the user's list) done in two simple steps (modulo the web browser's idiosyncrasies explained before).
- Besides getting access to the private webs, another motivation for upgrading to a VI certificate is the fact that when EL certificates expire, anyone can issue a new one with the same name/email and thus fake the previous user. To avoid that, we made the system accept EL certificates only up to two weeks from the initial inclusion in the users list, regardless of the age of the certificate.
- To cope with many revocations caused the users intial experimentation with client certificates, we implemented a full-blown policy-based revocation verification system. At first, we used mod_ssl's built-in CRL verification features, but it had a few limitations: first, we had to have external scripts to download the CRLs and put them in files that mod_ssl could read – that is, we couldn't have *on-demand* CRL downloads. Secondly, mod_ssl breaks the SSL connection when it determines the client certificate has been revoked. Although it seems the right thing to do, that denies the application an opportunity to display a page explaining to the user why his access was denied. Worse still, when this happens, Internet Explorer displays a bogus dialog box complaining that "the site cannot be trusted", instead of something more truthful like "this client certificate has been revoked".

  Because of all that, we disabled mod_ssl's revocation checking and patched TWiki to support it natively. It proved to be quite a challenge itself, but in the end it supported on-demand CRL downloading (i.e., it only downloads a CRL when it is needed to check the user's certificate), which contributes to alleviate the classical CRL problem of every relying party wanting to download the

freshest CRL at the same time; it displayed a nice message explaining to the user what happened; and incorporates some features to make it resistant to transient CRL download failures.

We made several other small changes to TWiki's core functionality. Although many of them were security related (for instance, the search feature didn't respect the ACLs; we fixed that) and sometimes quite interesting by themselves, most have little relation with digital identity support and have been omitted here for sake of brevity.

The final result was quite satisfactory: we managed to keep the registration process very quick and simple from the point of view of the novice users wanting immediate access to the tutorials and public webs. And, by compelling users to upgrade to VI certificates, we achieved considerable certainty about their identity and that they could only see information they were strictly authorized. Some informal testing we made in trying to subvert the system was promptly detected, but much greater scale testing is still needed to evaluate its merits relative to other authentication technologies.

It is natural to ask whether how well and quickly the users grasp all those trust scoring rules. In our experience, most users only invest the time to understand what they strictly need. Since most of our users only wanted to get access to TWiki and other apps, they got to learn only the rules related to increasing their credibility score (and many promptly forget them after getting the VI cert). Even so, we consider the fact that many users can get their VI certs in something between a few minutes to a few hours a striking success.

People only dive deeper when a suspicion event happens or we compel someone to become an introducer, requiring a more thorough understanding of the whole process. When their curiosity is then aroused, these users usually didn't feel intimidated by the complexity of the system; many end up making us explain all those rules in great detail. The single biggest reason for user rejection, in our experience, has come from IE users when the EL express certificate issuance process fails – which, unfortunately, happens in more than half of the cases.

5 CONCLUSIONS AND FUTURE WORK

We proposed two CA families to implement a PKI mixing the PGP and X.509 models based on the realization that the process of aggregating strong identity guarantees to a certain key/certificate should not be tied to its issuance; it should be done at a later moment, if and when convenient to the certificate holder. In fact, there are many instances when it's simply not worth the hassle to go through an extremely strict identity validation procedure when a not-so-trusted certificate would do just fine.

In our system, the Entry-Level family of CAs provide this focus on user and administrative simplicity. We've argued that it provides roughly the same kinds of protections that the PGP infrastructure: confidentiality through encryption but with little certainty of who the keys onwers are in respect to other identification systems. The proposed scheme allows the certificate to be granted immediately, becoming well suited for replacing website registration systems and similar end-user applications. The short lived certificates, when combined with application demand, creates an incentive for the user to "upgrade" his entry-level certificate to the longer lived, more widely trusted, Verified Identity ones.

Space constraints prevented us from being able to report the many interoperability pitfalls we ran into, the nontrivial solutions we were often forced to adopt and several other interesting implementation details. These may make material for a future paper; meanwhile, the reader is invited to visit our implementation site: www.freeicp.org.

The proposed Verified Identity family of CAs provide the higher identity assurance levels. It can be seen as a framework to unify several identification services and strictness criteria. It encompasses both the human-operator-based identity check systems now common on commercial or institutional CAs and a novel idea of a trust scoring web application that allows borrows the PGP's web-of-trust model but implemented over a centralized database to provide online-only, semi-automated identity validation – vaguely resembling the credit scoring systems now common in financial institutions. We argue that its collaborative nature may be exploited to make near-zero-cost certificates possible and thus allowing the "commoditization" of trustable digital certificates.

A trust management system was described that allows the users to tie their certificates to automatically verifiable real world identities and accumulate credibility by having these identities verified by veteran users that act as trusted introducers. The proposed model uses a much more precise system based on numeric scores that evaluate the user's identity credibility, trustworthiness as an introducer, and the amount of dispute that the user is having to gain control of other user's identities. In fact, contention detection and control is another area that this system proposes and both PGP and X.509 lack. Precisely because of its novely, it deserves deeper study.

We have shown that Verified Identities CAs can use these trust metrics to decide, according to their own acceptability criteria, whether a particular user or internet host is eligible to one of its certificates. A simple threshold criterion was proposed that subjectively adheres to all the authentication metric design principles posed by Reiter and Stubblebine. An

interesting point is that the metric allows for an easy description of itself in natural language that could be added directly to an automatically generated Certificate Practice Statement.

Other interesting avenue being pursued is the use graph-theoretic algorithms to monitor the growth of the certification network and provide feedback to help calibrate the system parameters to achieve specific security guarantee goals. Their use as authentication metrics may be also considered.

The field of automated identity verification has been blossoming with interesting new proposals. For instance, in [1] it is described a system in which an automated voice system dials to the telephone number the user supplied in the enrollment process and requests the user to confirm a challenge number and record his name and affiliation, for audit purposes. A whole different idea, much more sophisticated, would be to accept digitized fingerprints to be matched against law enforcement's databases. The inclusion of those kinds of automated identity verification systems within an implementation of the framework proposed in this paper may become a worthwhile research avenue.

Finally, we studied the customization of a web application to suport user identification using our CA infrastructure. Several proeminent lessons emerged: first, web browser's UIs could be adjusted to provide simpler certificate generation that could bring us closer of the "express certificate" concept brought by the EL CAs – in particular, Microsoft's Internet Explorer proved to be an endless source of user frustration. Second, applications must undergo significative changes to support the "temporary limited access" semantics of EL certificates and wider privileges of the "Verified Identity" class of users. Besides, revocation checking can no longer be ignored; applications must have full revocation verification support and its interactions and potential vulnerabilities must be carefully understood; this might become quite a challenge by itself. Notwithstanding, we have shown a situation in which the final result was quite acceptable. We plan to add FreeICP-like support to many other kinds of applications.

6 ACKNOWLEDGEMENTS

Thanks are due to Aldo Albuquerque, João Paulo Campello and Felipe Nóbrega for their helpful suggestions, insightful criticisms and invaluable assistance in coding the prototype implementations. We also thank all users at C.E.S.A.R. for agreeing to be our test population base by using our PKI-enabled TWiki and the other FreeICP-compliant applications. Rômulo Albuquerque, Renato Martini and Robson Gomes were especially patient with the quirks of the first versions. We are also indebted to the anonymous referees for their most valuable coments and criticisms on the first version of this paper.

"The Simpsons" characters are trademark and copyright of Fox and its related companies, even though they have arguably became part of popular culture. Used here just for entirely non-profit illustration purposes.

7 REFERENCES

1. Authentify, Inc., *Authentify|Register™ and RSA Keon® OneStep – Assuring User Identities in the Registration Process*, http://www.authentify.com/images/pdf/AR_RSA_Keon.pdf
2. Marc Branchaud, *A Survey of Public-Key Infrastructures*, MSc. Thesis, Department of Computer Science, McGill University, 1997.
3. J. Callas, L. Donnerhacke, H. Finney, R. Thayer, *RFC 2440: OpenPGP Message Format*, 1998, www.ietf.org/rfc/rfc2440.txt
4. Comitê Gestor da ICP-BR, *Resolução nº 11 de 14 de fevereiro de 2002*, www.icpbrasil.gov.br/RES_ICP11.htm
5. Don Davis, *Compliance Defects in Public-Key Cryptography*, Sixth Usenix Security Symposium Proceedings, July 1996, pp 171-178.
6. Simon Garfinkel, *PGP: Pretty Good Privacy. O'Reilly & Associates*, 1994, ISBN 1565920988
7. Ed Gerck, N. Bohm, *X.509 Certificates: A Readable Unabridged Inside View,* www.mcg.org.br/x509cert.htm
8. Ed Gerck, *Overview of Certification Systems: X.509, CA, PGP and SKIP*, MCG Group, 1998, www.mcg.org.br/cert.htm
9. David Goodenough, *A Heretic's view of Certificates*, www.dga.co.uk/customer/publicdo.nsf/public/WP-HERESY
10. Richard Guida, *Rebuttal to "Ten Risks of PKI"*, Computer Security Institute Alert, n 204, 2000, www.gocsi.com/pdfs/expert.pdf
11. Peter Gutmann, *X.509 Style Guide*, 2000, www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt
12. Russel Housley, Warwick Ford, Tim Polk & David Solo, *RFC 3280*: *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, 2002
13. ITU-T, *Recommendation X.509/ISO/IEC 9594-8: Information Technology – Open Systems Interconnection – The Directory: Authentication Framework*, Internation Telecommunication Union, 1997.
14. Burton S. Kaliski Jr, *An Overview of the PKCS Standards*, RSA Laboratories, 1993

15. Neal McBurnett, *PGP Web of Trust Statistics*, 1997, bcn.boulder.co.us/~neal/pgpstat
16. Patrick McDaniel & Aviel Rubin, *A Response to "Can We Eliminate Certificate Revocation Lists?"*, Proc. Financial Cryptography 2000, February 2000.
17. Michael K. Reiter & Stuard G. Stubblebine, *Authentication Metric Analysis and Design*, ACM Transactions on Information and System Security, Vol. 2, No. 2, May 1999, pp 138-158.
18. Ronald Rivest, *Can We Eliminate Certificate Revocation Lists?*, Proceedings of Financial Cryptography 98, LNCS 1465, Springer-Verlag, pp. 178-183, Anguilla, BWI, February 1998
19. Bruce Schneier & Carl Ellison, *Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure*, Computer Security Journal, v 16, n 1, 2000, pp. 1-7, www.counterpane.com/pki-risks.pdf
20. William Stallings, *Cryptography & Network Security: Principles & Practice*, 2nd Edition, Prentice-Hall, 1998, ISBN 0138690170
21. THAWTE Inc., *Certifying your Credentials through the Thawte Web of Trust*, www.thawte.com/whitepapers/guides/pdfversion/wotguide.pdf
22. VERISIGN, Inc., *VeriSign PKI Disclosure Statement*, www.verisign.com/repository/disclosure.html
23. Alma Whitten, J. D. Tygar, *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0*, Carnegie Mellon University, Proceedings of the 8th USENIX Security Symposium, August 1999, www.cs.cmu.edu/~alma/johnny.pdf
24. Jane K. Winn, *The Emperor's New Clothes: The Shocking Truth About Digital Signatures and Internet Commerce*, 2001, faculty.smu.edu/jwinn/shocking-truth.htm
25. Phillip R. Zimmermann, *The Official PGP User's Guide*, MIT Press, 1995.
26. M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, *RFC 2560: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*, 1999, www.ietf.org/rfc/rfc2560.txt
27. PGP Corp Web Site: www.pgp.com
28. OpenSSL Project: www.openssl.org
29. Mod-SSL: www.modssl.org
30. Apache Web Server Project: www.apache.org
31. GNU Privacy Guard: www.gnupg.org
32. TWiki: A Web-Based Collaboration Platform: www.twiki.org.
33. Jonah Freeware PKIX implementation: www.foobar.com/jonah
34. Cryptlib security toolkit: www.cryptlib.orion.co.nz

Improving Message Security With a Self-Assembling PKI

Jon Callas *(PGP Corporation, Palo Alto CA, jon@pgp.com)*

April 4, 2003

Abstract

Public Key Infrastructures (PKIs) exist for a number of purposes. One purpose for a PKI is achieving wide-spread deployment of secure communication; the PKI makes it easy for two parties to communication securely. Another purpose is that of secure delivery to a recipient; the PKI makes sure that when Alice sends a message to Bob, it arrives to the very Bob that Alice intended it to arrive to. The paper describes a *Self-Assembling-PKI,* a new way of constructing the corpus of certificates that makes up a PKI with a solution to the first problem – widespread deployment of secure communication. It does not address the second one, but can use and interoperate with a PKI designed to achieve that goal. It creates itself by observing and monitoring existing message traffic, and transparently inserting security protocols into existing traffic. Most radically, its mechanisms trade perfect security for ubiquity.

1 Introduction

The *Self-Assembling PKI* is not a new technology, nor does it require new standards; instead it is a new way to think about existing PKIs, security standards, and systems to achieve these goals:

1. Wide-spread deployment of communications security. Presently, within organizations, secure email tends to be used by five to fifteen percent of the organization [PGPUSE]. This figure applies to a self-selecting group that includes people who already consider secure email to be important or have regulatory requirements. Network-wide, this number is much lower, probably no more than five percent of total users [GARTNER]. A number of factors cause this, all of which ultimately center around ease-of-use [JOHNNY].

2. Transparency of use. Even within a population of people who regularly use traditional public-key cryptography, human error is a formidable problem. The solution to these inevitable lapses in judgment is a system that requires no thought on the part of the user.

3. Ease of deployment. Not only must a PKI be easy for the end users to work with, but it must be easy for its administrators to set up, deploy, and run. Traditional PKIs have not been widely deployed, difficulty in deployment being much of the problem [GUTMANN]. This is another facet of ease of use, merely with another set of users.

4. Policy management. A PKI that is operating on behalf of end users must have a collection of rules that describe what actions it performs under what circumstances. These rules are the PKI policies; they describe both the mechanisms that apply to outgoing messages and incoming messages as well.
5. Risk Mitigation. Because the PKI is operating on behalf of humans and without their intervention, it is vital that it include detailed reporting. Inevitably, the PKI will contain errors, and its human administrators must themselves detect and correct these problems. This cannot happen without robust reporting.

Additionally, there is an old principle of security that the value of what is being protected affects the measures taken to protect it. This approach strives to the above goals, but is willing to trade perfection for ease of widespread deployment in the belief that a message security system with known limitations that can be used by anyone is more secure overall than one that can only be used by a few experts.

2 A Change in Metaphor

Traditionally, a PKI operates using a telephone metaphor. From the very first description of certificates, they were described to be analogous to a telephone number; Alice would find Bob's certificate similarly to the way that she might find his telephone number.

The Self-Assembling PKI is the metaphor of the *robot operator*. The PKI determines the connection and relationship between the sender and recipient, and processes it. Without prior registration into the PKI, however, there is no way to complete the connection – if you don't have a phone, you don't have a number, and no one can place a call to you.

The Self-Assembling PKI uses a different metaphor. It is a postal metaphor rather than a telephone metaphor. This operates as if it were a *robot messenger* that has the job of delivering information as securely as possible. The messenger operates on behalf of the sender and recipient, adding security to the system (which would otherwise be done with standard insecure protocols) as much as it can. It uses the policies and heuristics of both the sender and recipient to improve a transaction between them.

We look at the problem this way because of the realities of how users use Internet communications. When someone sends an email message, they expect it to be delivered post haste. Security is a feature they desire, but delivery is what they are after. Similarly, people rarely stop using other systems such as web browsing and instant messaging just because they are reasonably insecure. In fact, one of the main security concerns organizations have about instant messaging in particular is that users shift to it because it is fast, reliable, and convenient. Instant messaging sees great growth in organizations where email security policies make email inconvenient.

People need to communicate more than they need to communicate securely. As security system designers, we may not like this, but it behooves us to be their messengers rather than their switchboard operators. If we refuse to connect their call, they don't decide they didn't need to place the

call, they just find another way to do it. The messenger metaphor is the attitude that the mail must go through. It is a step towards making PKIs be used when they haven't in the past. It makes PKI be an enabling rather than disabling technology.

While the messenger metaphor applies most closely to email, it also fits other protocols and communication systems. I have already mentioned instant messaging, but the principles apply to many other systems as well.

3 How the PKI Self-Assembles

The whole point of a self-assembling PKI is of course that it does not require its administrators to construct it before it can be used. Often a PKI requires the people who construct it to understand the larger system it resides within. If they must do this while they are building it, it dramatically slows down construction.

Compounding this difficulty, even though the resulting PKI might in theory be completely accurate, the externals may have changed. If it takes longer for the PKI to be constructed than for the system it serves to change, then the PKI will never be accurate.

Self-assembly shifts the PKI staff from constructing the infrastructure to overseeing it. They correct inaccuracies, shape policy, and adapt the PKI to the larger system it serves. It is similar to the manufacturing principle of continuous improvement. It combines the great power of computers to rapidly, accurately do repetitive tasks with the power of humans to understand complexity and provide feedback to mechanism.

Components of PKI construction work by getting in the middle of the network processes, monitoring them, observing them, and constructing the PKI so that it reflects the actual use of the system.

Other components of PKI work within the active network processes, shaping them and adding security features. For example, email can be encoded to have a security envelope. An instant message can also be wrapped with added authentication and message privacy.

It is also important to note that a pre-existing PKI only enhances these newer, more flexible components. This need not replace existing PKIs. The robot messenger can exploit the work done to create robot operators.

3.1 Format Agnosticism

The robot messenger desires to deliver messages, and desires to deliver them as securely as possible, despite obstacles. A physical messenger must overcome rain, snow, and dark of night. The robot messenger has to overcome a wide variety of security standards including OpenPGP, X.509, S/MIME, SPKI, XKMS, TLS, and so on.

The robot messenger must therefore be without religion when it comes to message and certificate format. It must be able to speak a variety of protocols well enough to be understood, and well enough to abide by the policies that govern their use. The mail must go through.

While there are quite a number of possible combinations, navigating them isn't as difficult as it could be. If the messenger finds an OpenPGP certificate in a global keyserver, the recipient probably would prefer the message to be delivered in OpenPGP format rather than S/MIME. Similarly, an X.509 certificate in an LDAP directory probably calls for an S/MIME message. It's a safe bet. Heuristics do work.

Ironically, some incompatibilities can allow for better heuristics as well. There are a number of issues around using LDAP as a mechanism for distributing certificates [CHADWICK], but while the lack of a unified directory mechanism makes lookup harder, it also provides hints as to how to use the certificate.

Nonetheless, even when the system can completely infer how to use a certificate, implementing format agnosticism has a few rough edges that the implementations must overcome. Here are two obvious ones:

3.1.1 Multiple Certificates

The messenger might find multiple certificates for a given recipient. It is also possible that at least one of these certificates might be bogus, expired, or lost. Policy and heuristics can assign value to the certificates by weighting the authority of a CA, timestamps on the certificates, size of the key, and so on. While the general case can have many options, these can be truncated with obvious shortcuts such as using a certificate if supplied by some reasonable authority such as the recipient's domain.

In other cases, such as finding multiple OpenPGP certificates, a heuristic could be to use them all, or a reasonable subset. Note that in this case, there is a security issue. The issue is that Alice's message may be encoded to Bob but also to an eavesdropper, Eve, who is impersonating Bob. So long as the message is kept out of Eve's hands, the security of the message is preserved. In the general case, this can be guarded against with relatively simple mechanisms within the infrastructure – such as protecting the actual transport of the message via SSL/TLS, IPsec, or SSH. There are still cases where this does not guard against Eve, for example the case where Eve is the sysadmin of Bob's mail server. The infrastructure can help protect Bob in future messages, and some of these are described below.

The most counterintuitive situation is when the messenger finds two incompatible certificates of equal perceived value (for example, a X.509-S/MIME certificate and an OpenPGP certificate). Following the principle that the mail is to be delivered, the messenger could send the same content in two separate messages.

3.1.2 Multiple Recipients

Messages are often sent to a group of people. This creates similar issues to the section above, but with a small added twist. For example, if Alice is sending a message to Bob and Cindy, it may simply need to be coded in S/MIME to Bob, but XML-encrypted to Cindy.

3.2 Certificate Creation

The Self-Assembling PKI can use existing certificates, but to achieve its goals, it must create keys and certificates for all of its users. As mentioned above, it sits within the network infrastructure. A number of components of the PKI proxy existing protocols as part of their work. In this position, they can observe the appearance of authenticated users, and automatically create certificates. These certificates can be rewritten as more information is learned about the users.

The PKI can manage the certificates it creates for the users, or it can share them with the users for joint management. (There is an obvious third case in which a user creates their own certificate or has a key certified by a CA. For these purposes, this case is the same as using an existing PKI's certificates; this can be considered to be merely be a PKI of a single user.) PKI-managed certificates may be marked as being in the possession of a machine. Depending on policy, they may be considered lower-valued certificates than ones held solely by the end user. This is perhaps more important for a certificate used to sign rather than encrypt, but many people are uncomfortable with the notion of robot-controlled key pairs, and so we allow for (and encourage) full disclosure.

A few examples of how the PKI creates and manages certificates follow:

- Alice connects to her usual mail server over the POP3 protocol. A proxy mediates this connection, and upon observing her successfully authenticate to the actual mail server, creates a certificate for her.
- Alice sends Bob a mail message, which is itself authenticated using SMTP-AUTH. Part of the message, the "From" line of the message has Alice's full name. Her certificate gets updated to contain that common name. Alternatively, an LDAP company directory might supply personal information for that certificate. Since Bob is a user on the same mail server, the PKI creates a certificate for him. It encrypts Alice's message using the key in that certificate and sends it on to the mail server.
- Bob connects to his usual mail server over IMAP4. The same proxy mediates this connection, and when Bob reads his message from Alice, the proxy automatically decrypts it. Policy can govern whether this is wholly transparent, or whether the message is further modified to let Bob know that it was delivered securely.
- In further work with the server, rather than the clientless operation described above, software on either Alice's or Bob's computer could share the key with the server and decrypt the message locally.

- Alice could inform the PKI server (through a web browser or other means) that she prefers managing her secure messages herself. She gives the server a certificate with which it may encrypt her mail. Note that it is also possible for the server to infer this itself.

There are also opportunities for a portion of the larger PKI to use policies and strategies beyond the usual. Here are some examples:

- A team of support specialists in an operations center share the same key within their certificates. This key is changed every month, but all specialists have the same public key so that a workflow system can route tickets to any given support person.
- A high-security engineering team uses a variation of OpenPGP enhanced to support Perfect Forward Secrecy [PFS]. Their public encryption keys are essentially ephemeral, and part of the back end mail system manages the message security with paired FIPS 140 level 4 hardware security modules, and three-factor authentication on the engineers' laptops.

3.3 Side-Stepping Revocation

Certificate revocation is the hardest, stickiest, least well-implemented, and arguably most important part of managing certificates. There are numerous systems where revocation has simply been ignored and unimplemented.

Just as a Self-Assembling PKI can use an existing PKI, it can use an existing revocation scheme with CRLs, on-line checks, etc. However, within its own domain, there are pitfalls to avoid, and benefits to be gained by rethinking the part of certificate life-span where the certificate ceases to be valid.

There are also known ways to ameliorate, if not eliminate the revocation problem. SPKI [SPKI] uses the clever mechanism of simply declaring that certificates cannot be revoked. OpenPGP [OPENPGP] has revocation information travel as part of the certificate itself, with obvious advantages and disadvantages. The most important disadvantage is that it is possible for out-of-date or hostilely modified certificates to be missing revocation information. Revocation lists provide an authoritative place to get revocation information, but are vexing in many dimensions. Much work has been done on ways to eliminate them.

Being an on-line system, but one that should operate without human intervention, the SPKI solution of waving away the problem has many advantages. However, the pro, combined with some principles [RIVEST] that value new certificates over older ones, along with pushing the responsibility for certificate validity in two directions. First, the party accepting the certificate has the responsibility to decide if a certificate is good enough, and the party issuing the certificate has the responsibility to construct a certificate that the acceptor likes.

In the general case, there are potential problems with certificates that are close to expiring, as well as ones that have excessive life. However, these concerns are ones that any given messenger

may solve with its own rules. A messenger delivering high-value transactions will be pickier about certificates than one delivering chit-chat.

For our purposes, a productive and easy-to-deploy framework of the Self-Assembling PKI uses short-lived certificates. This tends towards the SPKI model, even when the data format of the certificate is X.509 or OpenPGP. It uses the validity timestamps in these certificates as freshness markers [STUBBLEBINE]. This way, internally generated certificates presented to the outside world will have a limited life, and if the keys in the certificates must be truly revoked, any "suicide notes" also have a limited life. By policy, the nominal life of the certificates managed by PKI servers range from weeks to minutes.

Additionally, these certificates may be in more detailed states than simply being valid or invalid. They could be inactive – not presently valid, but able to be re-enabled at any time. Certificates can be permitted to expire if they are not used, reported on if not used, and this can be part of the oversight into other infrastructure pieces. For example, if an employee leaves a company and the IT staff aren't told about this, inactivity of the user's certificates can alert the staff to this fact. Similarly, if a PKI discovers that a user's account on some server is no longer active, it can immediately remanufacture that user's certificate as expired, and alert the PKI administrators of this inconsistency in the infrastructure.

Certificates could also be *revocably revoked* – in effect, if not in actual syntax. It is not uncommon, for example, for an organization to regularly use the same contract staff for short stints. It is also unusual, but not unheard of for an organization to sack an employee in downsizing, only to hire that person back as a short-term contractor. As much as this situation might make security architects flinch, the *business reality* is that some important decisions don't simply happen. I might hand someone back their badge for three months. If I do, I need to hand them back their certificates as part of that "badge." The messenger metaphor is the idea that the system must work as well as possible. It is the idea that the mail must go through, the show must go on.

Reality thus gives us many reasons for remanufacturing certificates with short lives. Not only does it allow us to finesse many problems of revocation, these short-lived certificates in a flexible PKI make it possible to create adaptive solutions to real-world problems.

3.4 Certificate Trust and Search

We discussed some of the features of certificate use and search above when we discussed format agnosticism. However, much of the new thinking in Self-Assembling PKI takes place in the use of certificates.

Let us invoke again the messenger metaphor. The PKI, acting as a messenger for the user, makes a delivery to someplace else. Let us assume that the recipient has a suitable certificate, we merely have to find it. Secondarily, we need to differentiate between members of a set of certificates with varying validity. Which one(s) of those will we use? This is of course, again, a matter of policy, but policies need to be simple to create and understand. Fortunately, there are a number of simple ways to get flexible and simple search, validity, and trust systems.

Trust policy and search policy are closely related, but not the same. Search policy states where to look and in what order, trust policy states what to do with certificates as they are found. In many cases, this difference is moot – the messenger might look in an LDAP directory attached to a CA it considers authoritative. On the other hand, it might also have a local cache of authoritative and merely reasonable certificates.

There are basic mechanisms that the messenger can use as part of its overall trust policy. These relate to types of trust models.

3.4.1 Hierarchical Trust

Hierarchical trust is typical CA trust. A certificate is valid if it descends from a chain of certificates from some trusted root. Most certificate systems, including X.509 and OpenPGP allow for hierarchical trust. For example, a VeriSign Class 3 certificate is valid or not within the context of a hierarchical trust model. A reasonable policy might include that a VeriSign Class 3 certificate is not only valid, but *authoritative,* by which we mean that certificate search stops when finding an authoritative certificate.

3.4.2 Cumulative Trust

Cumulative trust is the typical PGP Web Of Trust. In this model, a certificate is valid if some collection of authorities all agree on a certification. Of all trust models, it is the least directly applicable to a robot messenger, because it relies on human judgment. However, there are two ways it fits neatly into the Self-Assembling PKI.

The most direct is that if the sender uses the web of trust to consider a certificate valid, the messenger can use that human's ruleset to consider a certificate valid or authoritative.

However, if the messenger's policy states that (for example) a VeriSign Class 1 certificate is valid, but not authoritative, this is a form of cumulative trust. The messenger will use that certificate if no better one is found, but it keeps looking for a certificate that is more authoritative.

3.4.3 Direct Trust

Direct trust is extensively used by humans who use OpenPGP-based or S/MIME-based systems, but not very much by machine-driven systems. In direct trust, we consider a certificate to be valid because we got it (or a reference to it) from the entity it represents. For example, many people print their OpenPGP key fingerprint on their business cards. This is a form of distributing direct trust. In other cases, users email each other certificates and trust them based upon that direct transfer of the certificate from user to user.

Years ago, when I met Carl Ellison for the first time, he gave me a business card with a PGP key

fingerprint. Now, nearly a decade later, with many changes of jobs and keys, I still consider his OpenPGP certificates valid based upon the direct trust from that long-lost business card.

Typically, machine-driven systems do not use direct trust, but it is a key mechanism for the Self-Assembling PKI. The messenger uses a number of rules in its policy. One of those rules is to ask the recipient's Internet domain for a suitable certificate.

There are a number of mechanisms that the messenger can use to ask the recipient's domain for a certificate. LDAP directories, DNS, HTTP-based certificate servers, or even queries via SMTP extensions.

Whatever the mechanism, the robot messenger mimics what a human messenger would do. It goes to the address of the recipient and talks to entities there. If the domain has a certificate for the recipient, it may use direct trust to consider the certificate valid or authoritative independent of any other mechanisms.

This policy has obvious limitations. Domains and DNS can be spoofed, although DNSsec, SSL/TLS server certificates, or IPsec certificates can all easily strengthen this. On the other hand, it is relatively easy to use those security mechanisms to enhance the security of coöperating domains such as business partners, so that each entity's PKI transparently and dynamically interoperates with the other one. Furthermore, as these other security systems build the strength of the overall Internet infrastructure, the security of the Self-Assembling PKI increases, and even with no DNSsec etc., it's a vast improvement on sending messages in the clear.

3.4.4 Local Trust Policies

It is also worth mentioning that any given domain and set of messengers can design their part of the PKI with other small tweaks and policies.

4 Look Before You Leap

Clausewitz said that no battle plan survives contact with the enemy. Similarly, no security policy survives contact with the actual users. Consequently, it is imperative to be able to test policies before they're implemented. A component in the PKI can be left on its own to do things such as create certificates, but before it actually uses them it should be possible to test to see what should be happening.

This is a vital feature of the Self-Assembling PKI which we call "learn mode." Its goal is to speed deployment, and people will deploy slower what they understand less. Consequently, it improves the overall security of the whole system to help administrator understand what they are installing and be able to stay informed while it runs.

Learn mode also permits the first contact a change in policy has with the live network to be benign.

Many policies sound good, but have obvious or inobvious drawbacks. There is perhaps not a person using email who has not thought aloud, "Hmmm, if all incoming email had to be signed, then I wouldn't be getting all that spam." Few organizations can actually live with such a policy. Getting a report on exactly how many important messages would have been bounced by such a policy could be an eye-opener. Administrators smile at the thought of being able to produce a report proving just how silly a policy change would be.

5 Bootstrapping the PKI

Here is an example of how a Self-Assembling PKI might be integrated into an existing messaging infrastructure.

Consider an organization, a.com, that installs a Self-Assembling PKI, implemented as proxy server between the domain's users and their existing email server, using SSL between it and the users.

The proxy server observes connections between the users and the email server as it proxies them. When it sees an authenticated connection, it checks its certificate database for a certificate for that user. If one does not exist, it generates a key and creates a certificate, storing it in the database. The information in that certificate is updated with other information in proxied messages. For example, an authenticated SMTP message contains the common name of the user in message.

Once the user's certificate has been created, incoming mail for that user can be encrypted. The certificate can also be published in a directory or given to other servers.

This same simple process continues for all the users on this server.

The proxy server also updates the validity dates on the certificates it creates, keeping the ones in use with valid dates.

6 Handling Outside Users

The last problem that any PKI designed for improving deployment needs to address is how to communicate with people who are not part of the PKI at all. Ideally, there are mechanisms in the system to handle this, as it helps the PKI to grow even further.

When the robot messenger cannot find a certificate for the recipient, it has a number of policy options. Least interesting, it can send the message anyway, in plain text. Only slightly less interesting, it could bounce the message back to the sender (this is one of those policies that is tempting, and cries out for being used just for the amusement or education value).

Beyond these two simple, obviously inadequate policies, the messenger has two more sophisticated options:

6.1 Smart Trailers

A number of systems that add in security enhancements to messaging, such as automatic virus scanners, frequently add a trailer to the message stating that it has been scanned. Similar to that is a policy that called the smart trailer. A smart trailer says something similar to:

```
-----
Do you know that email messages are as visible as postcards?  This message was
delivered in the clear for anyone to see, but could have been sent securely.
If this concerns you, click this url to find out how you can have your email
delivered to you fully encrypted:  <https://sapki.xyz.tld/secure-delivery.html>
```

The specified URL points back to one of the messenger servers, which explains how the PKI works, gives references to available software of all sorts, and provides a form into which a certificate may be placed. This certificate then becomes part of PKI and the next message to that person will be encrypted and formatted accordingly.

6.2 Boomerang Mail

Boomerang mail is a procedure for securely delivering a message and attachments to someone outside the PKI. If the messenger's policy calls for a boomerang message, it stores the actual message in a secure spot, and sends to the recipient a separate message similar to this:

```
Date:   Mon, 6 Jan 2003 18:25:58 -0800
From:   Jon Callas <jon@pgp.com>
To:     Important Person <vip@host.tld>
Subject:Important Secure Message


Jon Callas would like to send you an important message that should not be
delivered in plain text.  You may receive this message securely through your
web browser.  Simply click this link to receive it and follow the directions:

<https://boomerang.pgp.com/AVERYLONGURLSYNTHESIZEDFROMASECUREHASHFUNCTION>
```

The URL in the boomerang message allows the recipient to retrieve their message and any attachments. It also has links to the same information that a smart trailer points to, with the same options to supply a certificate, or even merely a password for further boomerang messages. Depending on the enthusiasm of the implementors, it may even have a webmail system with it.

There is, however, one last detail that must be solved. How do recipients authenticate themselves to the messenger to receive their message?

The simplest, yet least secure mechanism is one that we call *"first time good."* This mechanism has no password and simply relies on the fact that the vast majority of email is not read or intercepted

in transit. Like all policies, there are situations where this is adequate, and situations where it is not.

A more secure mechanism not only sends the boomerang message to recipient, but a second message to the sender. This extra message send to the sender contains a one-time, synthesized password that the recipient will need to retrieve their message. How that password gets to the recipient is left as an exercise for the sender, but phone calls, SMS, and carrier pigeons are all options. We call this *"out-of-band authentication."*

In either case, however, the recipient can specify a password to use for further messages, or give the messenger a certificate to use in the future. In some cases, the web server may even have downloadable software to further help spread the PKI.

7 Risks and Limitations

There are, as mentioned above a number of tradeoffs in this system to achieve its goal of widespread deployment.

- The authentication to the PKI and its systems is relatively weak; typically, it is merely the password that a user normally uses to authenticate the message system. While it's possible that this could be augmented with stronger authentication, the vast majority authentications are made with nothing stronger than SSL.
- The certificates and keys used by the messengers are held by them, perhaps with the users as well, but are nonetheless not entirely under the users' control. Depending on the implementation, these may be protected with key-management hardware, but they may be used in a completely software system.
- The certificates in such a PKI are thus necessarily low-valued. Perhaps this is more of an observation than anything more, but the CAs in such a PKI should note that these certificates have a weak semantic meaning. It would be a mistake to use them, in purchases of real estate, for example.
- This system provides no help to its users to assuring they are sending a message to the right person. In the so-called "John Wilson Problem," an organization has several members all named John Wilson, and they commonly receive each others' email. The Self-Assembling PKI dutifully, securely delivers mis-addressed messages to the wrong person. It is also vulnerable to weaknesses within the Internet infrastructure such as spoofed domains and hosts.

Nonetheless, in spite of these drawbacks, we are augmenting a system that would otherwise deliver messages in plaintext with the same authentication. There is a risk that the users of the system may believe it to be more secure than it is, but in the world of message security, we are not faced with message systems unimplemented because of security concerns. We are instead faced with message systems that are deployed without security. Even worse, there are many cases of users

migrating to less secure systems such as email on handhelds and instant message systems because of the convenience and utility of these systems. [IM1, IM2]

8 Conclusion

The Self-Assembling PKI is a collection of technologies, strategies, and policies with a goal toward spreading deployment of secure messaging. It also exploits a shift in metaphor from a telephone model to a postal model. This new metaphor gives a new way of examining the issues of creating a PKI that inspires us to create the PKI in such a way that it is easy to use, easy to deploy, and easy to maintain.

References

[CHADWICK] Chadwick, D.W. *Deficiencies in LDAP When Used to Support PKI.* Communications of the ACM, March 2003, pages 99–104.

[GARTNER] Graff, J. The Gartner Group, private communication.

[GUTMANN] Gutmann, P., *PKI: It's Not Dead, Just Resting* IEEE Computer, August 2002, pages 41–49.

[IM1] Festa, P. *Business: IM is getting out of control* ZDNet UK, 26 April 2001, http://news.zdnet.co.uk/story/0,,t269-s2085865,00.html

[IM2] Hu, J., *IM: From fad to big business and beyond* CNet News.com, 13 March 2002, http://zdnet.com.com/2100-1104-992391.html

[JOHNNY] Whitten, A., Tygar, J.D., *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0* Usenix Security Symposium, 1999.

[OPENPGP] Callas, J., Donnerhacke, L., Finney, H., Thayer, R. *OpenPGP Message Format.* RFC 2440, November 1998, http://www.ietf.org/rfc/rfc2440.txt

[PFS] Back, A., Brown, I., Laurie, B., *Forward Secrecy Extensions to OpenPGP* November 2000, http://www.apache-ssl.org/openpgp-pfs.txt

[PGPUSE] PGP Corporation internal survey of its own organizational customers, 2002–2003.

[RIVEST] Rivest, R., *Can We Eliminate Certificate Revocation Lists?* Proceedings of Financial Cryptography '98; Springer Lecture Notes in Computer Science No. 1465 (Rafael Hirschfeld, ed.), February 1998, pages 178–183.

[SPKI] Ellison, C., *SPKI Requirements.* RFC 2692, September 1999, http://www.ietf.org/rfc/rfc2692.txt

Ellison, C., Frantz, B., Lampson, B., Rivest, R., *SPKI Certificate Theory.* RFC 2693, September 1999, http://www.ietf.org/rfc/rfc2693.txt

[STUBBLEBINE] Stubblebine, S. *Recent-Secure Authentication: Enforcing Revocation in Distributed Systems* Proceedings of the1995 IEEE Symposium on Research in Security and Privacy, Oakland, May, 1995, pp. 224-234., http://www.stubblebine.com/95oak.pdf

Intrusion-Tolerant Password-Enabled PKI[§]

Xunhua Wang
Commonwealth Information Security Center &
Department of Computer Science
James Madison University
Harrisonburg, VA 22807 USA
wangxx@jmu.edu

Abstract

Password-enabled PKI facilitates the private key management by integrating easy-to-use passwords into PKI. In the first PKI research workshop, Sandhu *et al.* categorized password-enabled PKI schemes as virtual soft tokens and virtual smartcards [26]. Compared to the conventional PKI, password-enabled PKI introduces a security-critical server where large number of password-related credentials are stored. The compromise of this server will render these password-based credentials susceptible to the dictionary attack and, thus, damage the security of numerous private keys. In this article, using multiple servers, we propose an intrusion-tolerant virtual soft token scheme and an intrusion-tolerant virtual smartcard scheme. In our schemes, compromising up to a threshold number of these servers will not help an attacker mount a dictionary attack and, compared to previous work, our schemes can still function in the presence of some server failures. The multiple servers introduced in our intrusion-tolerant password-enabled PKI can be easily managed and PKI users can roam with human memorable passwords.

Keywords: Password-enabled PKI, Intrusion Tolerance, Virtual Soft Token, Virtual Smartcard

1 Introduction

In the conventional public key infrastructure (PKI), the private key of a public/private key pair is held by an end user (for digital signature or decryption) while the public key is certified as a digital certificate by a trusted third party. Ideally, the private key is stored in a smart card and should never leave the card when it is used. The user is capable of roaming easily with the smart card. However, PKI-based smartcards have not happened in the real world yet. Passwords, on the other hand, are commonly used for authentication in our daily lives and support user roaming very well. For instance, people have been using passwords for remote authentication over the Internet. To integrate passwords' convenience into PKI, two different approaches, called *virtual soft token* and *virtual smartcard*, have been proposed [25, 21, 26].

In the virtual soft token PKI [25, 21], a password is used to encrypt the private key of a public/private key pair and the encrypted private key is stored on a server. With his password, a user can remotely authenticate himself to the server, establish an authenticated and cryptographically strong session key (thus, a secure connection) with the server, download the encrypted private key via the secure channel, decrypt it and use the private key as in the conventional PKI activities. The first step of this approach authenticates a user before he can download a password-encrypted private key and the second step establishes a session key to protect the subsequent downloading of the password-encrypted private key since it is vulnerable to the dictionary attack [24]. These two steps can be accomplished by a *password-authenticated key exchange (PAKE)* protocol [2, 18, 31], in which a client (user) with a password and a server storing the related password verification data (PVD) can authenticate each other and establish a cryptographically strong session key to protect subsequent communication.

In the virtual smartcard PKI [26], an end user's pri-

[§]This research is supported in part by a grant from the Virginia Commonwealth Technology Research Fund (SE 2001-1) through the Commonwealth Information Security Center, James Madison University.

vate key is split into two parts, a *human memorizable password* and a *secret component*. The end user holds the password and the secret component is stored on a server. Let (N, e) be a RSA public key and d is the corresponding private key ($d \times e = 1 \bmod \phi(N)$ and ϕ is the Euler function). In [26], d is split into a password-derived value d_1 and another value d_2, $d = d_1 \times d_2 \bmod \phi(N)$, and d_2 is stored on a server. Note that $m^d = (m^{d_1})^{d_2} \bmod N = (m^{d_2})^{d_1} \bmod N$. To perform a cryptographic operation (digital signature or decryption) on a message m by the user's private key, the end user first authenticates himself to the server using the password and establishes an authenticated session key (thus, a secure connection) with the server. (Again, this can be accomplished by a PAKE protocol.) After securely receiving m from the user, the server applies the secret component, d_2, to m to get a partial result $c_2 = m^{d_2} \bmod N$. c_2 is then passed back to the user through the secure channel. In the end, the user derives d_1 from his password and computes the final result as $c = c_2^{d_1} = m^{d_2 \times d_1} = m^d \bmod N$. Note that, in the above process, the overall value of the private key, d, is never reconstructed on the client nor on the server. Both the virtual soft token and virtual smartcard allow a user to *roam* with a memorizable password solely and digitally sign a message (with a long-term private key) at a new location.

The problem. People tend to choose easily memorizable passwords (from a dictionary) and thus, password-based systems are notoriously vulnerable to the *dictionary attack* [24], in which an attacker does not brute-force all possible passwords but rather work on a much smaller dictionary of likely passwords *. Compared to the conventional PKI, password-enabled PKI introduces a security-critical server where both password-verification data (PVD) and password-related credentials (password-encrypted private keys in the virtual soft token PKI and secret components in the virtual smartcard PKI) are stored. This makes it subject to the *server compromise-based dictionary attack*: after breaking into this server and stealing the password verification data and password-based credentials, an attacker can mount dictionary attacks to find the password and recover the private key.

For the virtual soft token, if the server is compromised and the password-encrypted private key is stolen, an attacker can guess a password, use it to decrypt the stolen credential and verify the correctness of the guessing by checking the decrypting result with the corresponding public key ((N, e) for RSA). If an attacker also steals the password-verification data, the attack will be simpler: the attacker can simply guess a likely password, compute the corresponding PVD and compare it against the stolen PVD. If he observes a match, then he finds the password and it can be used to decrypt the stolen password-encrypted private keys.

As for the virtual smartcard PKI, if the server is compromised and d_2 is stolen, an attacker can simply guess a likely password, derive d_1' from it, pick a random m $(1 < m \leq (N-1))$, compute $c' = m^{d_1' \times d_2} \bmod N$, and verify the correctness of the guessing by checking if $m = c'^e \bmod N$ holds. In this case, an attacker can mount dictionary attacks against thousands, if not millions, of users' password-protected private keys stored on the corrupted server. If an attacker also steals the password-verification data (PVD), the attack will be simpler: the attacker can simply guess a likely password, compute the corresponding PVD and compare it against the stolen PVD. If he observes a match, then he finds the password, which can be used with the stolen d_2 to recover the private key. It is worth noting that, compared to the virtual soft token PKI, the dictionary attack against the virtual smartcard PKI is more subtle. An attacker does not need d_2 to mount an off-line dictionary attack and m^{d_2} for any public message m will be sufficient: if an attacker obtains the value of $c_2 = m^{d_2} \bmod N$ for some m, he can simply guess a likely password, derive d_1' from it, compute $c' = c_2^{d_1'} \bmod N$, and verify the correctness of the guessing by checking if $m = c'^e \bmod N$ holds.

Proactive password checking mitigates the dictionary attack problem but it does not fully solve it. Wu [32] showed that a proactive password checking system still allows about 8.28% of its passwords susceptible to the dictionary attack. Considering the large number of password-verification data and password-related credentials stored on the server, this server compromise-based dictionary attack could be large-scale and catastrophic. We argue that intense monitoring of the server may not be sufficient and server compromise (by outside attackers, inside attackers or through the mistakes of honest insiders) seems inevitable. For instance, an attacker might gain the `root` privilege of the server by exploiting bugs in server software (for instance,

*For a specific user, his password may not always fall within an attacker's dictionary. But an alarmingly high fraction of the actual passwords match passwords in a constructed dictionary [20].

through bugs [6, 7] in the Wu-FTP ftp server, bug [10] in the Apache web server, bug [8] in Microsoft IIS web server, and bug [9] in Kerberos server). It is our belief that, even though people have worked hard to fix these known bugs, root privilege-leaking bugs will not disappear since new bugs are being discovered continuously.

To avoid this large-scale dictionary attack, it makes sense to distribute the functionality of one server to multiple servers to tolerate intrusions [13]. Sandhu et al. [26] observed that a multiple-server scheme may degrade operational quality and is vulnerable to the common-mode failures — once an attacker knows how to break one server, likelihood of success on the other is quite significant in practice. In this paper, we believe that the common-mode failure can be significantly mitigated by the system diversity [14] — including both hardware diversity, operating system diversity and application software diversity — and breaking into one server will not necessarily increase an attacker's chance to break another server with diverse systems (hardware, operating system and software). In this way, introducing multiple servers using different hardware and software and distributing a secret component among these servers, if done properly, will significantly improve the security against the server compromise-based dictionary attack.

The main results. The contribution of this paper includes an intrusion-tolerant virtual soft token PKI scheme and an intrusion-tolerant virtual smartcard PKI scheme.

In our virtual soft token scheme, a password-encrypted private key, together with the password-verification data, is shared among n servers ($n > 1$). Compared to the multiple-server virtual soft token scheme given in [16], our intrusion-tolerant virtual soft token PKI scheme is threshold: any t ($t \leq n$) or more of these servers can collectively authenticate a user (using the shared PVD) and let the user *securely* download his password-encrypted private key shares **without reconstructing the shared PVD at any single location and the shared password-encrypted private key on any single server**. Any subset of size less than t of these n servers can *not* reconstruct either the *shared* password-encrypted private key or the *shared* PVD, hence tolerating intrusions against the servers.

In our virtual smartcard PKI scheme, the secret component of a private key, together with the password-verification data, is shared among the n servers. Any t ($t \leq n$) or more of these servers can collectively authenticate a user (via the shared PVD) and help the authenticated user securely perform a digital signature **without reconstructing the shared secret component and the shared PVD at any single location**. Corruption of any less than t of these servers will *not* help an attacker to get the secret component to mount a dictionary attack.

The key idea behind our intrusion-tolerant virtual soft token PKI is the application of an intrusion-tolerant password-authenticated key exchange (PAKE) protocol and the idea behind our intrusion-tolerant virtual smartcard PKI is the composition of an intrusion-tolerant PAKE with a password-adapted threshold cryptography scheme.

This article is organized as follows. Section 2 reviews some related work and Section 3 describes two building blocks for our intrusion-tolerant password-enabled PKI schemes. In Section 4 we present an intrusion-tolerant virtual soft token scheme and an intrusion-tolerant smartcard scheme, both of which are secure against the server compromise-based dictionary attack. Section 5 discusses some operational issues. Concluding remarks are given in Section 6.

2 Related Work

The concept of password-authenticated key exchange (PAKE) protocol was first developed in [2] and then studied in [3, 18, 31, 5, 1]. Perlman and Kaufman [25] applied the PAKE protocols and proposed the idea of virtual soft token. To resist the server compromise-based dictionary attack, Ford and Kaliski [16] proposed the first multiple-server approach for the virtual soft token PKI. However, it requires **all** of the multiple servers present when the user retrieves the distributively stored credential. This significantly degrades the availability of the resulting system — if one server goes down, the service provided will not be available. Jablon [19] improved the scheme of [16] but it still retains the all-server-present requirement. MacKenzie et al. [23] proposed the first *threshold* PAKE and, in our earlier work [30], we also proposed a threshold PAKE, which is used in this article to build our intrusion-tolerant password-enabled PKI.

Kwon [21] proposed a virtual soft token scheme where multiple servers are used. Compared to our intrusion-tolerant virtual soft token, the scheme of [21] used the *non-threshold* RSA given in [4] and thus, required *all* server to be available when a user retrieves its private keys.

To build the virtual smartcard scheme, Sandhu et al. [26] used a password-adapted 2-out-of-2 distributed RSA digital signature scheme given in [4], which is sequential and non-threshold. In contrast, in this article, we adapt the threshold RSA scheme proposed in [28] and use it to build the intrusion-tolerant virtual smartcard.

3 The Building Blocks

As stated earlier, we use an intrusion-tolerant password-authenticated key exchange (PAKE) protocol and a password-adapted threshold RSA as building blocks in our constructions of intrusion-tolerant password-enabled PKI. The intrusion-tolerant PAKE shares a PVD among multiple servers and is used in both the intrusion-tolerant virtual soft token scheme and the intrusion-tolerant virtual smartcard scheme. The password-adapted threshold RSA, on the other hand, shares a secret component among multiple servers and is used in the intrusion-tolerant virtual smartcard scheme. In this section, we will give the details of these two building blocks.

In the remainder of this paper, n is used to denote the number of the multiple servers. These n servers are numbered from 1 to n and are called server 1, 2, ..., n. We assume that there exist secure connections between these n servers, which can be implemented in Secure Socket Layer (SSL) [15]. Let $\hat{N}$ be a safe prime, $\hat{N} = 2\hat{q} + 1$ where $\hat{q}$ is also a prime. $\hat{g}$ is an element of finite field $F_{\hat{N}}$ with order $\hat{q}$. $(\hat{N}, \hat{q}, \hat{g})$ are system parameters for a PAKE (see the Appendix). For a set S, $a \in_R S$ means that element a is randomly and uniformly selected from S. $|S|$ denotes the cardinality (the size) of S. For two integers a_1 and a_2, $[a_1, a_2]$ denotes the set of integers x satisfying $a \leq x \leq b$. $gcd(a_1, a_2)$ denotes the greatest common divisor of a_1 and a_2.

3.1 An intrusion-tolerant PAKE

In a PAKE, a user possesses a password and the server stores a related password verification data (PVD). Using what they have, the user and the server can perform a password-authenticated key exchange protocol and establish an authenticated (and cryptographically strong) session key, which can be used to protect subsequent communication between the user and the server[†].

However, since the password-verification data stored on the server is derived from a password using a publicly known function, if an attacker manages to compromise the server and steal the PVD, he can still mount an off-line dictionary attack by just computing PVDs value with all likely passwords and comparing them with the stolen PVD. (If he observes a match, then the correct password is found.) The intrusion-tolerant PAKE developed in our earlier work [30] can be used to improve security against this attack, in which a PVD is shared among these multiple servers and is never reconstructed during a PAKE running. Each user of the intrusion tolerant PAKE registers himself with the servers in the user enrollment phase, during which the user's PVD, x, is shared, using a (t, n)-Shamir secret sharing [27][‡], among the n servers. Let $x \stackrel{(t,n)}{\longleftrightarrow} (x_1, x_2, \ldots, x_n)$ denote the secret sharing and each server i has PVD share x_i. Then, the user can remotely authenticate himself to Γ, a subset of these multiple servers, $|\Gamma| \geq t$, and establish a session with each of them *without reconstructing the shared PVD*. Any attacker who has compromised less than t servers will get no information about the shared PVD and, thus, cannot mount a dictionary attack. These servers can proactively update their PVD shares while keeping the shared PVD unchanged to further enhance their security. A user can also change his password as in normal password-based systems. The details of this intrusion-tolerant PAKE of [30]

[†] Just as passwords are always subject to the dictionary attack, a PAKE is subject to network-based dictionary attacks, including eavesdropping-based dictionary attack and active dictionary-based protocol attacks. Existing PAKE protocols such as EKE [2], SPEKE [18], SRP [31], provide either heuristic or provable security against network-based dictionary attacks.

[‡] A (t, n)-Shamir secret sharing splits a secret x into n secret shares x_i, $1 \leq i \leq n$, such that any t or more of these secret shares can be used to reconstruct x while any less than t secret shares could not. Shamir secret sharing is perfect in that any less than t secret shares leak no information about x.

is summarized in the appendix of this paper.

3.2 A password-adapted threshold RSA

Threshold cryptography researches on how to share a (cryptographically strong) private key among multiple parties and how a subset of these parties can perform a cryptographic computation without reconstructing the shared private key [11, 12]. Here, we integrate password into the threshold RSA given in [28] to obtain a password-adapted threshold RSA scheme and then, use it to build an intrusion-tolerant virtual smartcard. The password-adapted threshold RSA scheme is given below.

The key generation. A user picks his password $\hat{p}$ and a value d_1 is derived from $\hat{p}$ using a public function (the PBKDF2 function of [22] can be used for this purpose). Let $\Delta = 1 \times 2 \times \ldots \times n = n!$ (n is the number of servers). (N, e) is the user's RSA public key where $N = p \times q$; p, q are two primes; e is a prime, $4\Delta < e < \phi(N)$, $\phi(N) = (p-1) \times (q-1)$. d is the user's overall RSA private key, $1 < d < \phi(N)$. $d \times e = 1 \bmod \phi(N)$. d is split into d_1 and d_2 and d_2 is computed as follows: $1 < d_2 < \phi(N)$, $d_1 + d_2 = d \bmod \phi(N)$. d_2 is further shared among the n servers as follows: let $a_0 = d_2$, $a_i \in_R [0, \phi(N)-1]$ for $1 \leq i \leq (t-1)$; define $f(x) = \sum_{i=0}^{t-1} a_i x^i \bmod \phi(N)$; then, one can compute $d_{2i} = f(i) \bmod \phi(N)$ for $1 \leq i \leq n$ and server i is assigned d_{2i}, $1 \leq i \leq n$.

Observation. In the above key generation process, p and q are ordinary primes, as opposed to the safe primes in [28]. d_2 is picked as $(d - d_1) \bmod \phi(N)$ for efficiency reasons. In this way, the user and the servers can perform the cryptographic computations in parallel. One can also compute d_2 as $d_1 \times d_2 = d \bmod \phi(N)$, as did in [26]. In this case, the computations of the user and the servers are sequential.

Digital signature. Let Γ be the subset of the servers who will help a user digitally sign a message m, $|\Gamma| \geq t$. Let a, b be integers satisfying $4\Delta a + eb = 1$, which can be computed by the extended GCD algorithm [29]. The user derives d_1 from his password and computes $c_1 = m^{4\Delta d_1} \bmod N$. In parallel, each server $j \in \Gamma$ computes $c_{2j} = m^{2d_{2j}} \bmod N$ and sends c_{2j} to the user.

After receiving all c_{2j}, the user computes $c_2 = \prod_{j \in \Gamma} {c_{2j}}^{2\lambda_{j,\Gamma}} \bmod N$, where $\lambda_{j,\Gamma} = \Delta \times \prod_{k \in \Gamma, k \neq j} \frac{k}{k-j}$. He then combines c_2 and c_1 into $\omega = c_2 \times c_1 \bmod N$ and computes y as $y = (\omega^a \times m^b) \bmod N$. Note that $\omega = m^{4\Delta d} \bmod N$ and $y^e \bmod N = m^{4\Delta dae + be} = m^{4\Delta a + be} = m \bmod N$. That is, y is the digital signature of m by the private key d.

4 Intrusion-tolerant Password-enabled PKI

4.1 Intrusion-tolerant virtual soft token

Just as a virtual soft token is the composition of a PAKE and a secure download of the password-encrypted private key, an intrusion-tolerant virtual soft token scheme is implemented as the composition of an intrusion-tolerant PAKE and multiple secure downloads of the password-encrypted private key *shares*. In an intrusion-tolerant virtual soft token scheme, for each user, his PVD is shared among the n servers using a (t, n)-Shamir secret sharing over finite field $F_{\hat{q}}$. The user's password-encrypted private key is also shared among the same n servers using a (t, n)-Shamir secret sharing.

When a user needs to use his private key, he first runs the intrusion-tolerant PAKE protocol with Γ, a subset of these multiple servers, $|\Gamma| \geq t$, $\Gamma \subseteq \{1, 2, ..., \ldots, n\}$ (see Section 3.1 and the Appendix section for more details). Afterward the user will have one authenticated session key (thus, one secure connection) with each of the servers in Γ. Then, these subset of servers will send their password-encrypted private key shares to the user (via the secure connections). The user reconstructs the password-encrypted private key, decrypts it with the password and uses the private key as in the conventional PKI.

Remark. In the above virtual soft token scheme, neither the shared PVD nor the password-encrypted private key is reconstructed at any single server. The minimal number of servers required for a user login is $(2t - 1)$, $t \leq n$.

4.1.1 The parameter selection

In the above intrusion-tolerant virtual soft token, both the PVD and the password-encrypted private key are shared among the multiple servers. The sharing of PVD is performed in the finite field $F_{\hat{q}}$. The sharing of the password-encrypted private key can be operated in another finite field $F_{\bar{q}}$ where $\bar{q}$ is another prime. Note that the size of d is in the same order as the size of N and, the size of the encryption of d by a password (say, using the PKCS # 5 standard [22]) will *not* increase significantly. (If PKCS #5 is used to encrypt the private key, the salt and iteration count can be simply replicated to each server. And only the encryption of d is shared). Thus, the size of $\bar{q}$ should be no less than the size of N. Another option is to use $\hat{q}$ as $\bar{q}$. If this is the case, $\hat{q}$ should be no less than N.

4.2 Intrusion-tolerant virtual smartcard PKI

Just as a virtual smartcard [26] is the composition of a PAKE and a password-adapted non-threshold distributed RSA, our intrusion-tolerant virtual smartcard scheme is the composition of an intrusion-tolerant PAKE and a password-adapted threshold RSA. In the intrusion-tolerant virtual smartcard scheme, a user's PVD (related to password $\hat{p}$) is shared among the n servers using a (t, n)-Shamir secret sharing scheme. The user's RSA private key d is split as a password-derived value d_1 (derived from password $\hat{p}$) and d_2. d_2 is further shared as $d_2 \stackrel{(t,n)}{\longleftrightarrow} (d_{21}, d_{22}, \ldots, d_{2n})$. In addition to its PVD share, a server, i, $1 \leq i \leq n$, also holds d_{2i}.

When a roaming user wants to digitally sign a message, m, he first runs the intrusion-tolerant PAKE protocol with Γ, a subset of the multiple servers, $|\Gamma| \geq t$, $\Gamma \subseteq \{1, 2, \ldots, n\}$, and establishes an authenticated session key (thus, a secure connection) with each of them. Then, the user sends m, via the secure connections, to server j, $j \in \Gamma$. Server j computes $c_{2j} = m^{2d_{2j}} \bmod N$ and sends it back to the user through the secure channel. The user computes ω and y as described in Section 3.2, where y is the digital signature of m by the user's private key d.

Remark. In the above virtual smartcard scheme, none of the shared PVD, d_2 and m^{d_2} is reconstructed at any single server. The minimal number of servers required for a user login is $(2t-1)$, $t \leq n$.

5 Some Operational Considerations

Operational quality is a big concern for the intrusion-tolerant password-enabled PKI since introducing multiple servers increases the operational complexity [26]. However, we can automate the management to minimize the manual management overhead.

5.1 The user enrollment

Compared to the virtual soft token [25, 26], at the user enrollment phase, our intrusion-tolerant virtual soft token scheme introduces one additional step: the share generations of the user's PVD and password-encrypted private key and the share distribution to the multiple servers. This additional step can be fully automated by a management server, which performs the Shamir secret sharing on the PVD and the password-encrypted private key and, then *securely* sends, via SSL, these shares to the multiple servers.

Similarly, compared to the virtual smartcard [26], at the user enrollment phase, our intrusion-tolerant virtual smartcard also introduces one additional step: the share generations of the user's PVD and his secret component (d_2) and the share distribution to the multiple servers. We can also automate this step by using a management server, which performs the Shamir secret sharing on the PVD and the secret components and, *securely* sends, via SSL, these shares to the multiple servers.

Thus, our intrusion-tolerant password-enabled PKI schemes do not bring much operational overhead to the user enrollment stage.

5.2 User authentication

In the intrusion-tolerant password-enabled PKI schemes, when a user interacts with the servers to use his private key, he just needs to type in his password and all other steps are automatically performed by programs. Thus, our intrusion-

tolerant password-enabled PKI schemes do not increase user's operational complexity.

5.3 Password change

In a password-enabled PKI, a user may want to change his password while keeping his long-term private key unchanged.

In the intrusion-tolerant virtual soft token PKI, a change in the password requires the update of the corresponding PVD shares and the update of the password-encrypted private key shares (the private key remains unchanged but its encrypted form by the password should be updated accordingly). The intrusion-tolerant PAKE used in this article (see Section 3.1) allows a user to securely change his password and update the corresponding PVD shares stored on each server. This naturally enables the password change in our intrusion-tolerant virtual soft token: a user first runs the intrusion-tolerant PAKE protocol, securely downloads the password-encrypted private key shares, reconstructs the password-encrypted private key, decrypts it with the old password, re-encrypts it with the new password, generates a (t, n)-Shamir secret shares and securely uploads these new shares to the multiple servers respectively; then, he can run the intrusion-tolerant PAKE password change protocol given in [30] to update the PVD shares stored on each server.

In a virtual smartcard PKI, the change of a user's password causes the change of d_1 and thus, requires the update of the secret component, d_2, since d remains unchanged and $d_1 + d_2 = d \bmod \phi(N)$. As $\phi(N)$ is unknown to the user and the server, the change of d_2 is difficult unless the shared d is reconstructed and $\phi(N)$ is recovered. Technically, after authenticating himself to the server and establishing a session key, a user can securely download d_2 from the server, reconstruct d and recover $\phi(N)$, compute the new d_2 (from the new d_1 and the recovered $\phi(N)$), and update this new d_2 to the server. However, this process is computation-intensive and looks awkward. Indeed, in the only virtual smartcard PKI scheme proposed in [26], password change is not discussed. This difficulty remains in our intrusion-tolerant virtual smartcard PKI and is the topic of our future research.

6 Conclusion

Password-enabled PKI, including the virtual soft tokens and virtual smartcards, facilitates the private key management by integrating easy-to-use passwords into PKI. However, compared to the conventional PKI, password-enabled PKI introduces a security-critical server where large number of password-related credentials are stored. The compromise of this server will render these password-based credentials susceptible to the dictionary attack and, thus, damage the security of numerous private keys.

To address this attack, using multiple servers, we proposed an intrusion-tolerant virtual soft token PKI scheme and an intrusion-tolerant virtual smartcard PKI scheme. In our schemes, compromising up to a threshold number of these servers will not help an attacker mount the dictionary attack and the intrusion-tolerant password-enabled PKI schemes can still function in the presence of some server failures. Compared to previous work, our virtual soft token is threshold and does not require all servers when a user needs his private key. We designed our virtual soft token PKI by compositing an intrusion-tolerant PAKE with a secret sharing. Our intrusion-tolerant virtual smartcard PKI is achieved through the composition of an intrusion-tolerant PAKE with the password-adapted threshold RSA. The multiple servers introduced in our intrusion-tolerant password-enabled PKI can be easily managed and PKI users can roam with human memorable passwords.

Acknowledgement

The author is indebted to Samuel Redwine for the discussions and Hua Lin for reviewing the earlier drafts of this paper. The author also wishes to thank the anonymous reviewers for suggestions to improve this article.

References

[1] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances*

in Cryptology - EUROCRYPT 2000, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 2000.

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

[3] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: a password–based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 244–250, 1993.

[4] C. Boyd. Digital multisignatures. In H. Beker and F. Piper, editors, *Cryptography and coding*, pages 241–246. Clarendon Press, Royal Agricultural College, Cirencester, December 15–17 1989.

[5] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171, Bruges, Belgium, May 2000.

[6] CERT. CERT advisory CA-1999-13 multiple vulnerabilities in WU-FTPD. Available from http://www.cert.org/advisories/CA-1999-13.html, October 19 1999.

[7] CERT. CERT advisory CA-2001-33 multiple vulnerabilities in WU-FTPD. Available from http://www.cert.org/advisories/CA-2001-33.html, November 29 2001.

[8] CERT. CERT advisory CA-2002-09 multiple vulnerabilities in Microsoft IIS. Available at http://www.cert.org/advisories/CA-2002-09.html, April 11 2002.

[9] CERT. CERT advisory CA-2002-29 buffer overflow in Kerberos administration daemon. Available at http://www.cert.org/advisories/CA-2002-29.html, October 25 2002.

[10] CERT. Vulnerability note VU#124003 apache HTTP server on Win32 systems does not securely handle input passed to CGI programs. Available at http://www.kb.cert.org/vuls/id/124003, April 11 2002.

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

[12] Y. Desmedt. Threshold cryptography. *European Trans. on Telecommunications*, 5(4):449–457, July-August 1994. (Invited paper).

[13] Y. Deswarte, L. Blain, and J.-C. Fabre. Intrusion tolerance in distributed computer systems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 110–122, May 1991.

[14] Y. Deswarte, K. Kanoun, and J.-C. Laprie. Diversity against accidental and deliberate faults. In *Proceedings of the Computer Security, Dependability and Assurance: From Needs to Solutions*, pages 171–181, 1998.

[15] T. Dierks and C. Allen. The TLS protocol version 1.0. Internet RFC 2246, January 1999.

[16] W. Ford and B. Kaliski, Jr. Server–assisted generation of a strong secret from a password. In *Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises(WET ICE 2000)*, pages 176–180, Gaithersburg, MD, USA, June 14–16 2000.

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

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

[19] D. P. Jablon. Password authentication using multiple servers. In D. Naccache, editor, *Progress in Cryptology - CT-RSA 2001 Proceedings of the Cryptographers' Track at RSA Conference*, volume 2020 of *Lecture Notes in Computer Science*, pages 344–360, San Francisco, CA, USA, April 8–12 2001. Springer-Verlag.

[20] D. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the UNIX Security Workshop II*, August 1990.

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

[22] R. Laboratories. PKCS #5 v2.0 password-based cryptography standard. Available from http://www.rsasecurity.com/rsalabs/pkcs/pkcs-5/, March 1999.

[23] P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange (extended abstract). In M. Yung, editor, *Advanced in Crypto: – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 385–400. Springer-Verlag, August 2002.

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

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

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

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

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

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

[30] X. Wang. *A Distributed Password-Authenticated Key Exchange Protocol Secure Against Server Compromise-Based Dictionary Attacks*. Department of Computer Science, James Madison University, Harrisonburg, VA, USA, January 2003.

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

[32] T. Wu. A real-world analysis of Kerberos password security. In *Proceedings of the 1999 Network and Distributed System Security Symposium*, 1999.

APPENDIX

In this article, the intrusion-tolerant PAKE protocol given in [30] is used as a building block. Let $\hat{N}$ be a safe prime, $\hat{N} = 2\hat{q} + 1$ where $\hat{q}$ is also a prime. $\hat{g}$ is an element of finite field $F_{\hat{N}}$ with order $\hat{q}$. $(\hat{N}, \hat{q}, \hat{g})$ are the system parameters. H denotes a secure hash function such as SHA-1. We use $(\tau_1, \tau_2, \ldots, \tau_n) \xrightarrow{(t,n)} \tau$ to denote the fact that a value, τ, is reconstructed from t or more shares τ_i, $1 \leq i \leq n$.

For this protocol, Table 1 summarizes the password-authenticated key exchange data flow between a user and a particular server, i, $1 \leq i \leq n$, where server i stores the user's PVD share x_i. At the end of this protocol, the user and server i will share a cryptographically strong session key K. This protocol can be repeated between the user and any other server so that the user can share one session with each of the participating servers.

A user shows his ID, I, in the first step, to which server i responds with the user's salt, s. Server i also passes this ID I to a threshold number or more servers (so that they can locate the PVD share for this user). In step 2, the user computes x and l. In step 3, with the assistance of a threshold number or more servers, server i computes a value B (see the following paragraph for details) and passes it back to the user. In this step, the user picks $a \in_R [1, \hat{q} - 1]$, computes A and sends it to server i, which in turns passes A to the participating servers. In step 4, the user checks if the received B satisfies $B = 0 \bmod \hat{N}$ and will quit if this is the case. Otherwise, the user computes a value S_c using the given formula and server i uses the subset of the servers to compute value S_s (see the following paragraph for details). Note that $S_c = S_s$ and this value is denoted as S. Step 5 and 6 are used to verify that the user and server i share a common value S. The authenticated session key, K, is derived from S in step 7.

The server-side computation of step 3 Let Γ be the set of participating servers, $\Gamma \subseteq \{1, 2, \ldots, n\}$,

	Client		**Server** i	Server 1	Server 2	...	Server n
1		$\xrightarrow{I}$	(lookup s)	(lookup x_1)	(lookup x_2)		(lookup x_n)
2	$x = H(s, I, \hat{p})$	$\xleftarrow{s}$					
	$l = x^{-1} \bmod \hat{q}$						
3	$A = \hat{g}^{al \bmod \hat{q}} \bmod \hat{N}$	$\xrightarrow{A}$		**Collectively** generate shares of $b \in_R F_{\hat{q}}$ and			
		$\xleftarrow{B}$		**collectively** compute $B = \hat{g}^{bx \bmod \hat{q}} \bmod \hat{N}$			
4	$S = B^{al \bmod \hat{q}} \bmod \hat{N}$			**Collectively** compute $S = A^{bx \bmod \hat{q}} \bmod \hat{N}$			
5	$M_1 = H(A, B, S)$	$\xrightarrow{M_1}$	(verify M_1)				
6	(verify M_2)	$\xleftarrow{M_2}$	$M_2 = H(A, M_1, S)$				
7	$K = H(S)$		$K = H(S)$				

Table 1: Intrusion-tolerant PAKE

$|\Gamma| \geq t$. B is computed by the participating servers as follows:

1. Using a Joint-Shamir-RSS [17], the participating servers generate $(b_1, b_2, \ldots, b_n)$ $\overset{(\mathbf{2t\text{-}1},n)}{\longleftrightarrow}$ $b \bmod \hat{q}$, where b is a random value **unknown** to any individual server and each server j, $j \in \Gamma$, has share b_j.
2. Each participating server j, $j \in \Gamma$, computes $d_j = b_j \times x_j \bmod \hat{q}$, $B_j = \hat{g}^{d_j} \bmod \hat{N}$, and sends B_j to server i. Note that $(d_1, d_2, \ldots, d_n)$ $\overset{(\mathbf{2t\text{-}1},n)}{\longleftrightarrow}$ $d = b \times x \bmod \hat{q}$ and $(B_1, B_2, \ldots, B_n)$ $\overset{(\mathbf{2t\text{-}1},n)}{\longleftrightarrow}$ $\hat{g}^{bx \bmod \hat{q}} \bmod \hat{N}$.
3. Using the Lagrange interpolation formula in the exponent [17], server i first computes B as follows: $(B_1, B_2, \ldots, B_n)$ $\overset{(\mathbf{2t\text{-}1},n)}{\longrightarrow}$ $B \bmod \hat{N}$. (Note that, $B = \hat{g}^{bx \bmod \hat{q}} \bmod \hat{N}$.) Then, server i sends B back to the user in step 3.

$(2t - 1)$ servers are needed to perform the above steps.

The server-side computation of step 4 The computation of step 4 proceeds as follows:

1. Each participating server checks if $A = 0 \bmod \hat{N}$. They will abort if it is. Otherwise, they check if $A^{\hat{q}} = 1 \bmod \hat{N}$. They will abort the computation if it does not hold. Otherwise, they will continue.
2. Each participating server j, $j \in \Gamma$, computes $S_j = A^{d_j} \bmod \hat{N}$, and sends S_j to server i. Note that $(S_1, S_2, \ldots, S_n)$ $\overset{(\mathbf{2t\text{-}1},n)}{\longleftrightarrow}$ $A^{bx \bmod \hat{q}} \bmod \hat{N}$.
3. Using the Lagrange interpolation formula in the exponent [17], server i computes S as follows: $(S_1, S_2, \ldots, S_n)$ $\overset{(\mathbf{2t\text{-}1},n)}{\longrightarrow}$ $S \bmod \hat{N}$. Note that $S = A^{bx \bmod \hat{q}} \bmod \hat{N}$.

Decentralization Methods of Certification Authority Using the Digital Signature Schemes

Satoshi KOGA * and Kouichi SAKURAI †

* Dept. of Electrical Engineering and Computer Science,
Kyushu University, Japan
satoshi@tcslab.csce.kyushu-u.ac.jp
† Faculty of Information Science and Electrical Engineering,
Kyushu University, Japan
sakurai@csce.kyushu-u.ac.jp

Abstract.

A *Public Key Infrastructure* (PKI) is the one of the important techniques to support secure e-commerce and digital communications on networks. Many PKI trust models have been proposed and are widely used for various purposes.

The trust model that one *Certification Authority* (CA) issues all certificates is the simplest one. It is called *a single CA model*. In this model the certificate verification process is very simple, however it is attended with a danger of the high ratio of exposure CA's private key. While *a subordinated hierarchical model* which is constructed by the multiple CAs can mitigate that risk. For this reason, the distributed CA model like the subordinated hierarchical model is needed in the real world.

This paper discusses the advantages and disadvantages of the general distributed CA models. Especially we investigate in two points: (1) the effects in case that the CA's private key is compromised and (2) the certificate path processing. Then we present the new distributed CA models which the certification path is shorter than one of a subordinated hierarchical model by using *a forward-secure digital signature scheme* and *a key-insulated digital signature scheme*.

Keyword:
Public Key Infrastructure, Trust Model, Forward-Secure Digital Signature Scheme, Key-Insulated Digital Signature Scheme

1. Introduction

1.1 Background and Motivation

The third trusted party is called *Certification Authority* (CA) in *Public Key Infrastructure* (PKI). The CA guarantees by issuing certificates which link public key to identity information. These certificates are signed by CA's private key and certificate verifiers can verify it using CA's public key. In this way, trust relationships between the certificate users are established.

In PKI, the simplest trust model is *a single CA model* which one CA issues all certificates. Certificate verifiers can check all certificates as well as the certificate revocation information by using CA's public key. Like this, the construction and validation of certification path are very efficient and easy in this model.

However, the problems in this model are pointed out from the viewpoints of security and system [1, 3]. The main problem is that all certificates are affected in case of compromise of CA's private key. If CA's

private key is exposed, the CA keys should be revoked, and a new certificate must be issued and distributed to all certificate users. Moreover, any subordinate certificates and keys that were suspect should need to be replaced and the certificate users have to run through the registration process once again. These operations take as much time and effort as the number of certificate users and consequently the entire trust model may collapse [1].

In order to improve this problem, certificate users need to be partitioned and assigned to separate CAs, so compromise of one CA's private key will not affect the certificates issued by the others. In practical, it is necessary to construct *a distributed CA model* using the multiple CAs and one of the distributed CA models which has the hierarchical structure, is called *a subordinated hierarchical model*.

But, in comparison with a single CA model, a subordinated hierarchical model is less efficient in the point of the certificate verification process because certification path is longer. When certificate verifiers check the certification path, they need to get and verify the certificates on that path. In addition, they have to get and verify revocation information about the certificates by issued the CA.

1.2 Certification Path Processing

In order to validate a certificate, a certification path between the certificate and the trust anchor must be established. Moreover, every certificate within that path must be checked. This process is referred to as *certificate path processing* [4].

In general, certification path processing consists of two phases [2, 5]: (1) *path construction* and (2) *path validation* described as follows.

(1) path construction

Valid paths begin with certificates issued by a trust anchor. First, a verifier should discover a path leading from the certificate that is being validated to trust anchor that a verifier trust, it is called *certification path*. The next step is that verifiers need to obtain the certificates on path.

(2) path validation

Verifiers have to check that each certificate in the path is satisfied the following things [4].

1. The certificate must contain a cryptographically valid signature. Namely, certificate verifier can verify that the certificate contents have not been tampered, by using public key of issuer.

2. By checking the validity period, the certificate must be current.

3. The certificate must not have been revoked.

Thus, the verification process of the certificate is the burden when the certification path length is long.

1.3 Our Contribution

In this paper, we discuss the advantages and disadvantages of the general distributed CA models and suggest the decentralization methods absorbing these advantages. Our goal is to mitigate the burden of certificate verifiers. In short, we try to construct the distributed CA model which certification path length is short. Our methods are realized using the digital signature schemes: (1) *a forward-secure digital signature* scheme [6] and (2) *a key-Insulated digital signature scheme* [12]. These signature schemes are usually used generating and updating the user's keys. The idea of our methods is that the multiple CA's private keys are generating by using these signature scheme.

We use these signature schemes to construct a distributed CA model which the certification path is short. In brief, the multiple key pairs (a public key and private key) are generated using these schemes and assigned the distributed CAs. This model enables to mitigate the damage caused by CA's private key exposures because the distributed CAs issue the certificates to users. Moreover the verifiers can check all certificates using one public key (Root CA's public key), so the burden of certification path processing is mitigated.

1.4 Organization of Our Paper

In section 2, we explain the PKI trust models and consider the general distributed CA models. Then we also discuss the advantages and disadvantages of these models. In section 3, we refer to our decentralization methods and concluding remarks are made in section 4.

2. PKI Trust Models

By using multiple CAs, the trust relationships between CAs are built. This model is called *PKI trust model*. In real world, many kinds of trust models exist and are utilized according to various environments.

2.1 A Single CA Model

This model is the simplest trust model and consists of a single CA which one CA issues all certificates (Figure 1). Because of simple structure, this model has advantage that certification path processing is very easy. However, this model has the following problems.

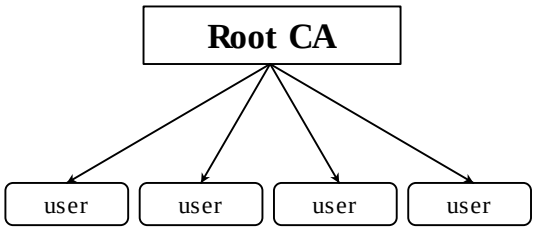

Figure 1: A Single CA Model

(problems)

1. Since only one CA issues all certificates, the burden of CA system become heavy in proportion to the number of certificate users.
2. When CA's private key is compromised, all certificate users are affected.
3. If the company locations are separate widely, this model is inconvenient to issue the certificates to all users.

2.2 Distributed CA Models

Taking into account the problems in a single CA model, the trust model that consists of multiple separate CAs is needed in the real world. We call this model *a distributed CA model*. Since the certificate users are assigned to multiple CAs, the load of system can be mitigated in this model. We try to construct the distributed CA models which has n CAs (Figure 2).

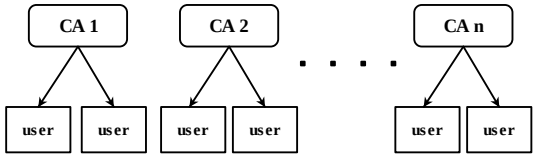

Figure 2: A Distributed CA Model

2.2.1 A Method of Using Cross-Certification

In Figure2, verifiers who trust one CA can not verify the certificates issued by the others, because certification path cannot be established. So we try to establish the trust relationships between CAs using *cross-certification*. The X.509 specification [4] defines *a cross-certification* in this way: "Two CAs exchange information used in establishing a *cross-certificate*. *A cross-certificate* is a certificate issued by one CA to another CA which contains a CA signature key used for issuing certificates." We show the distributed model using cross-certification in Figure 3.

1. compromise of CA's private key

When one CA's private key is exposed, certificate users who trust that CA are affected. However certificate users who trust the others are not affected, so the damage caused by the CA's private

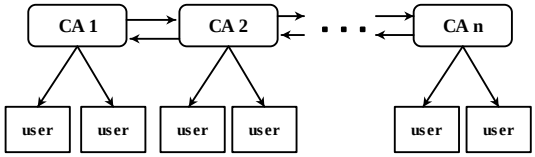

Figure 3: A distributed CA model using cross-certification

key exposures can be minimized.

2. certification path processing

In this model, certification path processing is not efficient. For example, when a verifier who trust *CA1* verify the certificate issued by *CAn*, the certification path length is n. That is, a verifier should check n certificates in the certification path.

As another method, every CA directly issues the cross-certificate to every other CA. This model is called *a fully connected mesh* (Figure 4). In this case, however, the number of cross-certification links required can be calculated as $n \cdot (n-1)$, where n is the number of CAs [1].

In general, discovering a optimal certification path is a very complex problem in case of using a lot of cross-certifications [2, 5].

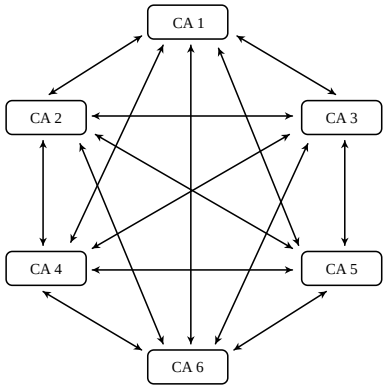

Figure 4: A Fully Connected Mesh (n=6)

2.2.2 A Subordinated Hierarchical Model

In this model, multiple CAs are formed the hierarchical structure with a Root CA at the top (Figure 5). This model has the following features [1].

In this model, only superior CA issues certificates to their subordinates and subordinate CAs do not certify their superiors. As the Root CA in this model is the single trust anchor, there is no other CA that can certify the Root CA. Root CA creates a self-signed certificate for itself and distributes it to all certificate users.

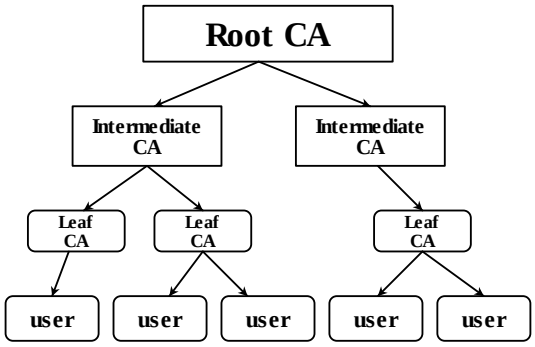

Figure 5: A Subordinated Hierarchical Model

1. compromise of CA's private key

When Root CA's private key is compromised, all certificates are affected. However, compromise of one intermediate CA's private key does not affect the certificates issued by the others in the same depth of hierarchy.

Compromise of Root CA's private key affected all certificates, as we mentioned above. This is the same problem as a single CA model. But in a single CA model, it increases the risk of exposing private key because all certificate requests go directly to one CA. While the danger of Root CA's private key disclosure can be mitigated in subordinated hierarchical model, since Root CA may issue certificates to its subordinates (Intermediated CAs). Therefore the Root CA's private key is used very infrequently

and the Root CA can spend most of its operation life offline, with its private key securely locked away [1].

2. certification path processing

The path construction in a subordinated hierarchical model is relatively simple. The certification path is leading the path from Root CA to the certificate that is verified.

But the path validation is less efficient than one of in a single CA model. Namely, the certificate path length is longer than one of a single CA model. In a single CA model, certificate path length is only one, while in subordinated hierarchical model, it becomes longer in proportion to depth of hierarchy. Therefore the number of certificate that verifier need to check is increasing.

3. Our Proposal Methods

Distributed CA models using general methods have the problem that the certification path length is long, so the load of verifier become heavy. In this section, we suggest the decentralization methods of CA. Using these methods, we realized the distributed CA models which the certification path length is short. We use two signature schemes: (1) *forward-secure signature scheme* (*FSS*) and (2) *key-insulated signature scheme*(*KIS*).

3.1 A decentralization method of using FSS

3.1.1 An overview of FSS

A lot of the digital signature schemes have been proposed, but they provide no security guarantees in case private keys (secret keys) are exposed. In many cases, it is easier to obtain a private key from a stolen device than to break the computational assumption on which the security of the system is based. If the private key is compromised, any message can be forged, so the leakage of the private key is quite realistic threat.

In order to minimize the damage caused by private key exposures, the notion of forward security for digital signatures was initially proposed by Anderson [7]. The basic idea is to extend a standard digital signature algorithm with a *key update* algorithm. The user's private key can be change frequently (for example, day by day) , while the public key stays the same (Figure 6). The notion of *forward-security* is that if the exposure of the private key at some time-period t, the attacker cannot forge signatures that belong to previous time periods. Bellare and Miner initially formalized *forward-secure digital signature scheme* [6]. Several forward-secure digital signature schemes have been proposed [8, 9, 10].

We notice these features and propose decentralization methods of CA. First, we introduce the scheme [1] as follows.

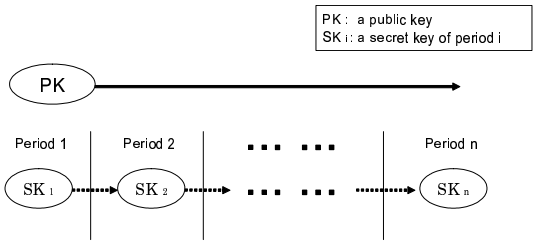

Figure 6: Forward-Secure Signature Scheme

(Definition)
p, q :prime number satisfying $p \equiv q \equiv 3 \pmod 4$.
T :the number of time-period
SK_j :the private key at time-period j $(0 \leq j \leq T)$.
PK :the fixed public key for verification signature
H :one-way hash function and output length is l-bit
l :security parameter
k :security parameter
M :message
Z_N :the set of integers $\{0, ..., N-1\}(= \mathbf{Z}/N\mathbf{Z})$
Z_N^* :the set of integers
$z \in \{0, ..., N-1\} : gcd(z, N) = 1(= (\mathbf{Z}/N\mathbf{Z})^*)$

(a) Key Generation Algorithm

$(1) p, q\ (\equiv 3 \pmod 4)$
$(2) N \leftarrow pq$
$(3) i = 1, ..., l;$
$\quad S_i \xleftarrow{R} Z_N^*$
$\quad U_i \leftarrow S_i^{2^{(T+1)}} \mod N$
$(4) SK_0 \leftarrow (N, T, 0, S_{1,0}, ..., S_{l,0})$
$\quad PK \leftarrow (N, T, U_1, ..., U_l)$

The signer generates key pair by running the above key generation algorithm, which takes as input the security parameter k determining the size of N, the number l of points in the keys, and the number T of time-period over which the scheme is to operate. The factor p, q of N are deleted once the key generation process is complete, so that they are not available to an attacker that might later break into the system on which the private key is stored.

(b) Key Update Algorithm

$SK_{j-1} = (N, T, j-1, S_{1,j-1}, ..., S_{l,j-1})$
For $i = 1, ..., l$ **do**
$\quad S_{i,j} \leftarrow S_{i,j-1}^2 \mod N$
$\quad SK_j \leftarrow (N, T, j, S_{1,j}, ..., S_{l,j})$
Return SK_j

Once key update algorithm are performed, the signer deletes the private key SK_{j-1}. This algorithm uses squaring modulo N and it is one-way function when factorization of N is unknown, it is computationally infeasible to derive SK_{j-1} from SK_j.

(c) Signing Generation Algorithm

$SK_j = (N, T, j, S_{1,j}, ..., S_{l,j})$
$(1) R \xleftarrow{R} Z_N^*$
$(2) Y \leftarrow R^{2^{(T+1-j)}} \mod N$
$(3) c_1 ... c_l \leftarrow H(j, Y, M)$
$\quad Z \leftarrow R \cdot \Pi_{i=1}^{l} S_{i,j}^{c_i} \mod N$
$(4) \langle j, (Y, Z) \rangle$

Signer generates the signature on message M by using signing generation algorithm.

(d) Signing Verification Algorithm

$(1) PK = (N, T, U_1, ..., U_l)$
$(2) c_1 ... c_l \leftarrow H(j, Y, M)$
$(3) Z^{2^{(T+1-j)}} \stackrel{?}{=} Y \cdot \Pi_{i=1}^{l} U_i^{c_i} \mod N$

Verifiers check the signature which is sent to signer, by using the fixed public key. Any time-period j, the signature can be verified by using PK. Verifiers check the signatures by signing verification algorithm at time-period j. If (3) is hold, verifier can certain the validity of the signature.

3.1.2 Decentralization processes

The processes of our method are as follows:

(1) Generate the key pair of Root CA

Let (PK, SK_0) be the key pair of the Root CA and they are generated by key generation algorithm (section 3.1.1) . The Root CA creates a self-signed certificate itself and distributes this certificate to all certificate users.

In case the exposure of SK_0, all certificates are affected as well as a single CA model and a subordinated hierarchical model. So, this private key must be stored very carefully. Some methods that the risk of exposure private key is mitigated are proposed [13].

(2) Update the private key

By key update algorithm (section 3.1.1), SK_0 is updated as follow and in result n private keys are generated.

$$SK_0 \rightarrow SK_1 \rightarrow SK_2 \rightarrow, ..., \rightarrow SK_n$$

In scheme [1], once update process is finished, the signer deletes previous private key. In our method, however, multiple private keys that are generated by update algorithm are stored securely in the

separate place.

(3) Decentralization of CA

In our method, multiple private keys that are generated by update algorithm are assigned to $CA_1, ..., CA_n$. That is, CA_i has the private key SK_i and stores it securely.

(4) Issue the certificate to users

Certificate users determine which CAs delegates authoring of certificates issuing to them. This decision is depended on situations. CA_i issues certificates signed own private key SK_i.

(5) Certificate Revocation

Occasionally, certificates are revoked for various reasons. For example, when user's private key is exposed, or user's personal information is changed and so on. In case that the certificates need to be revoked, certificate users must notify this point time, at which the certificate is invalid, to the CA which issued this certificate. The CA receives this notification, and publishes certificate users that the certificate has been revoked. As the typical publishing method is the distribution of *Certificate Revocation List* (*CRL*).

In our method, using the same publishing methods, the CA publishes a CRL. This CRL contains the information on revoked certificate and signed by issuer.

1. certification path processing

Figure 7 shows our model in case of $n = 3$. Certificate verifiers must verify the validity of issuer's signature contained the certificate. In our model, the verifiers who trust a Root CA can check all certificates by using the Root CA's public key. In other word, the certification path length is only one. In order to verify information on the revoked certificate which is issued at time-period j, verifiers need to get CRL which is issued by the CA_j. Now let CRL_i be the CRL issued by CA_i. CRL_i contains signature of CA_i. In our model, any CRL_i can be

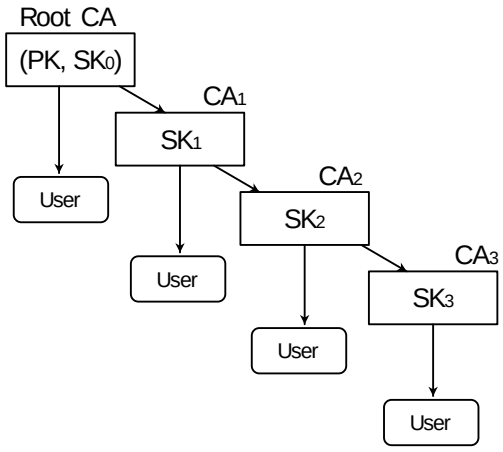

Figure 7: A Distributed CA Model Using FSS

verified by the Root CA's public key.

In this way, the length of certificate path is shorter than one of a subordinated hierarchical model, so the burden of verifier is able to mitigate.

2. compromise of CA's private key

Now, we consider in case of compromise private key SK_i, except for SK_0. In our model, it is computationally difficult for attackers to know $SK_j (j < i)$. Consequently, when SK_i is compromised, it is not affect to certificates that are issued by $CA_k (k < i)$.

Now, we consider the subordinated hierarchical model that certificate users are divided equally (just like the binary tree structure). $\#(U)$ denotes the number of certificates issued by the leaf CAs in subordinated hierarchical model. At this time, the number of certificates affected is at most $\#(U)/2$ when compromise of private key $SK_i (i \neq 0)$.

In our model, $\#(U)$ denotes the number of certificates issued by CA_i. And $\#(Cert_0)$ denotes the number of certificates issued by Root CA. The small number of $\#(Cert_0)$ is better, taking into account the risk of exposing the Root CA's private key. Consequently, the maximum number of affected certificate is near $\#(U)$ when the private key of intermediate $CA_j (j \neq 0)$ is compromised.

These two model's structures are different. The

structure of subordinated hierarchical model is like the binary tree, while structure of our model is like the linear list. Therefore in the former, the number of certificate affected is at most a half. While in our model, only one new private key is generated from one private key using key update algorithm in scheme [6]. Consequently in our model, we can not generate private keys which are mutually independent like a hierarchical model.

3.2 Improved Method Using FSS

In this section, we proposed the trust model like hierarchical structure using FSS. When one CA's private key is exposed, the attacker can not forge certificates which are issued by another CAs. In order to realize this model, we generate multiple private keys and assign them to the multiple CAs. A CA has the multiple private keys and issues the certificate which contains the multiple signatures signed by these multiple private keys.

3.2.1 Decentralization processes

Let n be the number of private key that Root CA has.

1. Generation of key-pairs

   Using key generation algorithm of scheme [1], we create n key-pairs (public keys and private keys) and we denote them $(PK, SK_0), ..., (PK_n, SK_0^n)$. These key-pairs are Root CA's. Then Root CA issues self-signed certificates, which are signed by each own private keys, to all certificate users.

   As well as before methods, when all multiple private keys of Root CA are compromised, all certificates are affected. So these private keys must be stored securely.

2. Update the private keys

   We assume to $T < n$. Using key update algorithm, multiple private keys of Root CA are updated in time and new private keys are generated as follows.

$$
\begin{aligned}
SK_{0,1} \rightarrow SK_{1,1} \rightarrow \cdots \rightarrow SK_{T,1} \\
SK_{0,2} \rightarrow SK_{1,2} \rightarrow \cdots \rightarrow SK_{T,2} \\
\vdots \\
SK_{0,n} \rightarrow SK_{1,n} \rightarrow \cdots \rightarrow SK_{T,n}
\end{aligned}
$$

   n new private keys are created each time-period and total number of private key is $T \times n$.

3. Decentralization of CA

   We assume that $\{CA_1, ..., CA_n\}$ are the separate intermediate CAs. First, we divide all private keys into the number of CA_i and distributed to each CA_i. We choose one private key from each time-period j and the same operation is performed from time-period 1 to T.

   In result, each CA_i has T private keys.

4. Issuance of certificates

   In this method, a certificate contains multiple signatures by multiple private keys. An attacker needs to know all of T private keys in order to succeed in forgery of signature.

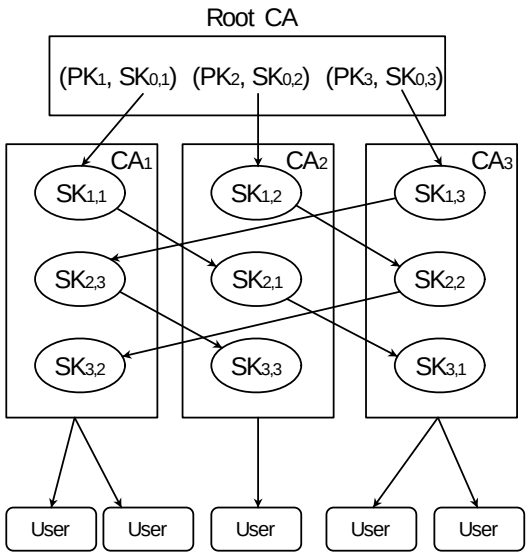

Figure 8: Our Model Like A Hierarchy Structure

1. compromise of CA's private keys

Figure 8 shows our model in case of $n = 3, T = 3$. We consider when the private key of intermediate CA is compromised. Suppose that an attacker succeeds in getting the multiple private keys of CA_1. $(SK_{1,1}, SK_{2,3}, SK_{3,2})$ Using key update algorithm, other private keys can be derived as the following.

$$SK_{1,1} \rightarrow SK_{2,1} \rightarrow SK_{3,1}$$
$$SK_{2,3} \rightarrow SK_{3,3}$$

However, because of the one-way function, an attacker can not derive $SK_{1,2}, SK_{1,3}, SK_{2,2}$, so an attacker cannot forge the certificates by issued CA_2 and CA_3.

From above evaluation, when one CA's multiple private keys are exposed, the other CA's multiple private keys can not be perfectly derived. Therefore, an attacker can not forge certificate which are issued by the others.

This model is like the hierarchical structure and can minimize the effect caused by exposing the CA's private keys.

2. certification path processing

In this model, the certificate contains multiple signatures as many as multiple private keys. When the verifier checks it, they use the corresponding multiple public keys, which are Root CA's public keys. Only the case all of signature is valid, the verifier may certain the validity of this certificate. Moreover, verifier can verify any CRL_i using multiple public keys of Root CA.

3.3 A decentralization method of using KIS

3.3.1 An overview of KIS

A key-insulated cryptography is the different means in order to minimize the damage caused by the private key exposures [11]. The notion of key-insulated security is that if the exposure of the private key at *time-period j*, the attackers can not derive the private keys any other *time-period*.

A *key-insulated signature scheme (KIS)* is proposed by Y.Dodis et al [12]. Signature computation is frequently performed on insecure device, so the private key is likely to be exposed. *KIS* is one way to mitigate the damage done when this occur. The private key stored on an insecure device is refreshed at discrete time periods via interaction with a physically secure device which stores a *"master key"*. As well as a forward-secure signature scheme, public key remains fixed and the signatures in any time periods can be verified by fixed public key. The attackers who obtain the private key, they can not forge signature for any of remaining time-period. We introduce the scheme [12] as follow.

(Difinition)
PK :a public key
SK^* :a master key stored physically secure device.
SK_i :the private key of time-period i.
$SK'_{i,j}$:a partial key.

1. *Gen*: key generation algorithm
   $(PK, SK^*, SK_0) \leftarrow Gen(1^k, N)$
   This algorithm take as input a security parameter 1^k and the total number of time period N. It returns a public key PK, a master key SK^*, and an initial key SK_0.

2. Upd^* :device key-update algorithm
   $SK'_{i,j} \leftarrow Upd^*(i, j, SK^*)$
   This algorithm take as input i, j for time period and the master key SK^*. It returns a partial private key $SK'_{i,j}$.

3. Upd :user key-update algorithm
   $SK_j \leftarrow Upd(i, j, SK_i, SK'_{i,j})$
   This algorithm is taking as input i, j, a private key SK_i, and a partial key $SK'_{i,j}$. It returns the private key SK_j for time period j.

4. Sig :signing algorithm
   $\langle i, s \rangle \leftarrow Sig_{SK_i}(i, M)$
   This algorithm is taking as input i, a message M, and a private key SK_i. It returns a signature $\langle i, s \rangle$ consisting of the time period i and a signature s.

5. $Vrfy$:verification algorithm
   $Vrfy_{PK}(M, \langle i, s \rangle)$
   This algorithm is taking as input the public key

PK, a message M, and a pair $\langle i, s\rangle$. It returns a bit b, where $b = 1$ means the signature is accepted. If $Vrfy_{PK}(M, \langle i, s\rangle) = 1$, we say that $\langle i, s\rangle$ is a valid signature of M for period i.

3.3.2 Decentralization processes

We try to construct a distributed CA model using KIS. Decentralization processes are the same as using FSS(Section 3.1.2). However a master key SK^* must be stored on a physically secure device. First, multiple private key are generated using KIS and assigned multiple separate CAs. All processes are finished, then a master key may be deleted.

1. compromise of CA's private key

If a CA's private key is exposed, the attackers can not derive another CA's private keys. So certificate issued by another CA are not affected and this model realizes to minimize the affluence when CA's private key is compromised.

2. certification path processing

As we explain section 3.3, all signatures in any time-period can be verified by a fixed public key. In this model, all certificates can be verified by Root CA's public key. That is, certification path length is only one, so the load of certificate verifiers can be mitigated.

In comparison with the method using FSS, the method of using KIS can minimize the damage caused by compromise of CA's private key. But there is no external device in FSS, while the physically secure device is needed in KIS.

4. Conclusion

First, we discuss the advantages and disadvantages of the distributed CA models. We propose the decentralization methods absorbing these advantages by using FSS and KIS. FSS and KIS have the features that any signature can be check by fixed public key. We utilize this feature and construct the trust models that CA is decentralized and certification path is very short. In our model, verifiers may not verify the multiple certificates, so the burden of verification processing is mitigated.

References

[1] A. Nash, W. Duane, C. Joseph and D. Brink, PKI - Implementing and Managing E-Security, Osborne Media Group, 2001.

[2] C. Adams, S. Lloyd, Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations, MACMILLAN TECHNICAL PUBLISHING , 2000.

[3] R. Perlman, An Overview of PKI Trust Models, IEEE Network, 13(6), 38-43, 1999.

[4] Internet X.509 Public Key Infrastructure Certificate and CRL profile, RFC3280, 2002.4.

[5] Steve Lloyd, Understanding Certification Path Construction, PKI Forum, Sep 2002.

[6] M. Bellar and Sara K. Miner, A Forward-Secure Digital Signature Scheme, Crypto 99 Proceedings, Lecture Notes in Computer Science. Springer-Verlag, Vol. 1666, 1999.

[7] R. Anderson, Two remarks on public-key cryptology, Sep. 2000, Relevant material presented by the author in an invited lecture at the Fourth ACM Conference on Computer and Communications Security (Apr. 1997).

[8] H. Krawczyk, Simple Forward-Secure Signatures From Any Signature Scheme, In 7th ACN Conference on Computer and Communications Security, 2000.

[9] M. Abdalla, L. Reyzin, A New Forward-Secure Signature Forward-Secure Digital Signature Scheme, Advanced in Cryptology - Asiacrypt 2000, Lecture Notes in Computer Science, vol. 1976, pp. 116-129, Springer-Verlag, 2000.

[10] T. Malkin, D. Micciancio, and S. Miner, Efficient Generic Forward-Secure Signatures With An Unbounded Number Of Time Periods, In Advances in Cryptology - EUROCRYPTO'02, Lecture Notes in Computer Science. Springer-Verlag. Vol. 2332, 2002.

[11] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu and Moti Yung, Key-Insulated Public Key Cryptosystems, EUROCRYPT 2002, Lecture Notes in Computer Science, Vol. 2332, pp. 65-82, Springer-Verlag, 2002.

[12] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu and Moti Yung, Strong Key-Insulated Signature Schemes, International Workshop on Practice and Theory in Public Key Cryptography (PKC 2003), Lecture Notes in Computer Science, Vol. 2567, pp.130-144, Springer-Verlag, 2003.

[13] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, Proactive Public Key and Signature Systems, Dec 1996.

MOCA : Mobile Certificate Authority for Wireless Ad Hoc Networks

Seung Yi, Robin Kravets
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{seungyi,rhk}@cs.uiuc.edu

Keywords : PKI, Key Management, Security, MANET, Ad Hoc Networks, Threshold Cryptography

Abstract

PKI has been recognized as one of the most effective tools for providing security for dynamic networks. However, providing such an infrastructure in ad hoc wireless networks is a challenging task due to their infrastructure-less nature. In this paper, we present these challenges in detail, identify the requirements for such solutions, and propose a practical PKI service for ad hoc networks. We employ threshold cryptography to distribute the CA functionality over specially selected nodes based on the security and the physical characteristics of nodes. The selected nodes that collectively provide PKI functionality are called MOCA (MObile Certificate Authority)s. Using these MOCAs, we present an efficient and effective communication protocol for correspondence with MOCAs for certification services. Results from our simulations verify the effectiveness and the efficiency of our approach.

1 Introduction

Since its birth more than two decades ago [20], public key cryptography has been recognized as one of the most effective mechanisms for providing fundamental security services including authentication, digital signatures and encryption. The effective management of digital certificates is a key factor for the successful wide-spread deployment of public key cryptography. PKI (Public Key Infrastructure), an infrastructure for managing digital certificates, was introduced exactly for this purpose [19]. The most important component of PKI is the CA (Certificate Authority), the trusted entity in the system that vouches for the validity of digital certificates. The success of PKI depends on the security and availability of the CA to the principals in a system (or the nodes in a network) since a principal must be able to correspond with the CA to get a certificate, check the status of another principal's certificate, acquire another principal's certificate, and so on. PKI has been deployed for wired networks [3, 1] and some infrastructure-based wireless networks [9]. Since good connectivity can be assumed in these networks, the main thrust of research in such environments has focused on the security of the CA and the scalability of the CA to handle a large number of requests.

However, it is unclear if such approaches can be extended to ad hoc networks. A wireless ad hoc network or a mobile ad hoc network (MANET) [13] is a network where a set of mobile devices communicate among themselves using wireless transmission without the support of fixed or stationary infrastructure. Due to its infrastructure-less nature, an ad hoc network can be deployed very fast at a relatively low cost enabling communication when it is not possible or too expensive to deploy a support infrastructure. A wide range of military and commercial applications have been proposed for ad hoc networks. For example, a unit of soldiers moving in the battlefield cannot afford to set up a base station every time they proceed to a new area. Similarly, setting up a communication infrastructure for a casual and spontaneous conference meeting among a small number of people cannot be justified financially. Additionally, ad hoc networks can be the perfect tool for a disaster recovery or emergency situation when the existing communication infrastructure is either destroyed or disabled. A large portion of research in ad hoc networks has focused on routing, medium access control and power management and only recently researchers have started looking at security issues in ad hoc networks.

Connectivity, which was assumed to be good in previous PKI solutions, is not easy to maintain in ad hoc networks. On the contrary, maintaining connectivity is one of the main challenges, since the inherent infrastructure-less nature of ad hoc networks inhibits guaranteeing any kind of connectivity. Another serious problem present in ad hoc networks is the increased physical vulnerability of the nodes themselves. Considering that many ad hoc networks will be deployed with mobile nodes [12], the possibility of the nodes being captured or compromised in a hostile environment is higher than in wired networks with stationary hosts. Mobile nodes in infrastructure-based wireless networks have the same vulnerability, but they can rely on the infrastructure for detection of compromised nodes, help with recovery and storage of sensitive information. Since there is no stable entity in an ad hoc network, ad hoc nodes cannot enjoy such conveniences.

Several proposed solutions for providing PKI for ad hoc networks address the increased vulnerability of the mobile nodes by employing techniques to distribute the CA functionality across multiple nodes, generally using threshold cryptography [21, 15]. These approaches also increase the availability of the CA. While these approaches share some similarities with the MOCA framework, they are either conceptual [21], not targeted for ad hoc networks [22], or vulnerable against attacks [15]. The MOCA framework provides a practical and secure key management framework for ad hoc networks with communication support that considers the dynamic nature of connectivity in ad hoc communication.

We identify two main challenges in distributing the CA functionality over multiple nodes. The first challenge is picking a set of nodes to collectively provide the CA service. The second and equally important challenge is how to provide efficient and effective communication between the mobile nodes and the CA nodes, even in dynamic networks with possible compromises or temporary network partitions.

To this end, we present the MOCA (MObile Certificate Authority) framework. A MOCA is a mobile node within an ad hoc network selected to provide distributed CA functionality. A network operator chooses MOCAs based on an observation of heterogeneity among mobile nodes, typically physically more secure, computationally more powerful, or more trustworthy nodes. MOCA nodes use threshold cryptography to share the responsibility and provide CA services with strong security and high availability. Client nodes are equipped with MP (MOCA certification Protocol) that enables contacting sufficient MOCAs in an efficient and effective way. We demonstrate the effectiveness of our protocol with extensive simulations. Based on simulation results, we also provide certain insights into how to configure such security services for ad hoc networks.

The remainder of this paper is organized as follows. In Section 2, we define important metrics for designing key management frameworks for ad hoc networks and use these metrics to evaluate some existing research. In Section 3 and 4, our approach using MOCAs is presented and Section 5 presents simulation results from our implementation. Section 6 suggests some possible extensions of this work and we conclude in Section 7.

2 Key Management for Ad Hoc Networks

Any successful key management framework for ad hoc networks requires fault tolerance, security, and availability. These terms are sometimes used interchangeably, mainly because they are not independent of each other. To avoid confusion, we clearly define these terms and apply them to some existing approaches for evaluation.

Fault Tolerance: The main concern of fault tolerance is the capability to maintain correct operation in the presence of faulty nodes. We restrict the definition of faulty nodes to observable faults. If a node is malfunctioning and other nodes can observe such malfunctions, a certain level of recovery is possible. We employ intelligent replication using threshold cryptography to provide tolerance of faulty nodes.

Security: Acting as the trust anchor for the whole network, the MOCA framework should be secure against malicious nodes or adversaries. While it may not be possible to be resistant to all levels of attacks, there should be a clear threshold of attacks a system can withstand while operating normally. MOCA nodes are selected based on their node characteristics and they form a distributed CA to resist adversaries.

Availability: Traditionally, the term availability has been used in conjunction with fault tolerance. But in ad hoc networks availability is also highly dependent on the connectivity of the network. In wired networks, if there are no faulty or compromised nodes, the system is by definition available for clients since connectivity is not a problem. In ad hoc networks, even when there are no faulty or compromised nodes, clients may not be

able to contact the desired services due to inconsistent connectivity. We provide a set of efficient and effective communication protocols for clients to contact MOCAs to address availability.

The simplest approach to providing CA functionality in an ad hoc network is to assign a single node to be the CA. The success of this scheme depends on that single CA node. This approach is not fault tolerant, since failure of one node breaks the system. Similarly this approach is highly vulnerable, since an adversary need only compromise one node to acquire the secret key. Finally, given the expected mobility and unpredictability of ad hoc networks, it may be possible that nodes will not be able to reach the CA in a timely fashion, making availability very unpredictable. Therefore, a single CA cannot effectively service a whole ad hoc network.

Robustness, which is missing in the single CA scheme, can be achieved by replicating a fully functional CA on *r* different nodes. With *r* replicas, the system can withstand *(r - 1)* failures because the CA service is available as long as there is at least one operational CA. Availability has also been improved since a client node has a better chance of reaching one of *r* CAs to get service. Unfortunately, the system has become more vulnerable. An adversary need only compromise one of the *r* CA nodes to acquire the secret key and so compromise the whole system. Therefore, using replicated CAs is not a viable solution in ad hoc networks. The problem of using replicated CAs stems from the fact that each replica has full knowledge of the system secret.

Zhou and Haas first proposed to use threshold cryptography to securely distribute the CA's private key over multiple nodes to form a collective CA service [21]. Using *k* out of *n* secret sharing [4] can provide a good level of fault tolerance and security. They address the problem clearly and present conceptual design issues, but do not address the problem of availability in their work. The authors continued their work in COCA, which is a distributed CA approach using threshold cryptography [22], designed to serve networks like the Internet. Again, connectivity between clients and the distributed CAs was not a concern.

Kong and others address availability by making all *M* nodes in the network share CA functionality [15]. A client need only contact *k* out of *M* nodes to get a certification service. Assuming there are *k* nodes in a client's one hop neighborhood, the client can get a certification service cheaply by using a one-hop broadcast for the request. While this solution addresses availability and fault tolerance, it compromises the security of the system. In general, the gap between *k* and *n* in secret sharing schemes defines the security of the system. *k* can be chosen between 1 and *n* in any secret sharing scheme. As *k* approaches *n*, thus closing the gap between *k* and *n*, the system becomes more secure because an adversary needs to compromise at least *k* nodes to collapse the system. But if *k* is too large, the system becomes less available to clients and also less tolerant to faults. When *k* approaches 1, making the gap larger, the effect is reversed and the system becomes more available but also less secure. Kong chose to keep *k* relatively small to address the availability problem and ended up with a vulnerable system where any adversary need only compromise a small number of nodes in the network to collapse the service.

Another notable scheme is proposed by Hubaux and others [10]. In their scheme, there is no concept of a CA. Every node acts as its own CA, similar to the PGP "Web of Trust" model. The main difference between PGP and their scheme is that there is no longer a well-known certificate directory where all certificates are stored. Rather, every user in the system carries a part of the certificate directory. In PGP, when two users wish to authenticate each other, they must search the certificate directory for a chain of certificates that links both users. In Hubuax's scheme, this problem is transformed into finding an intersecting point between the certificate chains carried by each user. Hubaux proposed a shortcut-hunter algorithm for this problem. While their approach is practical for the totally self organizing networks they aimed at, it has the inherent problem of no definite trust anchor like the CA in other CA-based PKI approaches. Therefore, it is not meaningful to evaluate this scheme using our framework.

3 MOCA

In this section, we present a practical key management framework for ad hoc networks. We first discuss the impact of heterogeneity among mobile nodes in a given ad hoc network and how it can be exploited to help choosing the MOCA nodes better. Then, we describe the details of MOCA framework and present a set of parameters to tune our framework.

3.1 Heterogeneity within an Ad Hoc Network

Most research in ad hoc networking has implicitly treated all nodes as identical in many respects, including power, transmission range, computational capacity, and security. We contend that mobile nodes in many ad hoc networks will be heterogeneous in many respects, especially in terms of their security and that any security service or framework should utilize this *environmental* information. For example, consider a battlefield scenario with a military unit consisting of infantry soldiers, platoon commanders' jeeps, company commanders' command vehicles, artillery vehicles, transport vehicles, and even tanks. All of these nodes may have different ranks, power, capabilities, transmission ranges, levels of physical security, and so on. In such a case, it would be wise to pick nodes with higher ranks, more power, more capabilities to provide any security service to the rest of the network. While it may not be necessary to exploit this potential heterogeneity to enhance the basic ad hoc routing itself, certainly this heterogeneity can be used to make the network more secure by endowing "better" nodes with more sensitive information.

Similar situations can be imagined in emergency rescue operations, mining operations, or any other scenarios where ad hoc networks can play a critical role to facilitate operations. Even in a simple school field trip, it makes more sense to allow teachers to perform sensitive operations instead of students. In general, knowledge of such heterogeneity should be used to determine the nodes that will share the responsibility of the CA. For example, the chosen nodes could be the most physically secure nodes with maximum resources that are least prone to compromises or failure.

It may seem counterintuitive to limit the candidate nodes for MOCAs to a subset of mobile nodes. This decision puts a limit on the maximum number of MOCAs in the system, which in turn may reduce the level of security and fault tolerance achieved by the distributed nature of MOCAs. For example, in a 300 node network, an operator may have the choice of selecting 200 random nodes to support CA functionality. Or the operator may pick 30 nodes with higher physical security. Blindly comparing the number of MOCAs in the system, the first approach looks better because it has more MOCAs in the network, improving fault tolerance and availability. But by guaranteeing an adequate level of security of the 30 MOCAs in the second case, compromising them can be made much harder than compromising the randomly selected MOCAs in the first case, hence making the second case more secure against adversaries.

It is possible that an ad hoc network does not have enough heterogeneity among the nodes, which may make it difficult if not impossible to choose MOCAs based on this heterogeneity assumption. In such cases, we can fall back to random sampling to choose MOCAs. Our protocol still works as designed but the level of security will decrease since there is no guarantee on the security of each MOCA.

3.2 MOCA Framework

In our framework, n MOCA nodes provide the functionality of a CA to the whole network. Using threshold cryptography, these n MOCAs share the CA's private key and any set of k MOCAs can reconstruct the full CA key.

Threshold cryptography is an application of secret sharing that was first proposed by Shamir [4]. The basic idea of secret sharing is that it is mathematically possible to divide up a secret to n pieces in such way that anybody who requires the full secret can collect any k pieces out of those n to reconstruct the full secret. k becomes the threshold needed to reconstruct the secret. Threshold cryptography applies this technique to the keys for the cryptographic operations. Frankel and Desmedt [8] proposed to use secret sharing for the private key of public key cryptography and Shoup proposed a way to generate a digital signature from key pieces without reconstructing the full key at any point [18].

With a naive implementation, the CA's private key gets reconstructed per request at the client. To prevent this, we use threshold digital signatures [18]. Any client requiring a certification service must contact at least k MOCAs with its request. The contacted MOCAs each generate a partial signature over the received data and send it instead of sending their key share. The client needs to collect at least k such partial signatures to reconstruct the full signature and successfully receive the certification service.

Maintaining information on revoked certificates is one of the key tasks of the CA and this topic has received much attention in recent years [5]. In the MOCA framework, we use the simple certificate revocation list (CRL) approach and we plan to investigate a more adequate means of certificate revocation in ad hoc networks in the future. In the current framework, again k or more MOCAs must agree to revoke a certificate. Each MOCA generates a *revocation certificate* that contains which certificate to revoke and signs it with its key share. Then, each MOCA broadcasts the partially signed revocation certificate. Any node that collects k or more such partially signed revocation certificates can reconstruct the full revocation certificate. The list of revoked certificates or the CRL can be maintained by any node in the network since revocation certificates are not secrets but public information. The CRL can be stored at each

node, the MOCAs, or at a set of specially designated nodes. To avoid false revocation, unless the MOCA framework is compromised, it is not possible to forge a revocation certificate with a valid signature on it. In the MOCA framework, the partial revocation certificates are distributed to all nodes in the ad hoc network via a network-wide flood. While this imposes significant overhead on the network, we would expect a revocation to be a rather infrequent event and the cost would be amortized over time. In our current work, we are considering techniques to provide more efficient support for revocation.

3.3 Tuning Threshold Cryptography

The shape of a MOCA framework is determined by the total number of nodes in the network, the number of MOCAs, and the threshold value for secret reconstruction. Although the total number of nodes in the network, *M*, can change dynamically over time, it is not a tunable parameter. The number of MOCAs, *n*, is determined by the characteristics of nodes in the network, such as physical security or processing capability and it is also not tunable. In this system, *n* defines the limits of the system as an upper bound for *k*, the minimum number of MOCAs a client must contact to get certification services.

Given *M* and *n*, the last parameter *k*, the threshold for secret recovery, is indeed a tunable parameter. Once *k* has been chosen and the system is deployed, it is expensive to change *k*. Therefore it is important to understand the effects of varying *k* on a given system.

k can be chosen between 1 (a single CA for the whole network) and *n* (a client needs to contact all MOCAs in the system to get certification services). Setting *k* to a higher value has the effect of making the system more secure against possible adversaries since *k* is the number of MOCAs an adversary needs to compromise to collapse the system. But at the same time, a higher *k* value can cause more communication overhead for clients since any client needs to contact at least *k* MOCAs to get certification services. Therefore, the threshold *k* should be chosen to balance the two conflicting requirements. It is clear that no one value will fit all systems. Our goal is to provide some guidelines for choosing an appropriate *k*. To make our protocol more adaptive to varying network configurations, we introduce additional tunable parameters in Section 5.

4 MOCA Certification Protocol

In this section, we describe a key aspect for successful PKI in ad hoc networks: *communication*. The choice of which and how many MOCAs to contact must be made in coordination with the communication protocol used to access the MOCAs. Even after MOCAs have been selected and deployed in the system, it is useless if clients cannot contact them and receive services. The communication pattern between a client and *k* or more MOCA servers is *one-to-many-to-one*[1], which means that a client needs to contact at least *k* MOCAs and receive at least *k* replies. To provide an effective and efficient way of achieving this goal, we propose MP (MOCA certification Protocol).

In MP, a client that requires certification services sends Certification Request (CREQ) packets. Any MOCA that receives a CREQ responds with a Certification Reply (CREP) packet containing its partial signature. The client waits a fixed period of time for *k* such CREPs. When the client collects *k* valid CREPs, the client can reconstruct the full signature and the certification request succeeds. If too few CREPs are received, the client's CREQ timer expires and the certification request fails. On failure, the client can retry or proceed without the certification service.

The CREQ and CREP messages are similar to Route Request (RREQ) and Route Reply (RREP) messages in on-demand ad hoc routing protocols like AODV [7] and DSR [11]. The management of routing information is also similar to these protocols. As a CREQ packet passes through a node, a reverse path to the sender is established. These reverse paths are coupled with timers and maintained long enough for a returning CREP packet to be able to travel back to the sender. If no CREP is returned within the time-out period, the reverse path entry in the routing table expires and is purged. If a CREP traverses back through the previously set-up reverse path to the sender, the routing table entries are refreshed and the bidirectional path remains in the routing table for potential reuse. This similarity to on-demand routing presents a potential for our certification protocol and the existing on-demand routing protocols to benefit from each other by sharing routing information.

[1] We term this pattern of communication "Manycast".

4.1 Flooding

The simplest means of reliable data dissemination, flooding, can be used to reach all MOCAs in the network [17]. As shown in previous results, while this flooding approach is effective, it generates a large amount of traffic. First, the overhead generated from a network-wide CREQ flood is large. Second, since a client has no way to limit the dissemination of a CREQ, all the MOCAs that receive a copy of the CREQ respond with a CREP and the client receives more responses than it actually needs to reconstruct the full signature. Any partial signatures beyond the required *k* are discarded and waste networking and processing resources.

4.2 Unicast-based Optimization

To reduce the amount of overhead from flooding while maintaining an acceptable level of service, we introduce β-unicast, where the client can use multiple unicast connections to replace flooding if the client has *sufficient* routes to MOCAs in its routing cache. β in the name represents the *sufficient* number of cached routes to MOCAs to use unicast instead of flooding. If this sufficiency is achieved, β-unicast sends multiple unicast CREQs instead of flooding the network with CREQs. β-unicast does not initiate any form of route discovery as in on-demand ad hoc routing protocols where a network is usually flooded with route discovery packets. Instead, β-unicast only utilizes the existing information in the route cache. Blind use of unicast with insufficient cached routes can result in service failure, which in turn causes another round of flooding. To prevent such a situation, our protocol uses flooding when there are not enough routes cached.

The definition of sufficiency is tightly coupled to the value of *k*, but is also highly dependent on the state of the network. If the network is very stable with low mobility, having just *k* cached routes may be sufficient since the client can expect to receive all *k* replies back. If the the network is highly mobile and routes are unstable, sending out exactly *k* unicast CREQs is dangerous since even one loss of a CREQ or a CREP results in the failure of the whole certification request. In this situation, the node should send out additional CREQs to increase the probability of success. The number of additional CREQs is defined by α, a marginal safety value used to increase the success ratio of β-unicast. α is node specific and can be determined based on the node's perception of the network status. How a node will perceive the status of the network is out of the scope of this paper and is an active topic of research. The sum of the crypto threshold *k* and the safety margin α is the unicast threshold, β, hence the name β-unicast.

Our previous work showed that a client often has a moderate number of cached routes to MOCAs under reasonable certification traffic in the network [17]. A result given in [17] shows that under a mobility of 10 second pause time and 10 m/s maximum speed, clients have cached routes to 45% of the MOCAs on average. One interesting question is how to choose among the MOCAs cached in the routing table. If there are only β cached routes, the client needs to contact every one. But if there are more than β routes in the cache, the choice of which ones to use can affect performance. We define three different schemes:

1. Random MOCAs - Choose β random MOCAs with cached routes.
2. Closest MOCAs - Choose β MOCAs with smallest hop counts in the cache. Intuitively, this approach has the benefit of the shortest response time and the smallest packet overhead since the CREQ packets travel the least distance.
3. Freshest MOCAs - Choose β MOCAs with the freshest cache entries. The most recently added or updated entries should not be stale, especially under high mobility. By choosing the freshest MOCAs, the client should be able to minimize the risk of failure under high mobility.

We provide simulation results for these certification protocols in the next section.

5 Evaluation

The focus of our evaluation of the MOCA framework is effectiveness and efficiency (or cost). Effectiveness is measured using the success ratio of certification requests. For flooding based protocols, success ratio is defined as the total number of received CREPs. For unicast-based optimizations, every CREQ that receives *k* or more CREPs is counted

as a successful certification request and success ratio is defined as:

$$\frac{(Number\ of\ successful\ certification\ request)}{(Number\ of\ total\ certification\ request)}$$

The cost of a certification protocol can be evaluated using the two metrics: packet overhead and additional communication delay caused by the certification process. The simulations demonstrate that our approach is practical for ad hoc networks providing adequate service availability without incurring prohibitive overhead.

For all simulations, there are three parameters that can be tuned according to the network configuration.

- Time-out Threshold τ - τ is used by a client to decide how long to wait for certification replies after sending out a certification request. Larger τ values can increase the possibility of success since the node waits longer for the CREPs to come back. But if there are not enough CREPs on their way back, the certification request will eventually fail and larger τ values can cause the node to wait needlessly. If τ is set too small, even when there are enough CREPs on their way back to the client, the client gives up too soon, discarding the CREPs on the way.

- Crypto Threshold *k* - *k* is the minimum number of CREPs required for a client to reconstruct the MOCA's full signature and render the certification request successful. If *k* is set low, a client only needs to collect a small number of *k* partial signatures to continue. Thus the success ratio increases and the packet overhead decreases, but at the same time an adversary only needs to compromise a small number of *k* MOCAs to compromise the framework. High *k* values can make attacks difficult, but the burden on clients and the cost in terms of packet overhead also increases since a client needs to contact a large number of MOCAs for any certification request.

- Unicast Threshold β - The unicast threshold β is the sum of the crypto threshold *k* and the margin value α. Larger α values make the framework more robust but limit optimizations because clients must have β instead of *k* cached routes to use the unicast-based approaches. Also a larger α value generates more overhead. When β-unicast is used, a larger α results in more unicast requests and replies. Also a larger α increases β, which reduces the probability of β-unicast being used and results in more flooding. Setting α to a low value makes it easier for a client to use unicast-based approaches, but may cause an excessive amount of certification failure due to the loss of too many CREQs or CREPs in the process.

By evaluating the results of the simulations, we provide some insight into how to configure the MOCA networks to achieve efficiency and availability at the same time.

5.1 Simulation Set-Up

We implemented our certification protocols in the ns-2 network simulator [2]. We test our protocol under two hypothetical scenarios. Consider a 1km by 1km battlefield with 150 or 300 friendly units including foot soldiers, jeeps, humvees, tanks and command vehicles. 30 MOCAs are deployed in both cases. 30 MOCAs represent 20% and 10% of the total nodes, which we believe to provide a reasonable number of MOCAs to support the ad hoc network. Each simulation is run for 10 minutes. One thing to note is that this scenario can be applied to other situations like a school field trip or a rescue operation. Although we use military examples to maintain consistency throughout the paper, none of our simulation factors depends on anything specific to military scenarios. Table 1 shows detailed simulation parameters.

We assume that any node that wishes to communicate with any other node in the network must first contact the MOCAs to either get the peer's certificate or to check the revocation status of the peer's certificate it acquired previously. The certification request pattern for the 150-node scenarios uses 100 non-MOCA nodes, each making 10 certification requests randomly distributed through the simulation timeline, for a total of 1000 certification requests. For the 300-node scenarios, 200 non-MOCA nodes make 10 certification requests each, adding up to a total of 2000 certification requests. Each requesting node makes one request per minute on average during the course of simulation. This is roughly 100 or 200 requests per minute and we believe that this is a reasonable number if not too pessimistic. Assuming each certification request precedes initiation of a new secure communication, starting one secure communication session per node per minute should be more than adequate for ordinary mobile nodes. Node movement follows the random waypoint mobility model implemented in the CMU Monarch extension [6] with pause times of 0 and 10

Total Number of Mobile Nodes	150 or 300
Number of MOCAs	30 or 50
Area of Network	1000m x 1000m
Total Simulation Time	600 seconds
Number of Certification Requests	10 requests each from 100 or 200 non-MOCAs
Node Pause Time	0, 10 seconds
Node Max. Speed	0, 1, 5, 10, 20 ms

Table 1: Simulation Parameters

seconds and maximum speeds of 0, 1, 5, 10 and 20 ms. Our simulation results show consistent results over different pause times, speed patterns and also number of MOCAs. Therefore in this section we only present the results for 0 second pause time, 10 m/s maximum speed and 30 MOCAs. Each line in Figures 1, 2, and 3 represents an average of three different runs with different mobility scenarios.

5.2 Flooding vs. Unicast

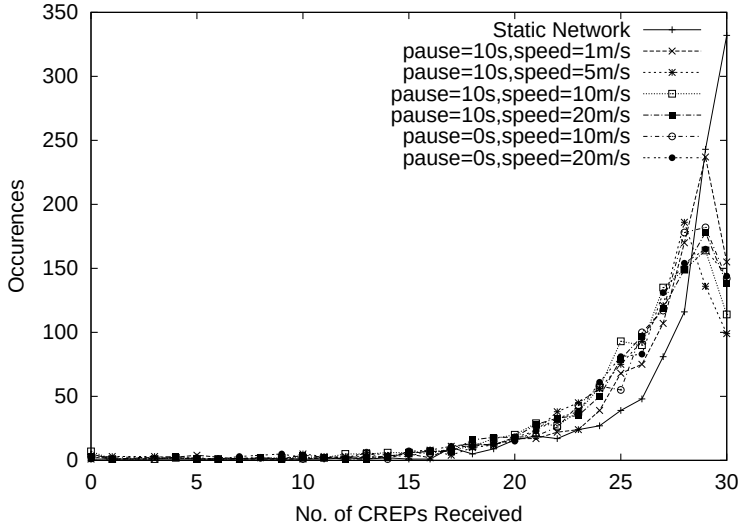

Figure 1: Flooding-based Certification Protocol

To evaluate the effects of employing unicast-based optimization, we first present results from a pure flooding-based approach. Figure 1 shows the number of CREPs received per CREQ under varying mobility. Under a stationary network, represented by the solid line, the flooding-based approach works very well. Almost all CREQs reach all 30 MOCAs and most CREPs make their way back to the client. The reason some of the CREPs get lost (there are many occurrences of nodes receiving 25 to 29 CREPs) is due to temporary network contention caused by the reverse packet storming effect generated by multiple CREPs traveling back to the client at almost the same time. As can be observed from the graph, a value of 15 or 20 for k can result in more than a 90% success ratio under all mobility scenarios and proves that flooding is indeed a very effective means of eliciting responses in ad hoc networks. More details on the flooding-base certification protocol can be found in [17].

Figure 2 shows the results from the Closest-Unicast approach with varying values of the unicast threshold β, with one line for the flooding-based approach for comparison. Consistent with the previous figure, the flooding line has a very high peak around 30, which is the number of MOCAs in the network. Each Closest-Unicast line has two peaks: one at β and another at 30. The peak around β shows that unicast is being used and works well.

Table 2 presents the total number of requests made as well as the number of requests using flooding and unicast. Note that the use of unicast CREQs decreases with higher β values, causing the height of unicast peaks in Figure 2 to

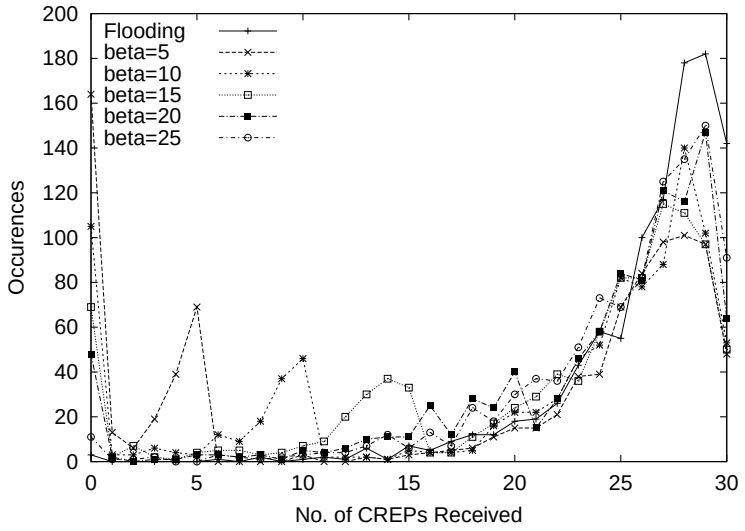

Figure 2: A Unicast-based Certifi cation Protocol with varying β (using Close-unicast)

Table 2: Effect of β on Usage of Unicast

β	5	10	15	20	25	Flooding
Use of Unicast CREQs	337	241	200	172	128	0
Use of Flooding CREQs	663	759	800	828	872	1000
Total No. of CREQs	1000	1000	1000	1000	1000	1000

decrease as β increases. For higher β values, it is more likely that not enough routes to MOCAs will be cached, hence unicast-based optimization is used less often.

Figure 3 presents a comparison of the three unicast-based approaches. The unicast threshold β is set to 15, which can be translated into $k = 10$ with $\alpha = 5$ or $k = 12$ with $\alpha = 3$. We can observe that Closest-Unicast performs best with unicast CREQs. Closest-Unicast also induces the least overhead among the three unicast-based approaches as shown in the next subsection. For the rest of this section, we use Closest-Unicast as our example except when providing a comparison between different unicast approaches.

5.3 Packet Overhead

We evaluate communication overhead, as measured by the total number of control packets used for certifi cation services. Table 3 shows the overhead from flooding and various unicast-based approaches under varying unicast thresholds, β. Generally, unicast-based approaches save 5 to 20 percent of control packet overhead. As the node chooses unicast more aggressively with lower β, the savings are increased. Note that when β is 20 or 25, there is little improvement over flooding. In these cases, β is very high and unicast is not used often since many nodes do not have enough cached routes to MOCAs. This causes most certifi cation requests to fall back to flooding, generating a similar amount of overhead as in flooding. Also, the amount of traffi c generated by β unicast CREQs increases as β increases, adding more overhead. In a more reasonable scenario of β =15 or less, unicast-based approaches save between 15 to 30 percent as compared to flooding. Setting β as low as possible results in the best improvements in overhead but has the adverse effect of implicitly lowering the upper bound of crypto threshold *k* to a very small number, endangering the security of the whole framework as described in Section 2.

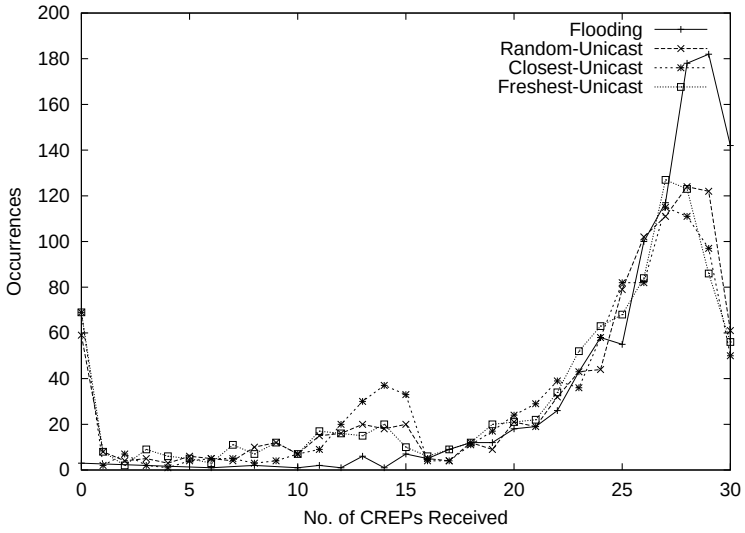

Figure 3: Comparison among Unicast-based Optimizations, β = 15

5.4 Certification Delay

The most frequent use of a certification service is to acquire the communicating peer's public key certificate. The delay to get the certification service is added to the start-up latency of any secure communication relying on PKI.

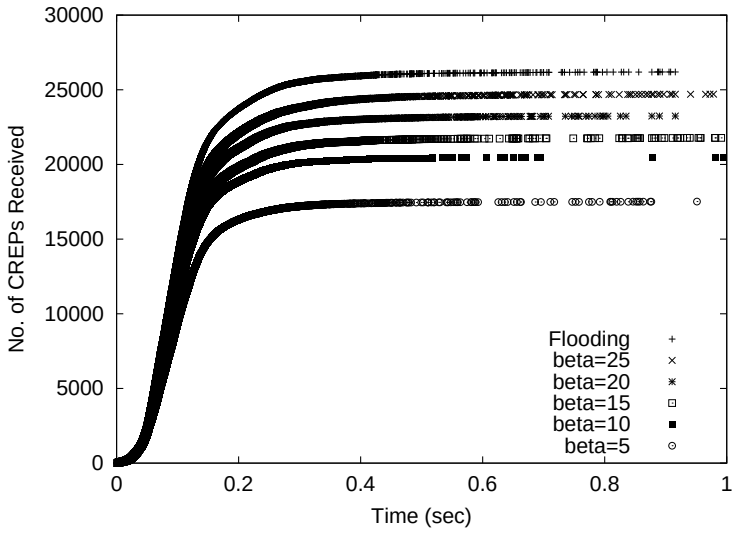

Figure 4: No. of CREPs received over the course of time, using Closest-Unicast

Figure 4 shows the distribution of arrival times of CREP packets with the Closest-Unicast approach with varying β under a moderate mobility pattern of 0 pause time and 10 ms maximum speed. Also, a line for flooding is presented for comparison. Over all cases, the lines flatten out quite quickly, indicating that a client can expect to receive most pending CREPs within 0.3 seconds from the time of certification request. If the client does not collect enough CREPs within that time, the chances are very slim that enough CREPs are in in-flight to arrive later and fulfill the certification request. Based on a appropriately chosen time-out threshold τ, a client can operate efficiently without wasting time. To clarify the choice of 0.3 seconds, Figure 5 shows a normalized view of Figure 4. The choice between flooding and unicast-based optimizations or the choice between different β values does not affect the timing behavior. This indicates that only the density of MOCA nodes affects timing behavior. If MOCAs are densely deployed, a client has

Table 3: Packet Overhead, n = 30

Number of Packets	CREQ	CREP	Total	Ratio to Flooding (%)
Flooding	119642	77959	197601	100
Random-Unicast				
β = 5	84230	54337	138567	70.1
β = 10	97132	61920	159052	80.5
β = 15	105599	67276	172875	87.5
β = 20	110217	69903	180120	91.2
β = 25	114805	73321	188126	95.2
Closest-Unicast				
β = 5	83174	54492	137666	69.7
β = 10	96781	62258	159039	80.5
β = 15	103749	66626	170375	86.2
β = 20	108543	68821	177364	89.8
β = 25	113859	73204	187063	94.7
Freshest-Unicast				
β = 5	85668	54966	140634	71.2
β = 10	97578	62470	160049	81.0
β = 15	105818	67285	173103	87.6
β = 20	111637	70619	182256	95.2
β = 25	114807	73454	188261	95.3

a better chance to discover enough MOCAs faster.

To get a better understanding of this graph, Figures 6 and 7 show a more detailed look at two of the lines from Figure 4.

Figure 6 shows the success ratios for different τ and *k* for the flooding line in Figure 4. When $\tau = 0.1$ seconds, the success ratio drops rapidly as *k* increases. As τ increases, the success ratios with higher *k* values approach a stable value. These results support the choice of 0.3 sec for τ. Similar trends can be observed from Figure 7 for Closest-Unicast. The success ratios of higher *k* values stabilize faster than in Figure 6. For example, the success ratios do not change very much after 0.2 seconds, because the MOCAs are chosen based on the hop count in Closest-Unicast and the CREPs will arrive earlier from the close MOCAs.

One thing noticeable from the two detailed looks at the success ratio is that α plays an important role in determining the success ratio within a given τ. For example, the set of leftmost data points in Figure 6 represents the success ratio with τ set to 0.1 seconds. Each point in the set represents the success ratio under varying values of *k*. For example, when *k* = 1, which is practically a replicated CA case, the success ratio within 0.1 seconds is almost 98%. In comparison, when *k* is set to 10, the success ratio within the same time-out threshold drops down to little less than 70%. The same general trend can be observed over all sets in Figure 6 and also with the unicast-based approach in Figure 7.

These detailed graphs can be helpful when deciding an adequate τ for a given *k*. For example, if *k* is set to 15 out of 30 MOCAs for a network using Closest-Unicast, τ can be chosen to be 0.2 seconds to maintain higher than 90% success ratio.

6 Future Work

Our future work is in two directions. First, we are planning to optimize the current certification protocol to be more efficient and adaptable. Second, we are investigating possible extensions of our framework to address the network partition problem and to integrate with other security services for ad hoc networks.

Our current β-unicast approach only exploits information in the local routing cache of a client. One potential extension is to let a node browse into neighboring nodes' routing tables. For example, a node may have one or two

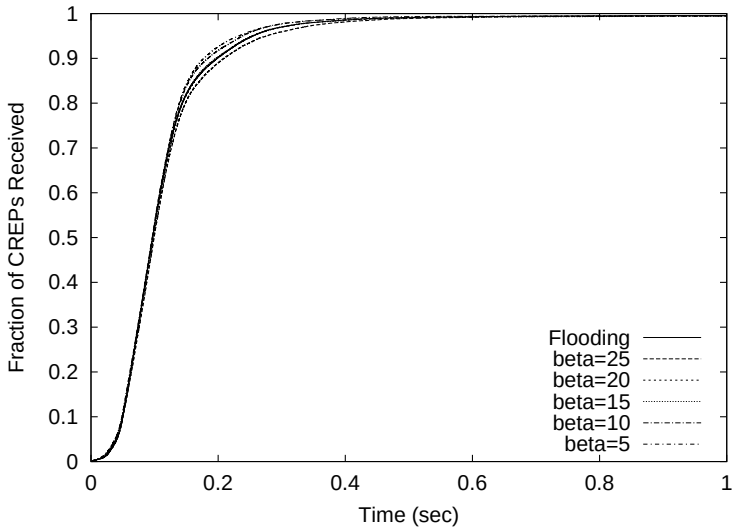

Figure 5: No. of CREPs received over the course of time, Normalized, using Closest-Unicast

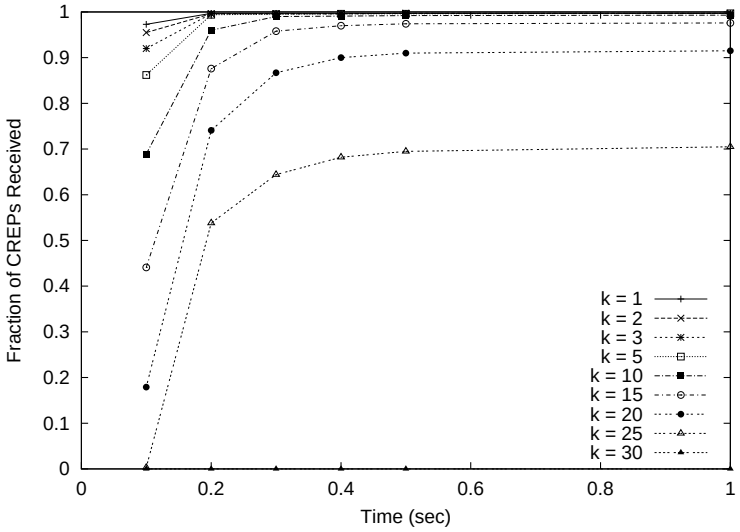

Figure 6: Success Ratio with Flooding

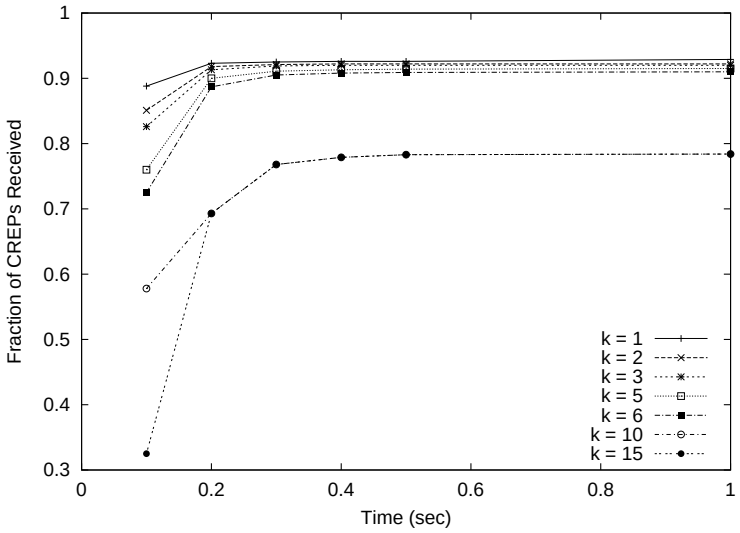

Figure 7: Success Ratio with Closest-Unicast

cached routes short of β and will have to fall back to flooding. If the node can peek at a neighbor's routing tables and find new cached routes, it can enable β-unicast and avoid flooding. The potential overhead from this approach would be the extra communication required between neighbors to exchange the information in routing tables. Whether the benefit would surpass the overhead is an interesting question to investigate.

All unicast-based approaches in our current protocol do not take into account the direction of CREQs. For an extreme case, all the MOCAs picked by our unicast approach could reside on one side of the network from the requesting node. Then it is possible that all the CREQs are sent into one direction sharing the same next hop nodes, potentially causing unnecessary contention that leads to a failure or at least delayed responses. One possible solution for such a situation is to utilize the next hop field in the cached routing table entries. For example, by selecting a set of MOCAs with all different next hops, we can expect to have a spatial load balancing effect in that each CREQ will go out in different directions.

Another interesting direction we plan to investigate is dynamic adjustment of the time-out threshold τ. As presented in Section 5, τ can be selected based on the MOCA density in the neighborhood, which is likely to change as the nodes move around in the network. We plan to investigate the mechanisms to adjust τ to reflect the updated perception of the new neighborhood, hence reducing the certification delay to a minimum.

While we have designed a PKI framework that provides balanced support for security and availability, we cannot avoid the inherent problem of ad hoc networks: unstable connectivity. In a pathological case, if the network is partitioned and there are less than *k* MOCAs available in a partition, it is simply not possible to get a certification service. Although we do not expect to see this kind of problematic situation too often or for too long a time period, if this happens our approach became powerless. To provide certification support for such scenarios, we are currently developing an extension of the MOCA by introducing a hybrid approach of MOCA and the PGP "Web of Trust" model. In the extended MOCA (EMOCA) framework, any node certified by *k* MOCAs will have the capability to act as a delegate of MOCAs to authenticate and issue certificates to new nodes or yet uncertified nodes in the network. If a node wishes to get a certificate but cannot reach enough MOCAs, it can then contact any nearby certified nodes and request a *temporary certificate*. Any already certified node can issue a temporary certificate based on its own authentication of the new node. This temporary certificate carries relatively small confidence compared to the one issued by MOCAs but still can be used as a temporary means for confirmed identity. Conceptually, a certificate issued by MOCAs can be considered as the voucher for confirmed identity by trusted entities (i.e. MOCAs). A temporary certificate serves a similar goal but with a smaller confidence value since the vouching entity is not a trusted entity but only a confirmed member of a network. In our preliminary investigation, we have discovered several interesting features of this hybrid approach and are currently studying the interaction between MOCAs and the delegates and their effects on performance and security.

There are many interesting and promising security services and applications that can be deployed in ad hoc net-

works using the support of PKI. For example, some secure ad hoc routing protocols that assume the existence of PKI support can readily utilize our framework [16, 14]. However, it is yet unclear how these different security services and applications will fit with each other. We plan to study how our approach can be integrated with other security services or applications and what kind of effects will occur.

7 Conclusion

In this paper, we present a practical key management framework for ad hoc wireless networks. We clarify the necessity and the problem of providing a PKI framework for ad hoc networks and identify the requirements for such a framework. Based on our observation of the potential heterogeneity among mobile nodes, we provide an intelligent way to pick a set of CA nodes. These selected secure nodes are called MOCAs and share the responsibility of collectively providing the CA functionality for an ad hoc network using threshold cryptography. To minimize the usage of scarce resources in mobile nodes, we develop a set of efficient and effective communication protocols for mobile nodes to correspond with MOCAs and receive certification services. Our simulation results show the effectiveness of our approach and we provide some insights into the configuration of such security services in ad hoc networks.

References

[1] Thawte, Inc. Company homepage available at http://www.thawte.com/.

[2] The Network Simulator - NS-2. Available at http://www.isi.edu/nsnam/ns/.

[3] VeriSign, Inc. Company homepage available at http://www.verisign.com/.

[4] A. Shamir. How to Share a Secret. *Communications of the ACM*, 1979.

[5] A. Arnes. Public key certificate revocation schemes. Available at http://citeseer.nj.nec.com/arnes00public.html.

[6] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of IEEE/ACM MOBICOM 98*.

[7] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *The Second IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, USA, February 1999.

[8] Y. Frankel and Y. G. Desmedt. Parallel Reliable Threshold Multisignature. Technical Report TR-92-04-02, Univ. of Wisconsin–Milwaukee, 1992.

[9] Janne Gustafsson, Janne Lassila, and et al. Pki-security in mobile business - case: Sonera smarttrust. Available at citeseer.nj.nec.com/466933.html.

[10] J.-P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceeding of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 01)*, 2001.

[11] J. Broch and D. B. Johnson. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. IETF Internet Draft, October 1999.

[12] J. Macker and M. Corson. Mobile ad hoc networking and the IETF. *Mobile Computing and Communications Review*, 1998.

[13] J. Macker and S. Corson. Mobile Ad-hoc Networks (MANET) Charter. IETF Working Group.

[14] K. Sanzgiri and B. Dahill and B. Levine and C. Shields and E. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP)*, November 2002.

[15] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. In *Proceedings of ICNP '01*.

[16] M. Zapata. Secure Ad Hoc On-Demand Distance Vector (SAODV) Routing. IETF MANET Mailing List, Message-ID:3BC17B40.BBF52E09@nokia.com, Available at ftp://manet.itd.nrl.navy.mil/pub/manet/2001-10.mail, October 8 2001.

[17] S. Yi and R. Kravets. Practical PKI for Ad Hoc Wireless Networks. Technical Report UIUCDCS-R-2002-2273/UILU-ENG-2002-1717, University of Illinois at Urbana-Champaign, May 2002.

[18] V. Shoup. Practical Threshold Signatures. In *Theory and Application of Cryptographic Techniques*, pages 207–220, 2000.

[19] Sthephen Kent and Tim Polk. IETF Public-Key Infrastructure Working Group Charter. Available at http://www.ietf.org/html.charters/pkix-charter.html.

[20] W. Diffi e and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 1976.

[21] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, November 1999.

[22] L. Zhou, F. Schneider, and R. van Renesse. Coca: A secure distributed on-line certifi cation authority. Technical Report, Cornell University.

Mediating Between Strangers: A Trust Management Based Approach

Joachim Biskup Yücel Karabulut

Fachbereich Informatik, Universität Dortmund, D-44221 Dortmund, Germany
{biskup,karabulu}@ls6.informatik.uni-dortmund.de

Abstract

Data sources in i-mediation, following property-based security policies, aim at supporting a wide range of potential clients, which are in general unknown in advance and may belong to heterogeneous and autonomous security domains. This raises the challenge how remote and autonomous entities can agree on a common understanding of certified properties, and other issues related to these properties (e.g. encoding formats). This paper proposes solutions that are based on secure i-mediation and a hybrid PKI model, which unifies X.509 and SPKI. We present a mediation functionality, called *f-mediation*. Secure f-mediation assists entities in finding partners for i-mediation and providing them with appropriate certificates and credentials. Thereby, among others, f-mediation deals with delegation and conversion of free properties into capability-like bound properties. An extension to the agent communication language KQML is used to implement the interactions among software agents in an instance of the f-mediation.

1 Introduction

In [3], we proposed a secure mediation approach considering the dynamics and conflicting interests of mediation participants. In a global computing infrastructure like the Internet, entities (strangers) need to reason about the trustworthiness of other entities in order to make autonomous security decisions. Based on this fact, in contrast to traditional (identity-based) access control mechanisms which operate under a closed world assumption, we followed a PKI-based approach in order to achieve the security goals with respect to confidentiality and authenticity. Our approach to secure mediation is based on evidences of clients' eligibility rather than user authentication and access control based on user identities. More precisely, we argued for basing the enforcement of these security goals on certified personal authorization attributes (e.g. organizational membership, security clearance) rather than on identification. In order to focus on the major mediation functionalities and to keep the design of secure mediation manageable, we assumed that the mediation participants agree on a *common*

understanding of personal authorization attributes, credential formats, and the certification policies under which the credentials are issued.

This paper is concerned with solutions to challenges which arise when we remove this assumption. We consider the following situation: Information sources may supply their information for purchase as well as for collaboration. While doing this, they may aim at determining a wide range of potential clients which could be interested in requesting specific services of the sources and which are qualified in terms of evidences of their eligibility.

What is the problem? Whatever data a source has to offer, it may aim at supporting a *wide range* of potential clients, which are in general unknown in advance and may belong to *heterogeneous* and *autonomous* security domains. *Why is the problem a problem?* The conceptual challenge arising in this situation concerns how remote and autonomous entities can agree on a common understanding of personal authorization attributes and other issues related to these attributes (e.g. credential formats). On the one hand, personal authorization attributes are assigned to clients in their autonomously operating security domains, in principle without knowing their later usage. On the other hand, sources independently define their security policies in terms of these attributes. *What is the solution?* In such situations the sources wish to be assisted to determine potentially eligible clients. To reach potentially eligible clients, a source could use a specific mediator which could mediate between a source's property-based security policy and clients' personal authorization attributes which have been asserted by some trusted parties. *Why is the solution a solution?* Electronic business transactions will involve asserted commitments, properties, etc. from many parties and the participants of such transactions will, in general, not be in a position to understand or manage everything that is involved. To reach potentially eligible clients, which might belong to remote security domains, the sources will need to *trust* mediating agents having the required domain expertise as well as the relationships with the potential clients.

As a concrete solution, we propose an additional mediation functionality, called *entity finding mediation*, f-mediation for short. F-mediation employs our hybrid PKI model [4] and our authorization model [3]. In order to prove the key ideas of f-mediation, we extended KQML[1] [12] and developed an agent-oriented and KQML-based prototype implementation [18, 19].

2 Secure I-Mediation

In mediated information systems [26], a client seeking information and various autonomous sources holding potentially useful data, are brought together by a third kind of independent components, called *mediators*. Mediation is required to deal with heterogeneity and the autonomy of the sources, not only from the functional point of view but also with respect to all aspects of security, such as confidentiality and authenticity. The design of our approach of secure mediation [3] is shortly outlined as follows: A client proves his eligibility

[1] KQML stands for Knowledge Query and Manipulation Language.

to see a piece of information by a collection of so-called personal authorization attributes assigned by appropriate trusted authorities. Such assignments are encoded within digitally signed digital credentials[2]. An information source always receives a mediated request to deliver some information together with a set of credentials stemming from the pertinent client. Then the source decides on the permission of the request by evaluating the credentials and the contained personal authorization attributes with respect to its *confidentiality* policy. In case of an allowance, the returned data is encrypted with the public keys found in the credentials on which the permission decision has been based. Thus the returned data can only be decrypted by that client who has proven his eligibility by showing an appropriate collection of personal authorization attributes.

In such scenarios, the mediation participants appear in the following roles: A *client* is characterized by her properties (i.e. assigned by some trusted authorities) and seeks for information. Each heterogeneous and autonomous *source* offers data and follows a security policy expressed in terms of characterizing properties. A *mediator* has two major functions: *a*) From the functional point of view, the main role of a mediator is to retrieve, homogenize and assemble data from any sources the mediator may find worthwhile to contact. *b*) From the security point of view, the mediator contacts only the sources, whose security policies match a client's characterizing properties. Thus, seen from the client, a mediator acts as a kind of *filter* put in front of the sources. We call such a mediator an *information integrating mediator*, *i-mediator* for short, and the process of mediation *i-mediation* [3, 18]. The owner of an *i*-mediator may want to maintain data owned by the *i*-mediator himself. Thus, an *i*-mediator can nearly be treated like a source. As it is a source, an *i*-mediator may also want to protect its data with respect to confidentiality by following a property-based security policy.

3 Hybrid PKI Model Revisited

Most of the works, e.g., [21, 15, 27, 23, 7, 20], investigating the application of certificate/credential-based access control treat current PKI models [1, 11, 5, 6] as competing technologies, even some consider them as dueling theologies [2]. We take a different position. In many real-world scenarios, trust relationships consist of hierarchies, trust networks, and combinations of two. Therefore, we argue that a trust management infrastructure for a dynamic computing environment has to use and to link existing PKI models.

In a distributed system, neither the entities themselves nor their properties are directly visible to other entities. An entity uses the matching public parts of its key pairs as visible surrogates for itself. From the perspective of the visible virtual views, these surrogates are called *principals*. A property assignment to an entity in the (real) world is *presumably_captured_by* a *digital document* in the visible virtual world. Such a document is called a *certificate* or a *credential*,

[2] In the present paper, we would prefer to call the document a certificate in the sense of Section 3.

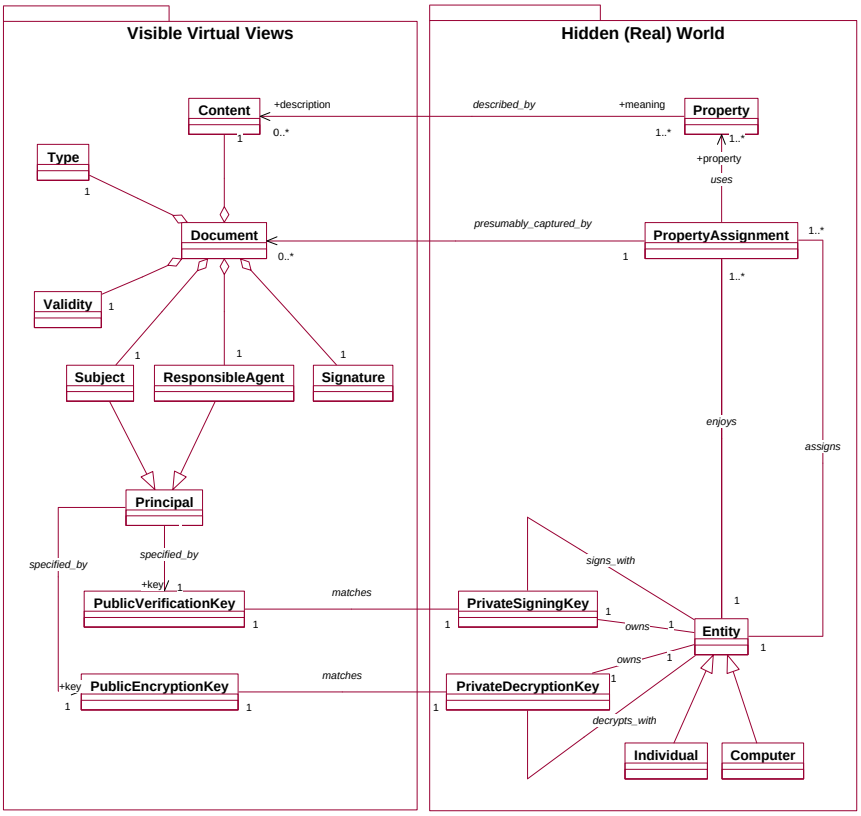

Figure 1: Property assignment: hidden (real) world and visible virtual views

depending on the details explained below. As a consequence, security policies and permission decisions of an entity as a resource owner are solely grounded on the locally available visible view on the global (real) world. This sketched exposition is visualized by Figure 1.

We distinguish two kinds of characterizing properties. A *free property* is intended to express some feature of an entity by itself (e.g. personal data, a technical detail, a skill, an ability). A *bound property* is intended to express some relationship between a client entity and another entity which might act as a server (e.g. a ticket, a capability, a role). While enjoying a free property usually does not entail a guarantee to get the permission for a specific service, enjoying a bound property entails the promise to get a specific service as expressed in the relationship. The assignment of a characterizing property to entities is regulated by corresponding *administrative* properties (e.g. delegatee, licensor) for the characterizing property which must be hold by the entities as responsible issuers.

Following and extending the basic approach of X.509 [1], free properties and the corresponding certificates[3] are handled by trusted authorities using licencing (as shown on the left side of the upper part of Figure 2). The crucial point here is that in general the issuer and the holder of the certificate are different from the entity which afterwards inspects the certificate.

Naturally, it should be possible for any entity, as owner of its resources, to define his own vocabulary for bound properties, to grant corresponding digital documents, and even, to express his trust in delegatees, each of which is entitled to assign a specific bound property (defined in the owner's vocabulary) to other entities. Following and extending the basic approach of SPKI [11], bound properties and the corresponding credentials are handled by owners of services using delegation (as shown on the right side of the lower part of Figure 2).

We are mainly interested in the analysis of how an entity can exploit its free properties in order to acquire bound properties or administrative properties for bound properties. Our hybrid PKI model suggests protocols which follow a *property conversion policy*. A property conversion policy specifies *which set of free properties an entity has to enjoy in order to obtain a bound property or a corresponding administrative property assignment.*

The middle part of Figure 2 visualizes the situation. The entity on the right is the grantor following a property conversion policy. The entity in the center requests a promise for a permission, i.e., a bound property. The grantor, after verifying the submitted free property-certificates with the supporting licences, applies his conversion policy on the free properties extracted from the submitted certificates, and finally, if all checks have been successfully completed, grants a bound property-credential where the subject (grantee) is the same as in the submitted free property-certificates. Figure 2 visualizes an instance of the hybrid PKI model linking previous PKI models.

4 Secure F-Mediation

4.1 Requirements

As stated in Section 1, information sources may supply their information for purchase as well as for collaboration. In the former case, the motivation of a source might be to broaden its potential customer base. In the latter case, a source's main motivation might be to broaden its potential collaborators base by sharing its resources with the *locally eligible* users of the potential remote partner-organizations. Then, a source's security policy can be based on the clients' characterizing properties which are related to the clients' organizational activities and responsibilities (e.g. organizational role membership or group membership) within the corresponding organizations to which the clients belong. The intended high level functionality common to both cases is *finding* potential clients and *stimulating* them to access resources. Accordingly, a mediator assisting the

[3] In contrast to X.509 attribute certificates, a certificate in our model is not associated with any identity certificate.

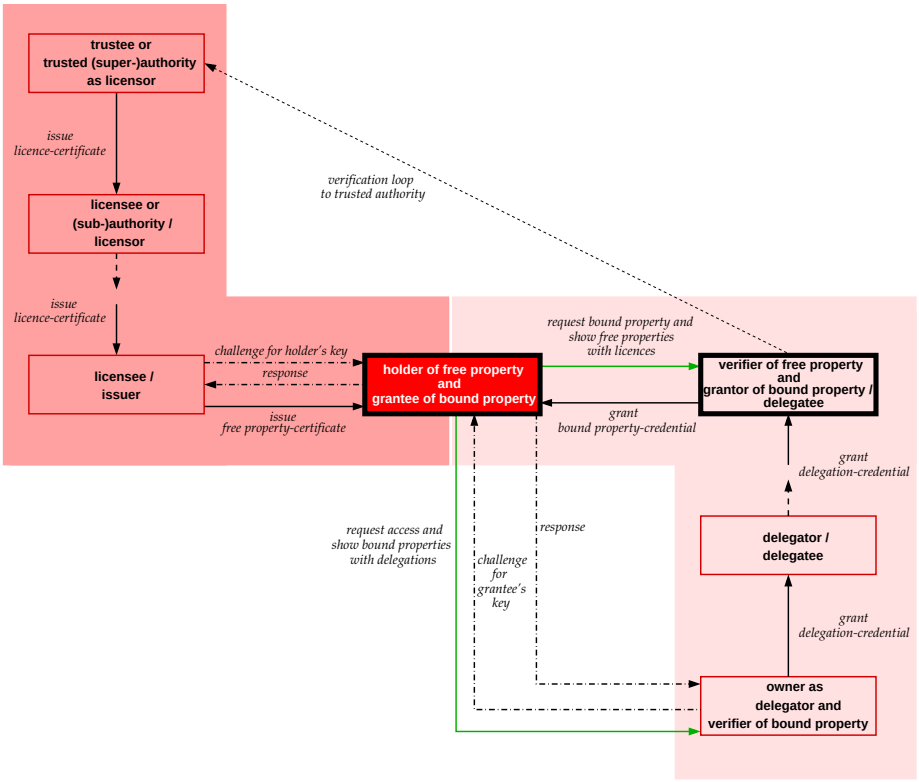

Figure 2: Outline of an instance of the hybrid model for a PKI

sources for this functionality has the following major functions: *a*) From the functional point of view, the main role of a mediator is to seek out an entity B for another entity D and stimulate B to contact D. *b*) From the security point of view, the mediator acts as a broker between independently operating security domains of the mediation participants by mapping the properties of an entity B on the property vocabulary and the security requirements of the other entity D. We call such a mediator an *entity finding mediator*, *f-mediator* for short. The process of mediation using f-mediators is called *f-mediation*.

We consider the following example scenario motivating our design and implementation of the *f-mediation* functionality. Autonomously operating forensic institutions of European countries offer anonymous forensic data about sex offenders, and a fictitious i-mediator conducted by the European Union is specialized in European forensic institutions. Besides serving spontaneous users, the i-mediator may also want to share some part of his data with a discretionary determined kind of users, e.g., the researchers working in US forensic institutions. For this purpose, the i-mediator may follow a security policy specifying that *an entity can be granted the local role "visiting researcher", if she is a "psy-*

choanalyst" and working in a US forensic institute as a "principal investigator". We can hardly assume that the i-mediator, as well as the forensic institutions abroad, can reach a common understanding about personal attributes and organizational roles implicitly. To reach his potential clients, the i-mediator could contact the head of a corresponding FBI unit having the required expertise about mapping between the properties used in the i-mediator's security policy and the properties used in the security domains of the appropriate US forensic institutions.

The other way round, f-mediators can also be utilized by the clients. In this case, a client's motivation might be to determine which *qualified* sources are willing to offer him a specific service and then request them. In such scenarios, the function of a f-mediator is to find the most appropriate sources which, on the one hand, are qualified according to the client's requirements and, on the other hand, may accept clients as potentially eligible entities based on their asserted characterizing properties. For the sake of simple exposition, in this paper, we assume that the f-mediators are only utilized by the sources. However, our design and implementation of f-mediation is flexible enough to realize other recasted scenarios.

Based on the requirements discussed above and and in Section 1, we present a general design for secure f-mediation in the following sections.

4.2 Design

4.2.1 Security Requirement

The following fundamental security requirement is considered:

Any *authorizer* autonomously follows a security policy which ensures that *requested information* is delivered only to appropriate *requestors*. In order to achieve this goal, requestors have to provide *evidence* that they are eligible for requested information, and authorizers have to maintain mechanisms to inspect such evidence and to decide whether and which information is returned. Furthermore, an authorizer has to ensure that information is actually usable to only that requestor which provided the inspected evidence.

4.2.2 Informational Environment

We assume that there exists a trust management infrastructure based on our hybrid PKI model (see Section 3). This infrastructure provides the basic PKI functionalitites required for the design of the secure f-mediation. As indicated in Section 4.2.1, the fundamental security requirement considers entities acting in two modes, as authorizer and requestor, respectively. However, all entities should be able to act in any of these modes during their lifetimes. For the sake of conciseness, for a particular instance of f-mediation, we only consider the specific entities *resource owner*, *f-mediator*, and *client* acting as follows: the *resource owner* only acts as authorizer, the *f-mediator* might act in both modes, and the *client* only acts as requestor.

4.2.3 Interactions and Basic Protocols

The fundamental security requirement (see Section 4.2.1) has several specific interpretations, each of which result from (a) appropriately replacing the mentioned modes (e.g. authorizer and requestor) by two of the three entities in an instance of f-mediation (e.g. resource owner, f-mediator, client) or in an instance of i-mediation (e.g. resource owner, i-mediator, client), respectively, and (b) specifying the kind of requested information and the kind of needed evidence. Tables 1 and 2 summarize the most important interpretations for i-mediation and f-mediation, respectively. These interpretations result from the concrete interactions among the entities involved.

Interaction	***authorizer***	***requested information***	***requestor***	***needed evidence***	***security policy***
I	resource owner	source-specific service	client	free properties	confidentiality policy
II	resource owner	bound properties	client	free properties	property conversion policy

Table 1: Instantiations of the fundamental security requirement for direct contact and i-mediation

Interaction	***authorizer***	***requested information***	***requestor***	***needed evidence***	***security policy***
III	resource owner	administrative property for bound property	f-mediator	free properties	delegation policy
IV	f-mediator	bound properties	client	free properties	property conversion policy
V	resource owner	source-specific service	client	bound properties	reconfirmation policy

Table 2: Instantiations of the fundamental security requirement for f-mediation

In the simplest case, a client contacts directly the source and shows his certified free properties. In secure i-mediation, as designed in [3] and outlined in Section 2, an i-mediator is involved. However, concerning the current point of interest, the i-mediator basically only forwards requests and responses. So we can consider both situations together.

Interaction I: A client acts as a requestor (possibly assisted by an i-mediator). A resource owner follows his confidentiality policy (as security policy) to allow or deny access to content information (source-specific service) based on shown personal authorization attributes (as evidence in form of specific free properties).

Interaction II: In a degenerated form, a resource owner may be able to deal with converting free properties into bound properties on his own. Accordingly, the client (possibly assisted by an i-mediator) applies for a bound property with respect to a source-specific service directly from the resource owner. The source,

after verifying the shown free properties, follows his property conversion policy (as security policy aiming at confidentiality). In the positive case, the resource owner assigns a bound property to the client and grants a corresponding bound property-credential, i.e. to express a possibly conditional permission to access a service.

In more advanced cases, additionally, a *f-mediator* is involved. The trustworthiness of the f-mediator may be determined based on the previous direct organizational or business relationships or on the recommendations or on the evidences of f-mediator's eligibility in terms of certified free properties (e.g. personal authorization attributes). In the context of this paper, we focus on the last case, as examined by the following interaction.

Interaction III: A f-mediator acts as a requestor. The f-mediator claims to be a qualified entity (e.g. the head of a specific FBI unit). He shows his pertinent certified free properties and requests an administrative property for a bound property from a resource owner. The source, as the owner of a specific service and a vocabulary for service-specific bound properties, after verifying the shown free properties, follows his delegation policy as a special property conversion policy (as security policy). In case of allowance, the resource owner assigns an administrative property to the f-mediator and grants a corresponding delegation credential, the content of which roughly means "*can speak for the owner*" (in the sense of [17]) to assign a specific bound property, i.e. to grant corresponding bound property-credentials. F-mediation can be transitively organized by using redelegation of received authorities. A f-mediator (as authorizer) can express his trust in another f-mediator (as requestor) to speak for the former f-mediator in turn. For the sake of simplicity, we don't consider the possible interactions among f-mediators.

Interaction IV: A client shows his certified free properties and applies for a bound property from a f-mediator who is acting as an authorizer on behalf of and in explicit delegation of a resource owner. In order to assign a bound property and a corresponding credential, the f-mediator performs the steps carried out by the source during the interaction **II**.

Interaction V: A resource owner is contacted by a client who requests a service. The resource owner, after verifying the submitted bound property-credentials with the supporting delegations, follows something like his "reconfirmation policy" for bound properties (as security policy aiming at confidentiality) to allow or deny access to content information (as source-specific service) based on shown bound properties.

The security policies applied during the interactions **II**, **III** and **IV** are based on a property conversion policy of our hybrid PKI model, as outlined in Section 3. The high level functionality common to these interactions is *showing* free-property certificates to an authorizer in order to *acquire* appropriate credentials. In the following we sketch a **protocol for secure credential acquisition** recasting our *protocol for secure query answering* [3, 18].

We distinguish a preparatory phase and an acquisition phase. In the *preparatory phase*, requestors and authorizers do not interact yet. A requestor, wishing to acquire a credential later on, collects his free property-certificates. On de-

mand and by interaction, the requestor also gets the issue requirements for the properties to be acquired, i.e. the requestor can ask to be informed about which free properties are likely to be sufficent to acquire which (bound) (administrative) properties. And an authorizer, entitled to assign (bound) (administrative) properties later on, defines an appropriate security policy which relates free properties to the amounts of (bound) (administrative) properties allowed for assignment. The set of free property-certificates, accepted by a security policy as input, must belong to a unique requestor or at least a group of requestors which consciously cooperate. It is not necessary for the authorizer to know the identity of that requestor.

The *acquisition phase* is outlined as follows:

1. The requestor sends a request ⟨return information, requested (bound) (administrative) properties, set of free property-certificates⟩ to the authorizer.
2. The authorizer verifies each free property-certificate and determines the associated free properties.
3. The authorizer evaluates the request by following the pertinent property conversion policy. The resulting set of properties is the intersection of the set of requested properties and the largest permitted set of properties (computed on the basis of the associated free properties).
4. For each of the resulting (bound) (administrative) properties, the authorizer grants a corresponding credential.
5. The signed credentials are sent back following the directions given by the return information.

By making some minor modifications to the protocol sketched above, we get a further protocol fulfilling the functionality outlined in the interaction **V**. For *i*-mediation [3], we presented a similar protocol employing free properties instead of bound properties, as outlined by the interaction **I**.

4.3 Implementation

We have developed an agent-oriented prototype implementation (which constitutes a testbed) for demonstrating the basic functionality of secure *f*-mediation in combination with *i*-mediation. In order to focus on the implementation of the basic *f*-mediation concepts, so far we limited the functionality of an *i*-mediator and data sources to an owner's functionality (see the model of owners and delegations in Section 3 and [4]). Figure 3 shows the security architecture of the implementation, and the structure of the common agent core (see Section 4.3.2).

We have implemented software agents of four kinds according to the activities of the entities involved in such a composed mediation scenario (see Figure 2): *trusted authority agents* representing issuers of free properties and corresponding administrative properties as trusted authorities and licencees; *f-mediator agents* representing verifiers of free properties and grantors of bound properties as delegatees; *user agents* representing holders of free properties and grantees of bound properties; *i-mediator agents* and *data source agents* representing grantors of administrative properties for bound properties, and grantors and verifiers of bound properties as delegators and owners. In Figure 2, *licensees* as well *delegatees* are

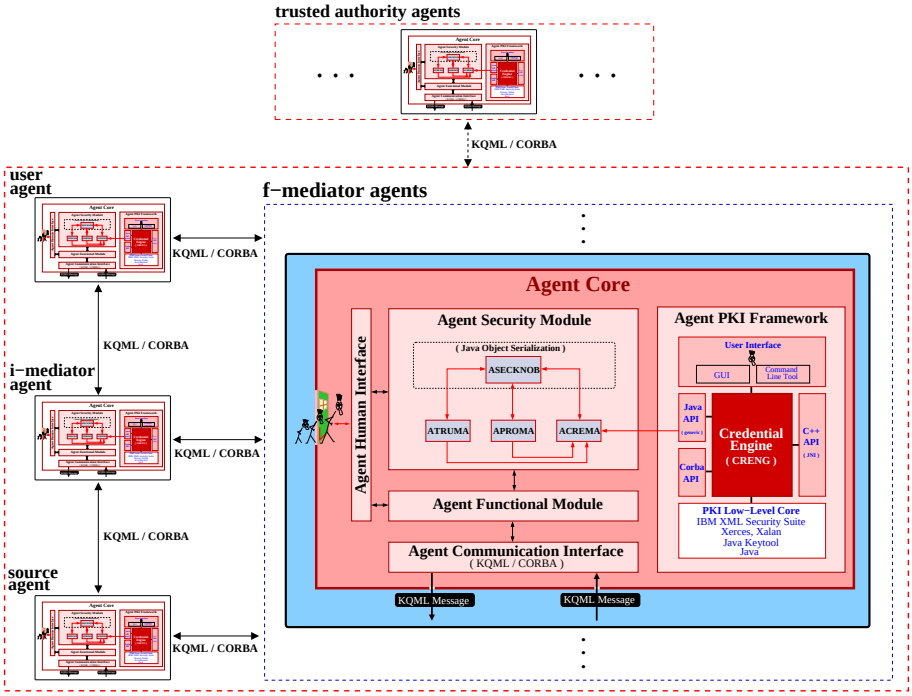

Figure 3: Security architecture

organized transitively, but not further considered in this paper.

For our prototype implementation we primarily focused on three aspects. The first aspect handles the internal authorization model needed for granting privileges including credentials to appropriately represented grantees, and for deciding on the service requests of the requestors. The second aspect is concerned with the structure of the software agents. And the third aspect is related to the KQML-based communication between the agents. We deal with each of these aspects in the following sections.

4.3.1 Authorization Model

As common base for all interactions of f-mediation as well as direct contacts and i-mediation, we need an internal authorization model that provides syntactic means for (a) expressions over free properties as *grantees* (see interactions **I**, **II**, **III**, and **IV**), (b) expressions over bound properties as *grantees* (see interaction **V**), (c) expressions over bound properties as *privileges* (see interactions **II** and **IV**), (d) administrative property for a bound property as *privileges* (see interaction **III**), and (e) source-specific services, interpreted as access allowances, as *privileges* (see interactions **I** and **V**). For this purpose, we employed an extension of our authorization model designed for i-mediation [3, 18], where an authorization is an aggregation including a privilege and a grantee.

4.3.2 The Agent Core

In general, all agents should be able to perform as any of the actors (e.g. grantor, issuer, etc.) during their lifetime (see Figure 2). To achieve this goal, we implemented an *agent core* providing a common *core functionality* which is available to all agents. The main features and the structure of the agent core are reported in [19] and sketched in the following (see Figure 3).

The agent core [18, 19] is implemented in Java and on a Solaris platform. The agent core consists of following five main modules: The **agent security module** has four components. The *agent security knowledge base* ASECKNOB maintains a property database, a trust relationships database containing the public keys of the trusted agents, the certificates and credentials, and a (Horn clause) rule base that specifies implications among properties. The *agent credential manager* ACREMA is programmed to perform the tasks related to issuing and evaluating XML-encoded certificates and credentials. The *agent property manager* APROMA implements a property conversion policy (see Section 3). The *agent trust manager* ATRUMA implements the operations needed to store and to retrieve the information (e.g. public keys) about the trusted agents. The **agent functional module** is a kind of scheduler analysing the incoming KQML performatives (see Section 4.3.3) and scheduling the protocol steps to be executed. The **agent human interface** is designed as an interface for the administrators of the agents to use and set up the corresponding agents. The **agent communication interface** implements classes and functions for sending and the reception of CORBA messages wrapping the KQML performatives. The **agent PKI framework** is a collection of tools providing basic PKI services.

4.3.3 KQML Extensions

The Knowledge and Query Manipulation Language, KQML [12], is a language that is designed to support interaction among intelligent software agents. In a KQML-based agent architecture, the agents communicate by sending certain kinds of messages, called *performatives*, to each other. For example, a KQML performative is the high-level "ask", which demands the recipient for a query operation on a knowledge base. KQML is complementary to approaches to distributed computing, like CORBA, which focus on the transport level.

In the original version of the KQML [12], security issues were not taken into consideration. The works in [22] and [14] made some changes with respect to secure communications and PKI related communications. We followed and extended the approaches of [22, 14] such that they satisfy the requirements of the composed secure mediation. Thereby, we proposed a new KQML ontology and recasted some performatives from [22, 14] and added new ones [18]. The new ontology is called secure mediation PKI, *smpki* for short, and enables the agents to know that the KQML performative they received concerns interactions involving in an instance of the composed secure mediation. We use XML as encoding format for the data communicated through performatives. The recasted

and added performatives are outlined below.

applyDocument This performative has a dual usage. It is used for applying for free property-certificates from trusted authority agents[4] as well as for applying for bound property-credentials (see interactions **II** and **IV**) and for delegation credentials (see interaction **III**). A requestor agent sends the following recasted performative to a corresponding authorizer agent:

```
applyDocument   :language XML
                :ontology smpki
                :content <requested properties and principal>
                [:certificateChains <free property-certificates and
                corresponding licences>]
```

authChallenge and authResponse: These recasted performatives can be employed in all interactions discussed in 4.2.3. For the sake of simple exposition in that section, we omitted the steps corresponding to these performatives. The performative `authChallenge` is used by the agents acting as authorizer. Before issuing a certificate or granting a credential, the issuer or grantor, respectively, has to *challenge*[5] the claiming requestor to prove that she holds the matching private key. In our prototype, the proof is accomplished by an appropriate *response* (i.e. encoded in a corresponding `authResponse` performative) which is generated with the matching private key.

An issuer or a grantor sends the following performative to the requestor agent acting on behalf of the claiming entity:

```
authChallenge   :language XML
                :ontology smpki
                :content <nonce>
```

The content of `content` in the `authChallenge` performative contains a randomly generated string which is to be signed by the claiming entity. The claiming entity signs the string received and sends the following performative including the signed string to the corresponding agent:

```
authResponse    :language XML
                :ontology smpki
                :content <signed nonce>
```

Obviously, also more advanced challenge-response procedures could be exploited.

issueCredential: This recasted performative is used by the agents to issue a certificate or a credential. The agent can send the following performative to other agents which have previously applied for a certificate or a credential by using the `applyDocument` performative:

[4] We omitted this interaction in Section 4.2.3

[5] In some cases, it might be sufficient for a grantor to evaluate solely the certificate chains sent by an agent as requestor before granting a credential, since the grantor may only want to gain assurance, whether the property encoded in the "main document" of a certificate chain is bound to the public key included in this document. In such cases, a grantor might not need to apply a challenge-response protocol.

```
issueCredential   :language XML
                  :ontology smpki
                  :content <issued certificate or granted credential
                  with the corresponding chain of supporting documents>
```

In addition to using the recasted performatives and standard KQML performatives (e.g. `ask_all`, `tell`, etc.), we designed the following new performative.

reduceCredential: The (so far limited) functionality of an i-mediator agent and data source agents is to handle the incoming `reduceCredential` performatives. This performative is designed to be used by user agents in order to apply for a reduced credential. A user agent may send the following performative to some receiver:

```
reduceCredential   :language XML
                   :ontology smpki
                   :content <chain of credentials>
```

The parameter `content` contains a chain of credentials which has been previously gathered from appropriate agents.

The receiver can immediately reduce the chain and sign the resulting reduced credential, if he himself is the origin of the chain. Otherwise, the receiver could still perform the reduction [9], but he cannot properly sign the result. In that case, the receiver forwards the chain to the origin, who in turn sends back the properly signed reduction result via the receiver to the user. Though the receiver, for instance an i-mediator, may not be able to properly sign a reduction result, he can nevertheless base his own access decisions on it.

5 Comparison and Conclusions

Existing works, which are related to the challenges we tackled in this paper, are rooted in three research areas: secure mediation, certificate/credential-based access control, and the employment of KQML for implementing PKI-based security architectures. Contributions to secure i-mediation [8, 25, 10] employ either identity-based or security clearance-based authentication and authorization approaches which appear to be less useful for i-mediation scenarios which we consider. To our knowledge, no credential-based secure f-mediation approach has been proposed to date for establishing interoperability between the entities of two heterogeneous and autonomous security domains. The traditional way of accomplishing this task is to build coalitions between these security domains by committing coalition agreements (e.g. [13]). Such agreements aim at building common vocabularies and a contractually involved cross-certification [24]. The main problem opposing the approach of cross-certification, is that ad-hoc cross-certification between commercial and organizational PKIs is difficult to achieve due to heterogeneous certification policies. Most of the works, e.g., [21, 15, 16, 27, 23, 7, 20, 13], investigating the application of certificate/credential-based access control treat previous PKI models (discussed in Section 3) as competing approaches and base their work on a sin-

gle PKI model. Even some of these works abstract from any particular PKI model (e.g. [13]). In contrast to these works, our proposal is based on a hybrid PKI model. The KQML extensions [22, 14] propose performatives with respect to secure communications and PKI related communications. As demonstrated in Section 4.3.3, our work defines a new ontology, recasts some of their KQML performatives and add new ones.

There are various topics for future research and development. For instance, we would like to integrate distributed role-based access control concepts [20], or negotiations [27, 7] into our approach. Further on, we plan to implement an advanced prototype which allows us to evaluate the actual performance of our approach in terms of effectiveness and efficiency.

References

[1] ITU-T recommendation X.509: The directory - public-key and attribute certificate frameworks, 2000.

[2] Dueling theologies. In *1st Annual PKI Research Workshop*, Gaithersburg, Maryland, USA, Apr. 2002.

[3] C. Altenschmidt, J. Biskup, U. Flegel, and Y. Karabulut. Secure mediation: Requirements, design and architecture. *Journal of Computer Security*. To appear.

[4] J. Biskup and Y. Karabulut. A hybrid PKI model with an application for secure mediation. In *16th Annual IFIP WG 11.3 Working Conference on Data and Application Security*, Cambridge, England, July 2002. To appear.

[5] M. Blaze, J. Feigenbaum, and A. Keromytis. The KeyNote trust management system version 2. RFC 2704, IETF, Sept. 1999.

[6] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *17th IEEE Symposium on Security and Privacy*, pages 164–173, Los Alamitos, 1996.

[7] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM Conference on Computer and Communication Security*, pages 134–143, Athens, Greece, Nov. 2000.

[8] K. S. Candan, S. Jajodia, and V. S. Subrahmanian. Secure mediated databases. In S. Y. W. Su, editor, *12th*, pages 28–37, New Orleans, Louisiana, USA, Feb. - Mar. 1996. IEEE, IEEE Computer Society Press.

[9] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[10] S. Dawson, S. Qian, and P. Samarati. Providing security and interoperation of heterogeneous systems. *Distributed and Parallel Databases*, 8(1):119–145, Jan. 2000.

[11] C. Ellison. SPKI/SDSI certificates. `http://world.std.com/~cme/html/spki.html`, Aug. 2001.

[12] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. M. Bradshaw, editor, *Software Agents*. MIT Press, Cambridge, 1997. `http://www.cs.umbc.edu/kqml/papers/`.

[13] H. M. Gladney. Safe deals between strangers. Technical report, IBM Research Report RJ 10155, July 1999. `http://xxx.lanl.gov/ftp/cs/papers/9908/9908012.pdf`.

[14] Q. He, K. P. Sycara, and T. Finin. Personal Security Agent: KQML-Based PKI. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 377–384. ACM Press, 1998.

[15] A. Herzberg and Y. Mass. Relying party credentials framework. In D. Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference*, LNCS 2020, pages 328–343, San Francisco, CA, 2001.

[16] A. Herzberg, J. Mihaeli, Y. Mass, D. Naor, and Y. Ravid. Access control meets public key infrastructure, or: Assigning roles to strangers. In *IEEE Symposium on Security and Privacy*, Oakland, USA, May 2000.

[17] J. Howell and D. Kotz. A formal semantics for SPKI. In *Proceedings of the 6th European Symposium on Research in Computer Security (ESORICS 2000)*, LNCS 1895, pages 140–158, Toulouse, France, Oct. 2000. Springer-Verlag.

[18] Y. Karabulut. *Secure Mediation Between Strangers in Cyberspace*. PhD thesis, University of Dortmund, Sept. 2002.

[19] Y. Karabulut. Implementation of an agent-oriented trust management infrastructure based on a hybrid PKI model. In *1st International Conference on Trust Management*, Heraklion, Crete, Greece, May 2003. To appear.

[20] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, Berkeley, California, USA, May 2002.

[21] P. Nikander. *An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems*. PhD thesis, Helsinki University of Technology, Mar. 1999.

[22] C. Thirunavukkarasu, T. Finin, and J. Mayfield. Secret agents - a security architecture for the KQML agent communication language. In *4th International Conference on Information and Knowledge Management - Workshop on Intelligent Information Agents*, Baltimore, Maryland, USA, Dec. 1995.

[23] W. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the 8th USENIX Security Symposium*, Washington D.C., Aug. 1999.

[24] J. Tumbull. Cross-certification and PKI policy networking. `http://www.entrust.com/resources/pdf/cross_certification.pdf`, Aug. 2000.

[25] G. Wiederhold, M. Bilello, and C. Donahue. Web implementation of a security mediator for medical databases. In T. Y. Lin and S. Qian, editors, *Database Security, XI: Status and Prospects, Proceedings of the 11th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 60–72, Lake Tahoe, California, 1998. IFIP, Chapman & Hall.

[26] G. Wiederhold and M. Genesereth. The conceptual basis for mediation. *IEEE Expert, Intelligent Systems and their Applications*, 12(5):38–47, Sept.-Oct. 1997.

[27] T. Yu, M. Winslett, and K. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1), Feb. 2003.

Electronic Signature System with Small Number of Private Keys

Ahto Buldas
ahtbu@cyber.ee
Tallinn Technical University,
Tartu University, and Cybernetica AS

Märt Saarepera
marts@neoteny.com
Independent

Abstract

We propose a simple server-based electronic signature system in which a small number of common private keys are used. The motivation of such a system is to escape the scalability and complexity problems that arise if a large-scale Public Key Infrastructure (PKI) is used. We argue that the assumption of personal private keys is the main reason for those problems and high cost of electronic signature systems. We conclude that elimination of personal private keys is justified and further argue that this does not reduce the security.

1 Introduction

Remembering and proving events of the past is an important characteristic of the human civilization. Oral testimonies were used before the literary language was invented, and they are still in use today. Due to increasing complexity of business and public relations, people started to use written documents as external memory. Numerous measures have been developed to protect the content integrity of written documents. For example special inks, paper, seals etc. are used. Further, in order to bind the content of a document with a person responsible for it, handwritten signature is used.

Today, written documents as well as all kinds of data in general are processed, transmitted, and preserved in digitized electronic form (mostly referred to as *electronic content*). Currently, cryptographic means are used to protect the integrity of electronic content and to bind content with persons responsible for it. Cryptographic checksums computed by using asymmetric cryptography are often viewed as electronic analogues of handwritten signatures. Signatures are created by using *private keys* and verified by using *public keys*.

Cryptographers have been studying electronic signature technologies for decades since the discovery of one-way functions [4]. Several electronic signature schemes are (mathematically) proved to be secure under some complexity theoretical assumptions (see [12] for an overview).

In numerous countries, including the USA, electronic signatures are legally admissible. Considering the advantages of electronic data management (creation, transfer, storage) over the traditional paper-based one we could expect that there is an obvious need for electronic signatures in the society. However, electronic signatures are still not widely used. In our opinion, it is for two reasons:

(i) *Security concerns.* Leakage of private keys may cause unlimited risk, because of the number of (possibly forged) signatures cannot be limited.

Secure key management is too complicated for general public.

(ii) *Technical complexity and Cost*. Private key management as well as massive authentic distribution of public keys is costly.

Rapid growth of a technology where the main concerns are related to security and cost – *Internet banking* – suggests that these concerns can be solved for electronic signatures as well. In the Internet banking, (a) the risks are always limited (at least to the amount of money in the user's account), and (b) existing infrastructure (like web browsers) provides simple and user-friendly interface to customers. Why not to design an electronic signature system in a similar way?

In most electronic signature systems it is assumed that private keys are distributed among the users. At the same time, there are electronic signature systems that use a very small number of keys. For example, in the system presented by Asokan et al [2] personal private keys are eliminated and the signature function is delegated to a server. The server authenticates clients and creates electronic signatures in their name by using one single private key. It seems to be a common opinion that electronic signature systems where the private keys are distributed among users are of the highest possible security. We do not think this opinion is sufficiently argued. Moreover, we claim that personal private keys are the main reason for high cost and technical complexity of the systems. Eliminating personal keys may lead to considerably more cost-efficient electronic signature systems, which in addition may be more secure than the previous systems.

In this paper, we show (using an elementary risk analysis methodology) that server-based electronic signature systems may be as secure as those with personal private keys. Moreover, in server-based systems it is easy to restrict the number of signatures, which is a necessary feature to limit risks. In server-based systems, possible abuses are more easy to inspect – the server can log all events, while personal private keys can be abused off-line in an unrestricted way. Hence, even if each (signed) transaction has limited value, an attacker (who abuses a personal key) can still create large numbers of low-value transactions.

We propose a new scalable electronic signature system that uses even a smaller number of keys than the system of Asokan et al [2]. In our system, the signature servers themselves use meta-level signature services to create their signatures so that only few public (and private) keys are needed for the whole service. Due to the small number of keys, it is easy to preserve the validity of signatures in long-term scale.

The paper is organized as follows. In Section 2, we give a general description of electronic signature systems and present two special cases: PKI-based electronic signature, and server-based electronic signature. We point out their relative advantages and drawbacks. In Section 3, we analyze the practical security of electronic signatures, considering both signer's and verifier's view points. In Section 4, we describe techniques to improve the scalability and security of electronic signature systems. In Section 5, we outline a technically simple and efficient server-based solution to electronic signatures.

2 Electronic Signature Systems

In the most general setting, an *electronic signature* [1] is authentic and reliable information that answers the question "Who signed What and When?". In order to use electronic signatures, one has to organize a system that satisfies the following security requirements from both *signer's* and *verifier's* view-point:

[1] We intentionally use the term *electronic signature* instead of *digital signature*, because the latter is mostly used in association with signature schemes that use asymmetric cryptography.

(i) Signers are able to sign messages only in their own name.

(ii) Potential verifier can check the validity of a signature. The verifier is provided with methods that ensure that valid signatures cannot be denied or invalidated later.

In the following, we describe two (totally different) electronic signature systems. The first system has a maximum number of keys – every user has her own private (signature) key. The second system has a minimum number of private keys – there is a single private key that is maintained by a server, which identifies users and creates signatures in their names. We show what are the security and cost concerns in these systems for a signer and for a verifier. The analogies for these two systems in the paper-world are *personal handwritten signature* and *notarized or delegated signature*. We analyze the history of these signature systems and the security concerns in both systems. Massive use of personal handwritten signatures became possible only when literacy became a common skill. Notarized signatures were used way before. We claim that considering the "electronic literacy" of general public, the society is not yet ready to use "personal" electronic signatures. Moreover, we are not able to imagine how "electronic literacy" will become a common skill in the near future.

2.1 PKI-Based Signatures

Each user A has a private key sk_A, which is assumed to be under a sole control of A, and a public key pk_A, an authentic copy of which is assumed to be available to all potential verifiers. To sign a message M, A applies a signature function SIG to a pair (sk_A, M) of the private key and the message. To verify a digital signature $s = \text{SIG}(\mathsf{sk}_A, M)$ one has to apply a verification function VER to a triple (pk_A, s, M), which returns Yes if the signature is correct.

The mechanism for authentic distribution of public keys depends on particular systems. The users may themselves distribute their keys, which is suitable if each user has a small number of communication partners. An example of such system is PGP [14]. In order to simplify the distribution of authentic public keys, a trusted party – Certification Authority CA – is introduced. As any other user, also the CA has its private key $\mathsf{sk}_{\mathsf{CA}}$ and its public key $\mathsf{pk}_{\mathsf{CA}}$. To bind the identity ID_A of A and the public key, the CA issues *public-key certificate* $c = \text{SIG}(\mathsf{sk}_{\mathsf{CA}}, (\mathsf{ID}_A, \mathsf{pk}_A))$. A complete signature of A on M consists of two parts: the signature s and the certificate c. To verify such a signature, one needs to have an authentic copy of $\mathsf{pk}_{\mathsf{CA}}$.

For several reasons, we also have to add time to an electronic signature. In order to prove the time when the signature was created, another trusted party – Time Stamping Authority (TSA) – is introduced [17]. By a time stamp for a signature s we mean a signed statement $ts = \text{SIG}(\mathsf{sk}_{\mathsf{TSA}}, (s, t))$, where t is a time value. Hence, the signature is a triple

$$\underbrace{\text{SIG}(\mathsf{sk}_A, M)}_{s},\ \underbrace{\text{SIG}(\mathsf{sk}_{\mathsf{CA}}, (\mathsf{ID}_A, \mathsf{pk}_A))}_{c},\ \underbrace{\text{SIG}(\mathsf{sk}_{\mathsf{TSA}}, (s, t))}_{ts}, \quad (1)$$

for the verification of which we need authentic copies of two public keys $\mathsf{pk}_{\mathsf{CA}}$ and $\mathsf{pk}_{\mathsf{TSA}}$. Note that the scheme presented above is considerably simplified compared to its real implementations. However, the simplified description is completely sufficient for the goals of this paper.

The installation procedures of private keys, their protection mechanisms, authentic distribution of public keys and their status checking mechanisms make large-scale PKI systems very costly [5]. The main threat for A is that someone abuses her private key. The main threat for a verifier is that the signature (1) becomes invalid, which may happen due to

exposure of the keys $\mathsf{sk}_{\mathsf{CA}}, \mathsf{sk}_{\mathsf{TSA}}$ or due to the cryptographic algorithm SIG becoming insecure. Note that in practice, the signatures can potentially be denied by alleged signers, which is also a threat for the verifier. However, considering the highly non-technical nature of this threat, we do not discuss it here. For example, there may be several different ways of solving "fantom withdrawal" cases between banks and their clients, depending on the contracts (between banks and clients) and the legal environment in which the contracts have been made. We only consider the threats that cause the signature (1) becoming *technically incorrect*.

2.2 Server-Based Signatures

We have a single private/public key pair $\mathsf{sk}_S/\mathsf{pk}_S$ in the system that is maintained by a server S. Every user has means to authenticate herself to the server, in order to create electronic signatures. The exact way how the authentication is performed is not important. The server maintains a database of signature events described as triples (ID_A, M, t). Each such triple means a statement "ID_A signed M at time t". In order to sign a message M, a user A sends S a request which comprises M (or its cryptographic digest). After verifying the identity of A (e.g. via password), S creates and stores a triple (ID_A, M, t), where t is the current time. The verification of a signature is either *server-aided* or *off-line*.

In the case of *server-aided verification*, (a) a verifier B sends M to S, (b) S makes a query to its database and finds all triples of the form $(*, M, *)$ and sends all of them to B. From the technical side, such a scheme is extremely simple and does not require digital signature schemes at all. Though, it has been proved by Halevi and Krawczyk [8] that in password-based authentication protocols (under certain security assumptions) asymmetric cryptography is still necessary.

In the case of *off-line verification*, the server (instead of storing triples in its database) signs a triple (ID_A, M, t) by using its private signature key sk_S and communicates the signature

$$\text{SIG}(\mathsf{sk}_S, (\mathsf{ID}_A, M, t)), \tag{2}$$

back to A. It is not hard to notice that a server-based signature (2) is much simpler than a PKI-based signature (1). Both the installation costs at the user side and the public key distribution costs are lower. The main threat for A is that someone impersonates her during the identity check procedure, which may be possible due to a leakage of passwords etc. The main difference from PKI-based signatures is that the service provider S itself is able to create signatures without users' intent. Hence, S must be absolutely trustworthy. In the next subsection, we argue that trust assumptions in these two systems are only seemingly different.

2.3 Personal and Delegated Signatures: Historical Metaphor

In the case of a hand-written signature, the main skills needed from a person are: (a) knowledge of written language because the signer has to know what she is signing for; and (b) understanding and controlling the functionality of a pen. The signer has to be convinced that the pen cannot sign anything by itself, without user's intent. The "pens" for electronic signatures are much more complicated. The users who really like to have control over their private keys must be well educated in electronics, hardware design, operating system design, the software etc. Even if the signer has all the knowledge necessary to understand electronic signatures, she still does not know whether the signature device really behaves as specified. Trapdoors in software and even in hardware are not just science fiction but are rather

common practice. Today, the assumption that people may have sole control over their private signature keys is thereby just an illusion, and probably will stay an illusion in the near future. No single person (and most of the institutions and companies) is able to control her signature device. At present, the methods and devices to reliably control private keys are affordable to very few institutions in the world. When using electronic signatures, most of us have to trust technology and hence also the providers of technology. In this sense, we are in the role of "illiterate" people.

But also illiterate persons can sign documents: they just write X-s at the bottom of the document in the presence of a *trusted notary* who confirms that the X-s are written intentionally. In the past when overall literacy was not yet established, numerous contracts were signed that way. In some sense, any present-day electronic signature is just a confirmation created by the providers of the technology, who are in the role of "notaries". We cannot eliminate trust by adding technological security measures (like providing users with personal private keys) to the system.

Trust assumptions are only one aspect that affects security. In order to show that server-based signature systems are practically not less secure than the PKI based system, we have to use more precise definitions of practical security. In the next section, we present a practical security analysis that uses a commonly accepted method of practical security evaluation – *risk analysis*.

3 Practical Security of Signature Systems

Theoretical cryptography focuses on preventing particular threats. In practical security, the primary goal is rather to reduce risk. It is possible that preventing a threat does not reduce the overall risk. In this section, we first present the basic principles of risk analysis that are used in later analysis of electronic signature systems. Then, we analyze and compare the security of PKI-based electronic signature systems and server-based systems. We use the so called *attack tree* method[15] that has been successfully used in several practical security-critical systems.

3.1 Threats, Risks, Attacks

Risk is commonly defined as mathematical expectation of loss. This definition is, however, somewhat inconvenient to use when the threats are related to attacks. The reason is that it is often impossible to estimate directly the probabilities of attacks. But attacks are the most important threats if we estimate the security of electronic signature systems.

One of the most methodical approaches to attack analysis is the *attack tree* method [15, 18]. An attack tree is a graph that represents the decision-making process of a well-informed attacker. The roots of the tree represent the main threats, which are the main goals of attackers. Each node represents an attack. The graph has two types of nodes: AND nodes and OR nodes. The child nodes of an OR node represent a list of conditions (sub-attacks) each of which is sufficient for the attack being successful. The child nodes of AND node represent a list of conditions (sub-attacks) each of which is necessary for the attack being successful. The leaves of the tree represent "atomic" attacks the costs (and other characteristics) of which are known.

3.2 Security of Signer

In order to compare the security of PKI-based and server-based electronic signature systems, we use a generic model that simultaneously describes both systems. The model consists of the following parts:

(1) *Client workstation*, which is the signer's interface to the system,

(2) *Technology providers* that produce or sell all kinds of technology used in electronic signature systems,

(3) *Signature server* that participates in the signature creation process (not present in PKI-based signature systems), and

(4) *Signature service* that runs the signature server (not present in PKI case).

In a PKI-based system, Client workstation computes client's signature by using the private key of the Client. The key may be stored in the memory of a workstation or in an IC-card. In a server-based system, Client workstation is connected securely to a Signature server (via a secure SSL connection etc.). After succesfully authenticating the signer (by using passwords etc.) the Server creates an electronic signature in signer's name.

We assume that attackers' main goal (root of the attack tree) is to forge a signature. We consider four general sub-attacks, each of which is sufficient for the goal of the attacker:

(a) *Attack client workstation* - steal the key/password, insert a Trojan horse, etc.

(b) *Bribe an employee of a technology provider* - bribed employees may add vulnerabilities to the system. Yung and Young [19] proved that trapdoors can easily be inserted even into cryptographic algorithms.

(c) *Bribe an employee of the signature service provider* - bribed employees may add vulnerabilities to the system or create forged signatures.

(d) *Attack signature server in a technical way* - try to attack the server and abuse the signature key.

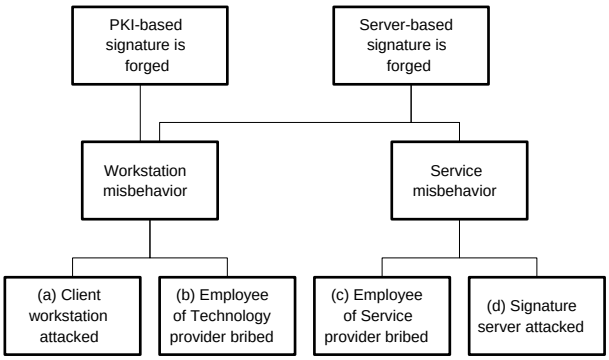

Figure 1: Merged attack trees for PKI- and server-based signatures.

What we claim is that the components (3) and (4) of an electronic signature system do not add additional risks to the system and hence server-based systems are at least as secure as PKI-based systems. This claim is a consequence of the following assumptions:

A0: *A well-informed attacker always chooses the easiest (least costly) attack.*

A1: *It is easier to attack a client workstations than to attack signature servers*: $\mathsf{Cost}[a] < \mathsf{Cost}[d]$, because service providers are commonly more experienced to protect their computers than general public. The above is true for the attacks by outsiders. It may actually be easier for an inside attacker to attack the server. However, once we assume that trusted services may be compromised by insiders, we should also agree that personal keys would not help much. For example, if Microsoft on-line client service is compromised, it would be able to suitably "update" client software in almost any networked client workstation and thereby also to get access to personal keys. Most of the average-skill users do not protect their computers enough to pre-

vent web services from running malicious code in their computers.

A2: *The costs of bribing employees of Technology providers and of the Signature service are comparable*: $\mathsf{Cost}[b] \approx \mathsf{Cost}[c]$. At first glance, this assumption may be doubtful – to bribe scientists and technology experts seems much harder than to bribe a "minimum wage guy" who guards the server room. However, the term *technology provider* in this paper has a wider meaning than in common language. For example, also the shops that sell computers are viewed as technology providers, because they have an influence on the behavior of the computers they sell.

A3: *By using security measures of moderate cost (firewall, etc.) it is possible to make technical attacks to the signature server more costly than bribing an employee of the service provider*: $\mathsf{Cost}[c] < \mathsf{Cost}[d]$.

These assumptions imply that in a PKI-based system (where only (a) and (b) are meaningful attacks) as well as in the server-based system (where all attacks (a),(b),(c),(d) must be considered) either (a) or (b) has the lowest possible cost and hence the attacker always chooses one of them. Thereby, if we have reasonable cryptographic measures used in the signature server, and reasonable organizational means used by the signature service provider, then the attacks (c) and (d) simply do not increase the overall risk of the system.

3.3 Security of Verifier

The most important threat for the verifier is that an accepted valid signature becomes invalid. We only consider the case of off-line verification in both types of systems. We do not consider the attacks that target verifier's workstation because these attacks are equivalent in both types of signature systems. The most important (threats) attacks to consider are the following:

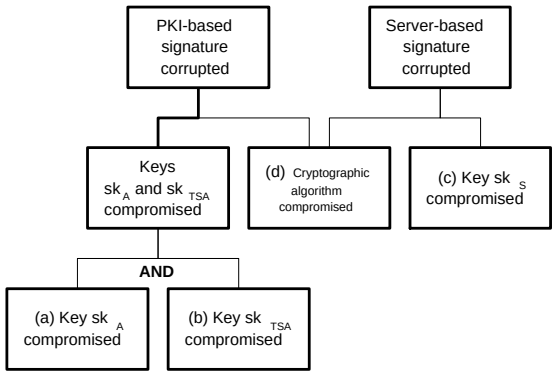

Figure 2: Merged attack trees for signature verification.

(a) Private key sk_A of the signer becomes compromised: either because of attacker or intentionally by the signer (in order to escape from liability).

(b) Private key $\mathsf{sk}_{\mathsf{TSA}}$ of a time-stamping service becomes compromised.

(c) Private key sk_S of a signature server becomes compromised.

(d) Cryptographic algorithm becomes compromised.

We assume that the Signer and the CA have a mutual written contract, which states that the Signer possesses (and agrees to use) a particular key. This contract can be used as evidence in later disputes, if the Signer tries to deny having been related to the key. Hence, the compromise of CA key alone does

not affect the validity of PKI-based electronic signatures, because the certificate is just a copy of a written contract.

If we compare a PKI-based signature with a server-based signature, we notice that the validity of them relies on the validity of cryptographic algorithms and keys. At first sight, it may seem that server-based signature is more easily corrupted because its validity depends on the validity of one single key (pk_S), while the PKI-based signature becomes corrupted only if two keys (pk_A and $\mathsf{pk}_{\mathsf{TSA}}$) are compromised. Note, however, that in most signature systems, users are allowed to revoke their signature keys. Once A decides to deny her signature, she may try to make her signature technically invalid by immediately revoking her key. Hence, also the validity of PKI-based signature depends on a single key - $\mathsf{sk}_{\mathsf{TSA}}$ - and hence, from the viewpoint of the verifier, there is no difference between the security of the said two signature systems.

4 Techniques to Improve Efficiency and Security

As shown above, we can reduce the cost of electronic signatures by eliminating personal private keys and the related PKI. In this section, we describe some state of the art techniques to improve the efficiency and security of electronic signature systems. Batch signatures [13] is a solution to efficiency and multi-component signatures increase the security of electronic signature system. We also discuss the *randomly chosen servers* approach that was proposed by Haber et al [7] and observe that in the context of our electronic signature system this approach is impractical because of large signature size.

4.1 Batch Signatures

One of the main problems of server-based electronic signature systems is their low scalability. The reason of the problem is that asymmetric cryptography is slow. *Batch signatures* [13] is a method that allows one to sign a multitude of messages at the time and thereby to speed up the signature process. The most efficient batch signature scheme [13] is based on Merkle hash trees [9, 10]. It was first proposed by Micali [11] but was later "rediscovered" by several researchers [13, 6, 1]. For creating a (Merkle tree based) batch signature for a list of messages $M_1, \ldots, M_\ell$, a signer first composes the messages using a Merkle tree [9]. The resulting hash value d is then signed by an ordinary signature scheme. Each message M_i is then provided with a pair of (a) the ordinary signature on d, which is common for all messages, and (b) an authentication path $\Pi_i = \Pi_i(M_1, \ldots, M_\ell)$, which proves that M_i took part of the computation of d. If the number of messages is large, we achieve up to thousand-fold speedup in computations. It is argued in [13] why this scheme is as secure as ordinary digital signature schemes.

Batch signatures are not recommended for end users – the number of signatures is not limited and hence the risk is indefinite. For service providers, batch signatures could be the basic mechanism to achieve scalability.

4.2 Multi-Component Signatures

In a server-based signature scheme the server must be ultimately trusted. There is no way to prevent the server from creating signatures in users' name. One way of reducing the trust assumption is to use *threshold trust*. Suppose, we have a multitude of servers $N_1, \ldots, N_n$, each N_i possessing a private signature key sk_i with the corresponding private key pk_i. For signing a message M, a user P authenticates itself to

all servers and sends M to each server. By a *multi-component signature* on a message M given by a user A, we mean a sequence of digital signatures

$$\text{SIGN}[M] = (\text{SIG}_{\mathsf{sk}_1}(M, \mathsf{ID}_A), \ldots, \text{SIG}_{\mathsf{sk}_n}(M, \mathsf{ID}_A)).$$

The signature of A on M is defined as *valid* if at least $t > 1$ servers have signed (M, ID_A). If the servers are controlled by independent parties then the risk of simultaneous misbehavior of these servers is quite low. In particular, no single server can sign in A's name. Different threshold signature schemes are extensively researched [16].

As shown in Section 3, unconditionally trusted server is not the hardest security problem for large majority of users. We can conclude that applying threshold trust at end user level is hardly practical. However, *multi-component signatures are still useful to the service providers,* who are able to guarantee sufficient level of security in their servers.

Multi-component signatures remain valid even if one of the (component) signatures is corrupted, because the other components still protect the authenticity of electronic signature. It the components are created by using different signature schemes then the signature resist the breakage of signature schemes.

Another (more complex) approach is to use *shared signature schemes* [16]. The key is shared between a multitude of servers so that only a coalition of t servers are able to produce a valid signature. The main advantage of such approach is that only one ordinary digital signature is produced, and hence, the size of a signature is smaller than in the multi-component signature approach. Main drawback of shared signatures is that they do not withstand the breakage of a signature scheme. Hence, we prefer the use of multi-component signatures.

4.3 Randomly Chosen Servers

Haber et al [7] proposed a method how to use smaller threshold values t, so that the system would still be relatively secure against attacks performed by $p > t$ colluding servers. The main idea is to use a public pseudo-random function G, which given as input a message M outputs a list of t servers the signatures of whose are necessary for the multi-component signature on M being valid. Their solution may seem very attractive but it leads to impractically large signatures.

If there are n servers in total and a set S of p malicious servers try to forge a signature on M. Attacker chooses slight modifications M' of M, so that the meaning of M' stays almost the same (by rewording sentences or changing numerical values etc.). The goal of this attack is to find M' such that $G(M') \subseteq S$. For each M', the probability that $G(M') \subseteq S$ is $\pi = \left(\frac{p}{n}\right)^t$. The probability of success after N trials is $\pi_N = 1 - (1-\pi)^N$. Note that in most cases, there is no problem to generate a large number of modifications of M. For example, if there are 30 words in M that have at least one synonym then the number of modifications is 2^{30}. If we have two numerical values each having at least one thousand modifications then we have one million modifications. Hence, one may first fix the words and numerical values that may be changed and then use a computer to generate all combinations M' one at a time and check whether $G(M') \subseteq S$. Hence, the number N of trials must be sufficiently large. From the assumptions $\pi_N \leq 1/2$ and $N = 10^k$ we conclude that $t \approx k \cdot \frac{\ln 10}{\ln \frac{n}{p}}$. If one third of the servers are corrupted ($p/n \approx 1/3$) then $t \approx 2k$. Hence, if $N = 10^{20} \approx 2^{66}$ then $t \approx 40$ which is clearly impractical. If again $t = 10$ (which correspond to a reasonable signature size) then an adversary must try about 10^5 random modifications, which is feasible to almost any computer. Hence, for this method to increase practical security, the size of

multi-component signature must be few hundred K bytes. For this reason, we do not use the pseudo-random choice method in our system.

5 Electronic Signature Service

Our goal is to design a server-based signature system that is capable of serving billions of clients. The cost of the system must be much lower than PKI-based solutions, while the security must be comparable or better.

As our goal is to minimize the number of private keys in the system, we use two layers of servers (Fig. 3). The front-end *Proxy servers* authenticate clients and process signature requests. The back-end *Notary servers* sign the processed requests.

Each Notary server can serve up to one thousand Proxy servers. Since we use multi-component signatures, each Proxy uses at least two Notary servers. Consequently, in a system with few thousand Proxy servers we need about ten Notary servers. Such a service would potentially be capable of serving the whole on-line Internet community. Users of such system would need just a web browser to sign or verify messages. User authentication could be carried out with tools already incorporated in web browsers (including PKI-based authentication).

As there are about ten key-pairs in total the system has potentially enough resources to guarantee sufficient protection of private keys. The Public Key Infrastructure related to authentic distribution of public keys is very small and could be efficiently implemented. All keys could be stored in browsers' code and hence their use could be completely transparent to end users. Even if Notary keys are changed annually, all the history of keys would still fit into the code of web browsers for hundred years.

5.1 Authenticating a User

User authentication is one of the most costly parts in electronic signature systems. To create a reliable database for user authentication, we probably need face to face communication with all clients. Assuming that only 15 minutes is spent for each user, we deduce that to create a database for one million users, we need at least 1300 man months in total. However, we mostly do not have to start systems from scratch – there are many client bases already developed. For example, numerous banks have internet banking systems with several hundred thousands clients. Though the authentication methods used are different, it would still be reasonable to reuse the existing authentication systems rather than built new ones from the scratch – that would reduce the overall costs. In server-based electronic signature system, the use of a variety of different authentication methods does not affect the simplicity and uniformity of electronic signatures, because only the result of the authentication is included into the signature.

5.2 Signing a Message

For signing a message M, a user A authenticates itself to a Proxy server P and sends M to P. Proxy server immediately replies with electronic signature, which can be verified in client's browser. For users, electronic signature system is just a web service.

5.3 Creating a Signature

After successful authentication of a user, a Proxy server composes a signature statement (M, ID_A) that includes the message M to be signed and a representation of user's identity. The statement may comprise other information, like signature policy, liability constraints, time/date, etc. Proxy server does not sign each signature statement separately, but instead works in rounds and signs the signature statements

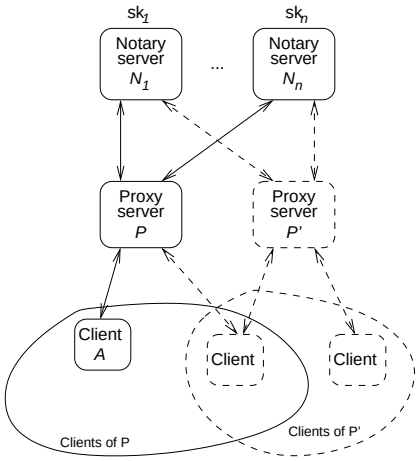

Figure 3: Main structure of the signature system.

in "batch mode". During each round, it collects signature statements. At the end of a round, Proxy server computes a cryptographic digest d of all statements of this round and sends d to n Notary servers $N_1, \ldots, N_n$.

Each Notary server N_i authenticates P and signs a triple (d, ID_P, t_i), where t_i is the current time, and ID_P is a representation of P's identity. Strong cryptographic authentication, like Message Authentication Codes [12] can be used to make impersonation of Proxies very difficult. Having received digital signatures $\mathrm{SIG}_{\mathsf{sk}_1}(d, \mathsf{ID}_P, t_1)$, ... , $\mathrm{SIG}_{\mathsf{sk}_n}(d, \mathsf{ID}_P, t_n)$, the Proxy P composes complete electronic signatures for all clients who sent their requests during the round. An electronic signature of A on message M is of the form

$$[\mathsf{ID}_A, \Pi, \mathrm{SIG}_{\mathsf{sk}_1}(d, \mathsf{ID}_P, t_1), \ldots, \mathrm{SIG}_{\mathsf{sk}_n}(d, \mathsf{ID}_P, t_n)], \quad (3)$$

where Π is *authentication path* – a set of hash values which proves that (M, ID_A) participated in the computation of d (the root of Merkle tree [9]).

5.4 Verifying a Signature

To verify a signature (3) one has to possess authentic copies of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$. Verification consists of the following steps:

(i) The root of the Merkle tree is recomputed by using M and the authentication path Π. If the recomputed root hash d' does not coincide with d then the result of the verification is Invalid. Otherwise, the verification continues with the next step.

(ii) The signatures

$$\mathrm{SIG}_{\mathsf{sk}_1}(d, \mathsf{ID}_P, t_1), \ldots, \mathrm{SIG}_{\mathsf{sk}_n}(d, \mathsf{ID}_P, t_n)$$

are verified using the public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ and the verification procedure VER. If at least t of those signatures are valid, then the result of the verification is Valid. Otherwise, the signature (3) is Invalid.

Note that if all authentication procedures are omitted from the signature creation process, we obtain a *time stamp* [7] instead of electronic signature. Hence, the same service can be used to obtain time-stamps.

Time stamps are needed for long-term preservation of electronic signatures. In case one of the component-signatures of s is broken, or if one of the hash functions (either the function h used to create the hash of the message or the one used to create the Merkle tree) used is suspected of getting broken soon, it is sufficient to take a new and secure hash function H and obtain a time stamp for a message $(s, H(M))$ just as described by Haber and Stornetta [3]. The time stamp is added to the signature in order to preserve its validity.

6 Conclusions

Personal private keys do not necessarily mean higher security. They certainly mean high cost and complexity of electronic signature systems. Elimination of personal private keys could considerably simplify the system, and as we have shown, not at the price of security.

Personal private keys were introduced in order to solve the problems with trust. We claim that no technology – personal private keys or any other measure – can solve problems with trust. Trust relations cannot be imposed by technology. They evolve in natural ways.

In some sense, our society is still in the stage of ”electronic illiteracy” – blind trust to technology is inevitable – and it is hard to see how this situation will change in the near future. Nevertheless, electronic signatures could still be used massively. We have shown that electronic signature service can provide a sufficiently secure solution to electronic signatures.

We presented an electronic signature system that is capable of covering the needs for electronic signatures for the whole Internet community. All the components and primitives we used in our system are well known. The new system is extremely simplified, but still remains as secure as any other electronic signature system known to date.

Acknowledgements

We would like to thank Margus Freudenthal, Jaan Priisalu and Viljar Tulit for fruitful discussions on practical aspects of security. We are also grateful to anonymous referees for their helpful comments.

References

[1] Arne Ansper, Ahto Buldas, Meelis Roos and Jan Willemson. Efficient long-term validation of digital signatures. In *Public Key Cryptography - PKC’2001*, Cheju Island, Korea. Feb. 13-15, 2001. LNCS 1992, 402-415. Springer-Verlag, 2001.

[2] A.Asokan, G.Tsudik, M.Waidner. Server-supported digital signatures. In proceedings of ESORICS’96, Rome, Italy, Sept. 25-27, 1996.

[3] Dave Bayer, Stuart Haber, W. Scott Stornetta. Improving the Efficiency and Reliability of Digital Time-Stamping. In *Sequences II: Methods in Communication, Security, and Computer Science*, eds. R. Capocelli, A. DeSantis, and U. Vaccaro, pp. 329-334. Springer-Verlag, 1993.

[4] W.Diffie and M.E.Hellman. New directions in cryptography. IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp.644-654.

[5] Ford, W. A Public-Key Infrastructure for U.S. Government Unclassified but Sensitive Applications. Produced by Nortel and Bell-Northern Research for NIST, September 1995.

[6] I. Gassko, P. S. Gemmell, and P. MacKenzie. Efficient and fresh certification. In *International Workshop on Practice and Theory in Public Key Cryptography – PKC’2000*, LNCS 1751, pp. 342–353, Melbourne, Australia, 2000. Springer-Verlag, Berlin Germany.

[7] Stuart Haber, W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.

[8] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In Proceed-

ings of the Fifth Annual Conference on Computer and Communications Security, pages 122–131, 1998.

[9] Ralph C. Merkle. Protocols for Public Key Cryptosystems. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, pp. 122-134, 1980.

[10] United States Patent 4,309,569. Ralph Merkle. Method of providing digital signatures. January 5, 1982.

[11] United States Patent 6,097,811. Silvio Micali. Tree-based certificate revocation system. August 1, 2000.

[12] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. CRC Press series on discrete mathematics and its applications. CRC Press, 1996. ISBN 08493 -8523-7.

[13] Chris Pavlovski and Colin Boyd. Efficient Batch Signature Generation using Tree Structures. *International Workshop on Cryptographic Techniques and E-Commerce – CrypTEC'99*, City University of Hong Kong Press, pp.70–77.

[14] `http://www.pgp.com`

[15] Bruce Schneier. Attack Trees: Modeling security threats *Dr. Dobb's Journal*, December 1999.

[16] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology – EUROCRYPT'2000*, LNCS 1807, pp. 207–220. Springer-Verlag, Berlin, 2000.

[17] RFC 3161. Time-Stamp Protocol

[18] John Viega, Gary McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way.* Addison Wesley Professional, 2001. ISBN: 0-201-72152-X. (Chapter 6) `http://www.sdmagazine.com /documents/s=818/sdm0208a/`

[19] Adam Young and Moti Yung. Kleptography: Using Cryptography Against Cryptography. In *Advances in Cryptology – Eurocrypt'97*, LNCS 1233, pp. 62–74. Springer-Verlag. ISBN 3-540-62975-0

Privacy-enhanced credential services

Alex Iliev
sasho@cs.dartmouth.edu

Sean Smith
sws@cs.dartmouth.edu

April 4, 2003

Abstract

The use of credential directories in PKI and authorization systems such as Shibboleth introduces a new privacy risk: an insider at the directory can learn much about otherwise protected interactions by observing who makes queries, and what they ask for. Recent advances in Practical Private Information Retrieval provide promising countermeasures. In this paper, we extend this technology to solve this new privacy problem, and present a design and preliminary prototype for a LDAP-based credential service that can prevent even an insider from learning anything more than the fact a query was made. Our preliminary performance analysis suggests that the complete prototype may be sufficiently robust for academic enterprise settings.

1 Introduction

In this paper, we identify a privacy risk in PKI and other organization-centered authorization systems; we also offer a design (and partial prototype) of a practical solution.

Hippocrates advised physicians to "first, do no harm." We would also like to apply this dictum to the design and deployment of new security infrastructure. While examining whether new technology solves existing security problems, we should also ask: does it create new ones?

Traditional hierarchical PKI (as well as other authorization schemes) can potentially solve many problems regarding interaction within and across organizational boundaries. However, an artifact of this focus on organization is *centralization*. Advanced cryptography and protocols provide security and privacy as an organization's members interact with each other and the world. But making this all work typically requires the organization to maintain a centralized credential server, that offers the right tokens and certificates to the participants when they need them.

Although the cryptography they enable can solve security and privacy problems, the existence of these servers creates new ones: because many interaction with user A require queries to the credential server, it is possible for the credential server to learn a great deal about these interactions, by monitoring who is asking for which credentials. Securing interaction against outside adversaries thus has the unwanted side-effect of enabling attacks on privacy by organizational insiders.

Section 1.1 and Section 1.2 will consider some immediate manifestations of this problem. Section 2 outlines the technology that we and others have helped produce, that may address these issues. Section 3 then explains our design and (not yet complete) prototype that applies this technology to solve this problem. Section 4 and Section 5 examine whether this design will perform well in practice. Section 6 concludes with some directions for future work.

1.1 Certificate Directories

In the standard[1] approach to PKI, hierarchies of CAs and users emerge that mirror organizational hierarchies.

Public-key interaction requires knowing the public key of the other party, and being able to bind this keyholder to some relevant real-world property (such as identity or role). Within a user population served by a single root, a *public key certificate* provides this information; in more complex hierarchies, a multi-step *path* of certificates may be necessary.

To provide these certificates, organizations set up directories (typically via LDAP).

Within a population, if Alice wants to send a secret message to Bob, she needs to obtain Bob's public key. Typically, she asks a directory for this. If Bob receives a message from Alice and wants to verify a signature, he needs her certificate. If Alice did not provide this with the message, then Bob needs to ask the directory; even if Alice did provide it, Bob may wish to check if it's still valid.

Across different populations, parties may need to ask directories for additional certificates to construct trust paths. In more general settings, such as trust decisions based on attribute certificates as well as identity certificates, additional directory queries may be involved.

Consider the privacy implications if the adversary Mallory operates the directory. Alice may be using encryption on her message because she wants to keep this secret. But because she needs Bob's certificate, Mallory knows that Alice is sending a message to Bob. If Bob needs to obtain Alice's certificate (or check whether it's valid), he must ask the directory, so Mallory knows that too. If Alice and Bob take the precautions of using protected channels for their message, Mallory still learns of the interaction via the PKI at the end points. Even if Alice and Bob did not take precautions, the use of PKI lowers the work required for Mallory—rather than monitoring network traffic, she can just log queries to a directory.

1.2 Shibboleth

Shibboleth is a developing system to facilitate user authorization for access to resources in remote sites [9]. The user is assumed to have a *home* site, which can provide information about her. The resources are located at the *target* site. The simplified procedure upon receiving a request for some data, illustrated in Figure 1, is that the SHIRE[2] at the target site establishes an opaque *handle* for a user, after which the SHAR[3] uses this handle to request user attributes from the AA[4] at the home site. The attributes are then used to make an authorization decision.

Because the user handle is opaque to the target site components, they do not learn anything about the user beyond the attributes given by the AA. This is the main reason why Shibboleth claims to be privacy sensitive. What is not covered though, is that the home site can learn a lot about their users' online activities—which target sites they visit, and in some cases even the exact URL's.

For example, say John from Dartmoor College occasionally needs access to `http://webofscience.com/LegalizeIt`, `salon.com/archives/palestine/`, and `pop-music-journal.com/sexpistols/`, and these sites require Shibboleth authorization for users of subscribing institutions[5]. The SHAR at each web site will ask the AA of Dartmoor for some attributes of John, by passing John's opaque session handle (opaque to the sites, not to the AA), and the URL being requested. The URL is

[1] We note that dissent exists.
[2] Shibboleth Indexical Reference Establisher
[3] Shibboleth Attribute Requester
[4] Attribute Authority
[5] Perhaps the sites also offer pricey personal subscriptions.

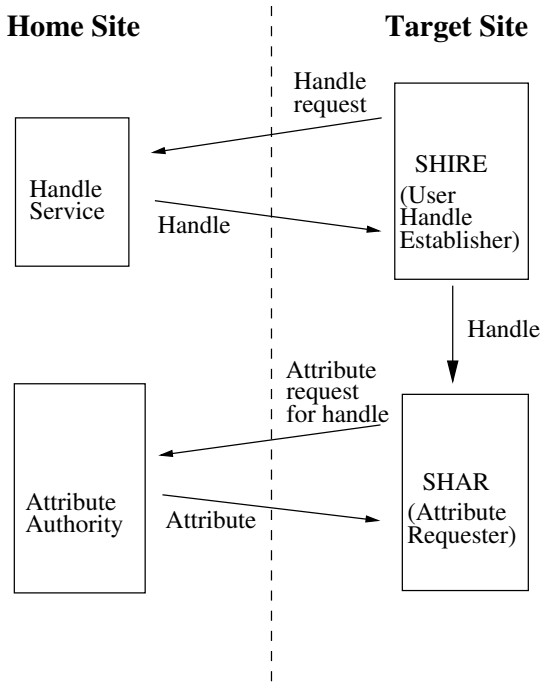

Figure 1: A simplified view of the Shibboleth procedure. On receiving a request, the target site establishes a handle for the user, then requests attributes for that handle, and makes an authorization decision based on the attributes.

passed to the AA so it can decide which attributes to release, based on Attribute Release Policies. The attributes will likely be non-identifying, like confirmation of institution membership, or age, so the sites do not receive any personal information about John. The AA at Dartmoor however does see which sites (including URLs) are asking for attributes for John, and if Mallory at the AA wished to do so, she could log this information.

2 Background

In this section, we examine some relevant technology which can enable a solution to the server privacy problem as described above. We note that work by Stefan Brands offers a much more general solution to privacy concerns in current PKI [4, 5]. Our solutions here are more incremental and have the potential to be deployed until such time as a more general overhaul of PKI is in place.

2.1 Secure Coprocessors

A secure coprocessor is a small general purpose computer armored to be secure against physical attack, such that code running on it has some assurance of running unmolested and unobserved [18]. It also includes mechanisms to prove that some given output came from a genuine instance of some given code running in an untampered coprocessor [12]. The coprocessor is attached to a *host* computer. Since the secure coprocessor we use is implemented as a PCI card, we sometimes refer to a secure coprocessor as a *card*.

2.2 Private Information Retrieval

The problem of *private information retrieval* considers how a user can obtain a particular record from a large set a server offers, without the server learning anything about which record was requested. Simply encrypting the records does not solve the problem; the server can still learn popularity of individual records, correspondence between requests, and (if the server colludes with a user) can learn what any given record decrypts to.

Theoretical computer scientists developed many algorithms (e.g., [7, 6]) through which users, servers, and sometimes other parties could carry out computation and achieve PIR.

2.3 Practical PIR

Smith and Safford [14] then proposed the problem of *practical* PIR: using existing systems, can we provide PIR along the lines of a Web model: the user establishes a shared key, issues a request, waits a short while, then receives the response?

Their solution used COTS[6] secure coprocessors and assumed that a coprocessor can only hold a fixed small number of records internally at one time. Their scheme consists of handling a query to a PIR server by having a secure coprocessor read sequentially through *all* the records in the database (which is kept on the host), keep the correct record internally and return it to the user. The running time of a query is linear in the database size.

Asonov et al [2] then improved the Smith-Safford scheme by decreasing the processing time for a query at the expense of a periodic preprocessing step. We elaborate on this scheme in the next section.

2.4 Asonov's Scheme

The setup for this scheme is that the database consists of N records, numbered from 0 to N-1. They may or may not be originally encrypted, depending on whether the contents need to be kept private. The records are stored on the host, and accessed from the secure coprocessor (the *card*) via a simple API:

- `Record-text read_record(position)` and
- `write_record(Record-text, position)`.

The Record text may be encrypted for reads, and will be encrypted and MAC'ed for writes. An assumption is that at least two records can be stored on the card at a time.

The scheme is divided into two parts:

1. Preprocessing, where the database is shuffled such that the host has no information about the positions of records in the shuffled version.
2. Retrieval, where the card fetches records from the shuffled database.

2.4.1 Preprocessing

Shuffling is done in $O(N^2)$ time, as follows. A uniform random shuffle vector `V` is generated such that record number `V[i]` goes to position `i` of the shuffled database. Then, for each postion `i`, the

[6] Commercial Off The Shelf

card sequentially reads *every* record from the host, keeps record number `V[i]` internally, and writes it out to position `i` of the shuffled database. The host does not know what `V[i]` is, so does not learn anything about the record going into shuffled position `i`.

2.4.2 Retrieval

For the first record retrieved after a shuffle, the card simply gets the record's shuffled position from its shuffle vector `V`, and retrieves that position. For the n^{th} retrieval (call it record R_n) after a shuffle, the card re-fetches *all* $n-1$ records previously retrieved, then fetches and returns R_n. If R_n was in the n-1 already retrieved records, it is kept internally to be returned, and the n^{th} record fetched is a random one not previously touched.

3 Our Extensions and Prototype

We can solve the privacy problem for credential servers by using a PIR server for the information source (e.g, the Shibboleth AA, or a certificate directory). Users could then have assurance that the system operator is not observing their queries [7].

The question, then, is can we build a credential server using PPIR technology and COTS hardware that performs reasonably well?

We decided that the outside interface should be LDAP[9], currently the most popular directory access protocol; we will make the limiting assumption that the querier can only specify one fully named record. This limitation is not unreasonable—asking a directory for the certificates of all Bobs does not seem like an indispensable operation.

We then consider the issues: Section 3.1 discusses extending PPIR to deal with *named* records; Section 3.2 presents a new approach to the shuffling step that decreases the time from $O(N^2)$ to $O(N \log N)$; Section 3.3 discusses our PPIR setup; Section 3.4 discusses our prototype implementation; and Section 3.5 discusses the overall credential server architecture.

3.1 Named Records via Hashing

Real database records are usually named as opposed to numbered. The approach we took to dealing with names in this prototype was to implement the database using hashing with chaining. Thus, hashing a record name yields a bucket number, and inside this bucket is the needed record. We effectively ran the Asonov scheme with numbered buckets. Within a given bucket, a record was retrieved by reading in all the records sequentially and keeping the right one.

The hash function we used is due to Dan Bernstein and was chosen as it was simple, seemed well-recommended, and performed well compared to several others we tried. The most important metric we used was the size of the largest bucket produced. The function is, for a string `str[0..n-1]`:

```
hash(str[0..i]) = hash(str[0..i-1]) * 33 ^ str[i].
```

Some other more complicated name resolution options we considered are

[7] For Shibboleth, further steps needed would be to make the HS[8] privacy-protected too, or a reasonable guess could be made at the identity of a request for attributes which comes soon after a login at the HS.

[9] Lightweight Directory Access Protocol

1. We could hold a data structure of all the record names inside the card, possibly with some index for fast searching, and use this to look up numbers from names. If we assume names to be 20 bytes on average, for 10,000 records this structure would be 200K memory minimum, which could possibly be kept inside the card. Also possible would be to outsource the name resolution to another card. This approach could also be useful with providing substring name matching.

2. We could use a *perfect hash function* [11]. This needs to be constructed especially for the current set of names, but then hashes each name into a unique bucket. Perfect hash functions tend to come with an index whose size is comparable to the name set's size, but since this index would consist of about N integers (sized 2 bytes each), it would certainly be a lot smaller than the full name table.

3.2 Private Shuffling with Permutation Networks

We have planned an alternative and faster shuffling algorithm which we shall sketch out here. It is based on a *Permutation network* - a network of *switches* wired together in a fixed manner and intended to perform a given permutation of its inputs [17]. A switch has two inputs and two outputs, and it may cross the inputs, or pass them on straight. By propagating values along the wires and through the appropriately set switches, any permutation of the input can be produced at the output. An example 4-input network is shown in Figure 2. A permutation network for N inputs can be built recursively as shown in Figure 3. [8] This network clearly consists of $\Theta(N \log N)$ switches. Setting all the switches to achieve a given permutation is possible with a $\Theta(N \log N)$ algorithm [16, 1].

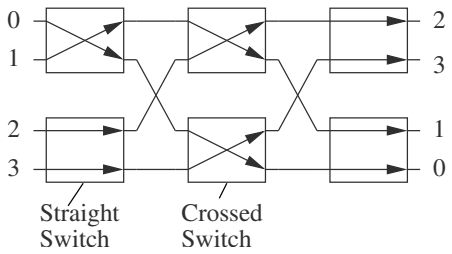

Figure 2: A Permutation network with 4 inputs, performing the permutation $\langle 2, 3, 1, 0 \rangle$

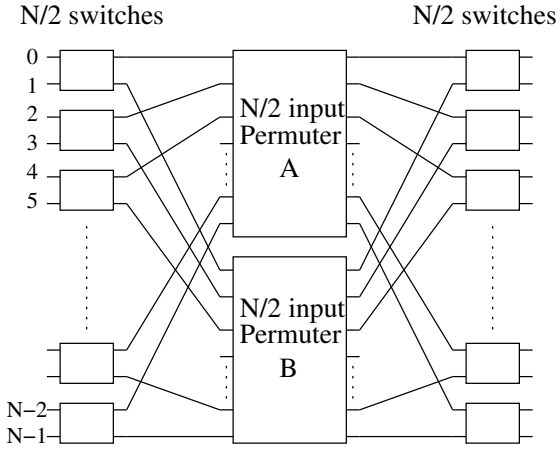

Figure 3: A Permutation network for N inputs built from N switches and 2 networks with N/2 inputs each. Each of the switches on the left have one output wired to permuter A, and one output to B. The switches on the right have their inputs similarly connected. This construction is straighforward but not entirely minimal—N/2 switches can be removed while still enabling any permutation to be executed [17].

Permutation networks have in fact been proposed for use in a related field—mix networks for anonymizing email [1]. The similarity lies in the shared goal of erasing any observable relationship between the inputs and outputs of a shuffle or mix net respectively.

A switch can be interpreted for our shuffling scenario as follows. The coprocessor reads in two records (the inputs), possibly switches their places, and writes them out to the same two positions. The host should be unable to tell if the two records were switched or not. This can be achieved by reencrypting the records with new keys for example.

Finally, a shuffle using such a permutation network would consist of the card internally generating a random permutation, generating a network for its chosen permutation, and then executing all the switches in order (column-major order looks sensible). Generating the network takes $\Theta(N \log N)$ time, as does executing the switches.

3.3 System Setup

Our prototype runs in the IBM 4758 secure coprocessor with Linux [13]. The 4758 is a commercially available device, validated to the highest level of software and physical security scrutiny currently offered—FIPS 140-1 level 4 [15]. It has an Intel 486 processor, 4MB of RAM and 4MB of FLASH memory. It also has cryptographic acceleration hardware. It connects to its host via PCI. Our host runs Debian Linux, with kernel version 2.4.2-2 from Redhat 7.1 as needed by the 4758/Linux device driver.

Linux is an experimental operating system for the 4758, which runs CPQ/++ in production, but Linux has considerable advantages in terms of code portability and ease of development—our prototype is written in C++, making extensive use of its language features and the Standard Template Library, and it runs fine on the card with Linux.

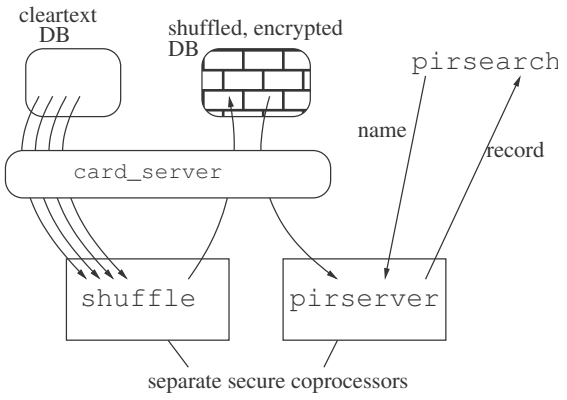

Figure 4: An overview of our PIR prototype. The card programs are `shuffle` which does the shuffling, and `pirserver` which handles retrievals. On the host, `pirsearch` performs a search by passing the name to `pirserver`, and `card_server` handles DB access requests from the card programs. Communication between the card and `card_server` is over SCC sockets, one of the mechanisms provided for 4758 Linux to talk to the outside. We serialize data using the External Data Representation (XDR) library in the RPC package.

3.4 PPIR Implementation

An overview of the components of our PPIR prototype is shown in Figure 4. Our implementation of retrieval with hashing is illustrated in Figure 5. Shuffling in this prototype is a straightforward implementation of the naive algorithm in Section 2.4.1. An implementation of shuffling with permutation networks is in progress. We make our code available at
`http://www.cs.dartmouth.edu/~sasho/privdir/`.

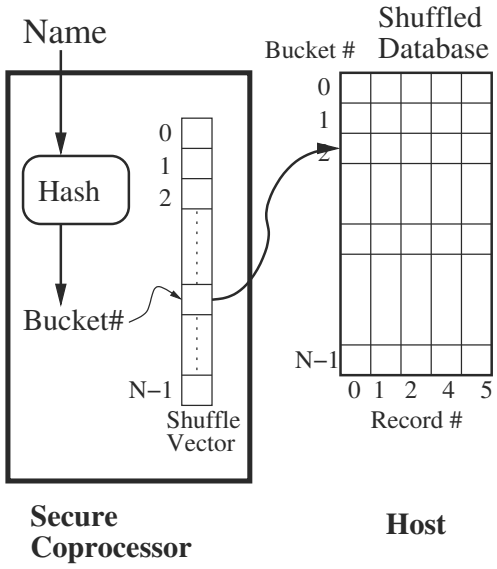

Figure 5: The Retrieval Procedure. Once the card has identified the correct bucket number, it retrieves all the records in that bucket, and keeps the correct one.

3.5 System Architecture

An overview of the whole system is shown in Figure 6. It consists of the PPIR system described in Section 3.4 connected to an OpenLDAP server by means of a *shell backend*. The OpenLDAP server has a variety of ways to access the actual data it provides LDAP access to. One of them is to run a shell command to retrieve or update records.[10] We wrote a perl script to allow the OpenLDAP server to use our `pirsearch` program (see Figure 4). We tested this whole setup by sending LDAP queries from the Sylpheed[11] mail client to our PIR prototype.

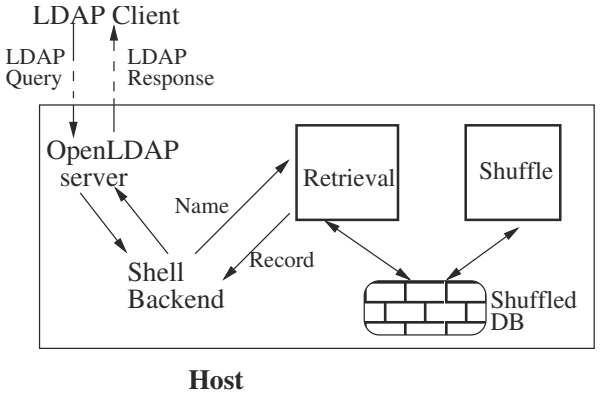

Figure 6: System Architecture. OpenLDAP is used to provide the gateway between our PPIR server and LDAP clients.

[10] There are more operations besides query and update which are idiosyncratic to the LDAP protocol.
[11] `http://sylpheed.good-day.net/`

This setup is a temporary way to achieve the connection to LDAP, and it is clearly not *root-secure*—secure even against an adversary running as root on the host—as queries are in the clear on the host before being handed to `pirsearch`. Our plan for a secured connection all the way from the client to the retrieval coprocessor is shown in Figure 7. It will make use of LDAP over SSL, and use OpenLDAP libraries for parsing of LDAP queries, and construction of LDAP responses.

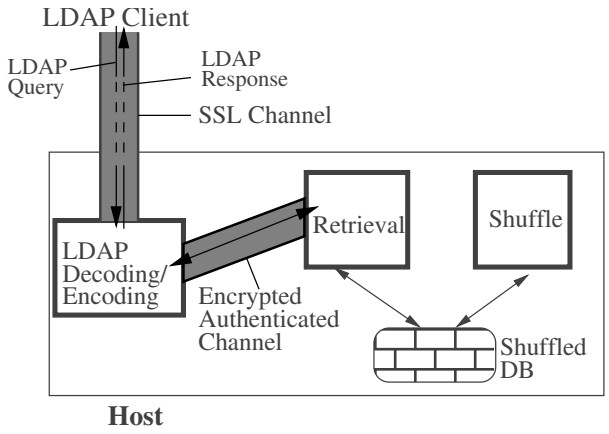

Figure 7: Final System Architecture, using a third coprocessor to handle LDAP and SSL operations. A separate coprocessor will likely be needed for these tasks because of space restrictions inside the coprocessors.

4 Experimental Results

4.1 Performance

One of the main purposes of our prototype was to get a feel for how this PIR scheme performs in practice in the credential server setting, and what may need to be improved to make it really usable. The most interesting source of performance numbers was from the shuffling step, which is shown in Table 1. The database size N was 1000, and hashing had resulted in every bucket holding 5 records. The times shown are for one pass of the shuffle algorithm, where all the buckets are read by the card in order to keep one of them to write to a given position in the shuffled database.

Record size (bytes)	**Time to Read 1000 Records (sec)**
115	18
530	24
1345	34

Table 1: Read time during shuffling vs. Record size.

The times for a run of 300 retrievals is shown in Figure 8.

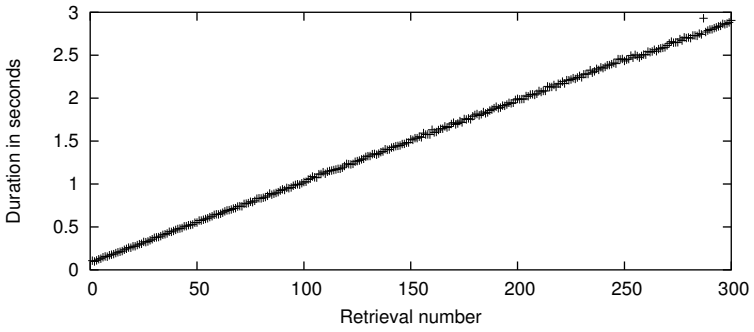

Figure 8: Times for each of 300 sequential retrievals of random records in the database.

4.2 Hashing

The main price of using a fixed hash function is that collisions inevitably occur, and in this case they are particularly damaging—since all buckets need to look the same to the host, they must all hold the same number of records. In our case the largest bucket received 5 records from the hash function, so we had to pad *all* the buckets to 5 records, thus having 4N dummy records—4000 for our test database.

5 Analysis

5.1 Related Work

One of the mechanisms used in Oblivious RAMs [10] relies on private shuffling of memory addresses, which is implemented using a sorting network—bitonic sorting. This is a network of *comparators* (analogous to switches in the permutation network), whose structure depends only on the problem size. It consists of $\Theta(N \log^2 N)$ comparators, thus being a log factor larger than a permutation network of equal input size.

Another approach to private database shuffling comes from Asonov [3]. It consists of splitting each record into p pieces to create p "database slices", each of which has one piece of each record. These slices are then shuffled, and reassembled. This approach can reduce the number of communications between the host and coprocessor from $\Theta(N^2)$ to $\Theta(N\sqrt{N})$. Asonov is in the process of publishing further refinements and experiments [Personal Communication].

5.2 Prototype's Shuffling

Several observations arise from the figures in Table 1. Firstly, a linear relationship between the record size (s) and read time for 1000 records (t) is $t \approx 16 + \frac{s}{75}$. This confirms that there are considerable overhead costs to the host-card communication, and that maximizing the amounts of data transferred at a time is desirable.

Secondly, the whole shuffle, which consists of N scans through the whole database, would in this case take $18,000$ seconds, or 5 hours, for the smallest record size. This brings into question the real usability of the scheme with naive shuffling for larger but quite realistic database sizes like $10,000$ records—the prediction in that case is 500 hours, or almost 3 weeks—shuffle that!

5.3 Retrievals

From Figure 8, around the 300^{th} retrieval the time reaches 3 seconds, which is a reasonable ceiling on the duration of a query, so we may want to set up the system so that a single shuffled database is used for about 300 queries.

5.4 Permutation Network Shuffling

A more detailed analysis of the expected running time of a permutation network is as follows. From the recursive construction of Figure 3, we can determine that a network with $N = 2^n$ inputs will have $2n - 1$ columns of $N/2$ switches each. If the switches are executed in column major order, each column will consist of N record reads, and N record writes. Now if we approximate a running time for one column as twice the time for reading N records, this should amount to about 36 seconds for our N=1000 database. With $2\lceil \log N \rceil - 1 = 19$ columns, the switch execution stage should last about 12 minutes, and it should dominate the shuffle time, so 20 minutes is a conservative estimate of the total shuffling time.

For a larger database with say $N = 10,000$, each column of the permuter should take about 6 minutes, and there will be $14 \times 2 - 1 = 27$ columns, for a total of 160 minutes spent permuting, so perhaps 3 hours for the whole shuffle.

Our initial experiments with an implementation of a permutation network suggest that these estimates are too optimistic, by constant factors, mainly due to the extra decryption and encryption involved. In particular, each switch of the network requires a decryption followed by a re-encryption (with a different key or IV) of the two records being switched. However the cost of these symmetric crypto operations will be much reduced when we begin to use the crypto hardware of the 4758 secure coprocessor—currently the TDES operations are done in software, and TDES stands to benefit a lot from special-purpose hardware, like that in the 4758.

5.5 Name Resolution

As we wrote above, hashing required us to introduce 4N dummy records into the hashed database. This brings about a factor of 5 increase in the running time of most procedures in the system. If we use a perfect hash function, each record name would hash to a unique record number, and there would be no need to carry the deadweight of dummy records. Thus our current running times for shuffling *and* retrieval could be reduced by up to a factor of 5. The cost would be the complexity of computing the perfect hash function when the name set changes (which should be infrequent).

In Table 2 we list a summary of our measured and predicted shuffle run times.

5.6 Consolidation and Feasibility

If shuffling with a permutation network is combined with a name lookup method with less overhead than hashing with chaining, the shuffle time for a 10,000 record database may be lowered from 3 to about one hour. In addition, reducing the hashing overhead should reduce retrieval times, as no dummy records would need to be fetched. The largest number of retrievals off one shuffled database should go up from 300 to perhaps 1000. If we have C coprocessors shuffling databases in parallel, they can produce C shuffled databases an hour, which give $1000C$ queries. The system would be able to deal with $1000C/3600 \approx C/4$ queries per second, answering each query in less than 3 seconds.

Interesting to note here is that there is no point in having a big collection of shufflers, as the

Scheme / DB Size	A	B	C
1,000	5 hrs	20 mins	6 mins
10,000	3 weeks	3 hrs	1 hr

Schemes:
A—Naive Shuffling, Hashing with chaining
B—Permutation Network Shuffling, Hashing with chaining
C—Permutation Network Shuffling, Low Overhead Name Resolution

Table 2: All Shuffling Times in One Place. The 5 hours figure was measured. 3 weeks is a prediction of our prototype's time on larger input. The other numbers are predictions of schemes we have analyzed in Section 5 and will be implementing.

retrieval coprocessor will not be able to deal with much more than one query in 3 seconds on average—if the query rate goes higher when retrievals are taking close to 3 seconds, a queue will quickly build and the response time will be really bad. Thus, the shuffling will no longer be such a bottleneck, and parallelism (for sustaining a higher query rate) can be achieved by duplicating shufflers as well as retrievers.

6 Future Work and Conclusions

We currently have a functioning prototype of a private credential directory accessible over LDAP. It has some fairly serious performance shortcomings, which we are currently addressing. In particular we are implementing a faster database shuffling algorithm, and faster resolution of record names. We strongly believe, on the basis of our current measurements and the details of our proposed changes, that these changes will yield usable performance. We will connect this next version of our prototype to the certificate directory currently being rolled into operation for Dartmouth's new campus PKI, and so get real usage experience for the system. Dartmouth also plans to deploy a Shibboleth prototype, so we will have a testbed for a private Shibboleth AA. We believe that this system has realistic potential to address the problems of server privacy exposed at the beginning of the paper.

Acknowledgments The authors have received support from the Mellon Foundation, the NSF, AT&T/Internet2, and the U.S. Department of Justice (contract 2000-DT-CX-K001). The views and conclusions do not necessarily reflect those of the sponsors.

References

[1] Masayuki Abe and Fumitaka Hoshino. Remarks on mix-network based on permutation networks. In Kwangjo Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 317–334. Springer, 2001.

[2] Dmitri Asonov and Johann-Christoph Freytag. Almost optimal private information retrieval. In *Privacy Enhancing Technologies*, LNCS, San Francisco, 2002. Springer.

[3] Dmitri Asonov and Johann-Christoph Freytag. Private information retrieval, optimal for users and secure coprocessors. Technical Report HUB-IB-159, Humboldt University, 10099 Berlin, Germany, 2002.

[4] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy.* The MIT Press, August 2000.

[5] Stefan Brands. A technical overview of digital credentials. At `http://www.credentica.com/technology/technology.html`, Feb 2002.

[6] C. Cachlin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, LNCS. Springer-Verlag, 1999.

[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45:965–982, 1998.

[8] Thomas Cormen, Charles Leiserson, Ronald Rivest, and Cliff Stein. *Introduction to Algorithms*, chapter 27. McGraw-Hill, second edition, 2001. Problem 27-3 on permutation networks.

[9] Marlena Erdos and Scott Cantor. Shibboleth architecture. Available from `http://shibboleth.internet2.edu/`, May 2002. Version 5.

[10] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[11] Bob Jenkins. Minimal perfect hashing. `http://burtleburtle.net/bob/hash/perfect.html`, 2003.

[12] Sean Smith. Outbound authentication for programmable secure coprocessors. In *7th European Symposium on Research in Computer Science*, Oct 2002.

[13] Sean W. Smith and Steve Weingart. Building a high-performance, programmable secure coprocessor. *Computer Networks*, 31:831–860, 1999.

[14] S.W. Smith and D. Safford. Practical server privacy using secure coprocessors. *IBM Systems Journal*, 40(3), 2001. (Special Issue on End-to-End Security).

[15] National Institute Of Standards and Technology. Security requirements for cryptographic modules. `http://csrc.nist.gov/publications/fips/fips140-1/fips1401.htm`, Jan 1994. FIPS PUB 140-1.

[16] Eli Upfal. A permutation network. `http://www.cs.brown.edu/courses/cs253/slide/class2.ps`, 2000. Course Lecture Notes.

[17] Abraham Waksman. A permutation network. *Journal of the ACM*, 15(1):159–163, Jan 1968.

[18] Bennet S. Yee. *Using Secure Coprocessors.* PhD thesis, Carnegie Mellon University, 1994.

On the usefulness of proof-of-possession

N. Asokan, Valtteri Niemi, Pekka Laitinen
Nokia Research Center, Finland
{n.asokan,valtteri.niemi,pekka.laitinen}@nokia.com

Abstract

Public key infrastructure standards assert that proof-of-possession of private key is an essential requirement during the enrollment process. Even though the justifications for this requirement seem to be well-known within the PKI community, they do not appear to be documented anywhere. In this paper, we document and examine potential rationales for proof-of-possession and discuss their merits. We conclude that if protocols and applications are designed "properly", proof-of-possession does not add any security. However, the world is not perfect. Many existing applications and protocols are in fact not properly designed. Proof-of-possession is a useful safety precaution for the users of such applications and protocols. But there is no simple automated way for a relying party application to check whether proof-of-possession was done during enrollment. Therefore, we argue that designers of public key protocols *must not* assume that CAs require proof-of-possession during enrollment.

1 What is proof-of-possession?

In a public key infrastructure (PKI), the process of submitting a certificate request to a certification authority (CA) or a registration authority (RA) is known as *enrollment*. After enrollment, the CA will issue a certificate to the enrolled public key. During enrollment, the end entity that submits the public key may be required to prove that it knows the corresponding private key and that it controls the use of this private key. This is commonly referred to as the *proof of possession* (PoP).

Every PKI standard asserts that PoP is essential. However, none of them explicitly lays out the threats that are intended to be addressed by PoP. The rationales and implications of PoP have been discussed in standards meetings and mailing lists [6, 7]. Yet, there does not appear to be any easily available or commonly known papers or articles that document these issues. It appears to be yet another case of undocumented folklore within the communities involved.

In this paper, we examine the potential rationales for PoP and discuss their merits. Our goal is to clarify the answers to the following questions:

- Should the designer of a new PKI require PoP during enrollment?
- Does the designer of a new public key based application or protocol benefit from having PoP done during enrollment?

Our work was motivated by the on-going work in the 3^{rd} generation partnership project (3GPP) for designing support for subscriber certificates [10]. 3GPP security group considered various ways of securing the enrollment messages. One of them was to use the cellular signaling channel which provides mutual authentication and integrity-protection. This channel is severely bandwidth-limited. Thus it was necessary to check that every bit sent through this channel is really essential. This prompted us to start investigating the conditions under which PoP is indeed indispensable.

In the rest of the paper, we use the term "PoP" as it is customarily used, without any additional qualification. The precise characterization is "proof-of-possession of private key during enrollment." Public key protocols often involve other types of proofs of possession: for example, every time a relying party verifies a signature, it is proof that the signer possessed the signing key; "plaintext-aware" encryption schemes [8] include a proof that an entity claiming to have produced a ciphertext actually knew (hence possessed) the plaintext. Such proofs of possession are not the subject of this paper.

In Section 2 we begin by defining the ways in which a private key of an asymmetric cryptosystem is used. In Section 3 we describe how the public key enrollment process is secured in PKIs. In Section 4 we describe attacks that are not intended to be prevented by PoP, and in Section 5 we describe potential attacks that *can be* prevented by PoP. In Section 6 we consider scenarios where mandating PoP is not advisable. In Section 7 we briefly describe the degree

to which existing PKI specifications require PoP. In Section 8 we consider other possible rationales for requiring PoP. Finally, in Section 9 we summarize our findings.

2 Use of a private key

There are three primary ways in which a private key of an asymmetric key pair is used:

- **commitment**: By signing a message with the private key, the purported controller of the private key commits himself to the signed message. Enabling non-repudiation is an example application of this kind.
- **claim**: By signing a message with the private key, or by decrypting a ciphertext (and demonstrating knowledge of the plaintext), the purported controller of the private key can stake a claim for a benefit or ownership. For example, if the signature is on a plaintext message m and the signature is accompanied by an identity certificate binding an entity X to the signature verification key, the claimed fact may be "X is certified to have control of the signing key corresponding to this signature verification key, and therefore message m signed by this key is known to X and uttered by X." In other words, message authentication is an example application of this kind.
- **encryption**: Of course, a keypair can also be used just for encryption. This has similar properties like the "claim" use case.

In standard X.509v3 certificates, it is possible to use the `keyUsage` or `extKeyUsage` parameters to indicate the types of uses (e.g., "non-repudiation") to which the public key certificate is limited.

3 Security of enrollment

To enroll in a PKI an end entity will send a certificate request to the CA/RA. A certificate request contains a claimed public key, and any additional information, such as a claimed identity and attributes, and additional certificates in support of the request. Attribute certificates do not contain a public key: instead they bind a name to a set of attributes. When requesting an attribute certificate, the request may not contain public key. We do not consider this case further because use of such certificates should be accompanied by an identity certificate that binds a public key to the said name. Therefore PoP does not appear to be relevant when requesting attribute certificates.

The enrollment process must be secured so that the CA/RA can associate the submitted public key with the correct authorizations allowed for the submitter. An example of such an authorization is the right to be bound to the identity of a specific end entity. Typically this is done by distributing a shared one-time key or password ahead of time, for example, by mailing scratch cards containing initial PIN codes to potential users of PKI. Such a key or password is known as an initial authentication key. The initial authentication key will be used to authenticate and integrity-protect messages in the enrollment process. There are also other ways of securing enrollment, for example by using any existing or derivable security association between the end entity and the CA/RA, e.g., as done in the PIC protocol [11]. This approach is applicable when we need to bootstrap a PKI from an existing infrastructure [10]. The purpose of this authentication is to allow the CA/RA to determine whether and how to approve the certificate request.

4 Examples of attacks not prevented by PoP

The certificate request is authenticated as mentioned in Section 3. If the cryptographic transforms used are inappropriate, it will allow an attacker to compromise the security of the enrollment process: an attacker on the network can change the contents of the request or fake a new request without being detected. We can call this the"weak password attack." PoP does not help against this attack. PoP is not intended to be a replacement for having to design appropriate mechanisms for the security of enrollment.

If PoP is required as part of certificate request, it assures the CA/RA that the requestor had access to the private key corresponding to the public key on which the certificate is requested.

Suppose the private key is used for commitments only, such as for non-repudiation. If PoP is not done during the certificate request process, a legitimate end entity Alice can indeed obtain a certificate binding her name to the public key of another end entity Bob. But this only leaves Alice, the supposed attacker, liable for commitments made by Bob, who controls the private key and can make signatures. In other words, this is not a protocol attack. (However, as described in Section 5.2.2, there are protocol attacks related to claim type uses.) In fact, Alice may use a slightly different variation, as we discuss in Section 6, to delegate authority to Bob.

Note that Alice still cannot send an arbitrary message *m* and claim that it belongs to and/or was sent by Bob. The security of the enrollment process (Section 3) is intended to prevent Alice from being able to send arbitrary signed message and fool the attacker into thinking that they came from Bob. See Section 5.1 for more discussion.

5 Attacks prevented by PoP

5.1 Replacing public key in enrollment request sent by victim

5.1.1 Attack description and assumptions

In a certificate request sent by Bob, the attacker replaces Bob's public key by Alice's public key. The effectiveness of this attack is contingent on the following assumptions being valid:

1. *Security of enrollment is weak*: Normally, this attack is prevented by the mechanism for securing enrollment. One aspect of securing enrollment is the protection of the certificate request message itself. As discussed in Section 4, PoP is not relevant from this aspect. A second aspect of securing enrollment is the access control on the devices used by the victim Bob. The access control mechanisms on Bob's device may not be robust enough to prevent the attacker from (i) changing the contents of the local public key repository without being detected, or (ii) accessing the secret keys used to ensure the security of enrollment. We can call this the "Trojan attack."
2. *Although the attacker is able to insert a public key into Bob's device, it is unable to insert a private key or intercept the communication path between the signing algorithm and the calling application*: if this were not true, the attacker could insert a whole key pair into Bob's device. Thereafter PoP is of no use. Note that Alice need not suffer any harm by inserting a private key: it can be the private key of a keypair that the attacker generated solely for this attack. Alternately, even if the attacker cannot insert the private key in the standard place where private keys are stored on Bob's device (e.g., a smart-card), it is enough if the attacker intercepts private key operation requests or replaces the responses.

5.1.2 Type of uses

In commitment type uses, if the public key is a signature verification key, then Alice can make commitments (signatures), have them supported by the certificate, and leave Bob liable for them.

In claim type uses, if the public key is an encryption key, then Alice can send it to a peer Carol and trigger Carol into encrypting, with this key, some confidential data intended for Bob.

In either case any protection provided by PoP is subject to both the assumptions listed in Section 5.1.1 being true. The assumptions are not very realistic because they require the access control on the victim's device to be faulty in a very specific manner. If the victim's device does not have adequate access control, PoP does not help because the attacker can make sure that the certificate request has the correct signature as described above in assumption 2. If the victim's device is indeed secure, then the Trojan attack should not succeed, and assumption 2 would not hold.

5.2 Using victim's public key in own enrollment request

5.2.1 Attack description and assumptions

In a certificate request sent by Alice, she can use the public key of another legitimate end entity Bob. As explained in the next section, the basic attack does not appear to rely on any strong assumption other than badly designed applications or application protocols.

5.2.2 Type of uses

Without PoP, CA may issue Alice a certificate containing Bob's public key. Depending on the type of use, this threat can be turned into concrete attacks as follows.

In claim type uses

1. if the public key is a signature verification key, then Alice could falsely claim ownership of messages that were signed by Bob. Note that if the application protocol would bind some identification of the sender within the signed message, and the verifying party's application would compare this identification with what is in the certificate used to verify the signature, then this attack would not work. We can call this the "sloppy application protocol attack."
2. if the public key is a signature verification key, then Alice can mislead a peer Carol into revealing to her some private data that Carol intended for Bob. This works as follows [12]:

   Alice obtains a certificate for Bob's public signature verification key. Then Alice waits until Bob sends a signed, encrypted message and Bob's certificate to Carol. Alice intercepts these. Alice does not know what the contents of the message were. She then forwards this intercepted message to Carol along with the certificate she obtained earlier. Carol concludes that the message actually came from Alice because all cryptographic checks

succeed. This might cause Carol to send some private information to Alice in the clear (e.g., something about the contents of the message she just received).

Again, this is a variation of the "sloppy application protocol attack:" if the messaging protocol required that the sender's identity must be included in the signed text, Carol's software would notice that the certificate and the signed data do not match.

In commitment type uses, there does not appear to be any effective protocol attack as a result of this flaw. However, there may be a software program like an e-mail client that does use public keys in both ways (commitment and claim). If the application is not properly written, it may either not pay attention to the limited use of a certificate (such as `keyUsage` set to "non-repudiation"), or if user is careless, she might not notice that the signature does not authenticate the sender. In either case she may incorrectly conclude the sender is authenticated and act based on this conclusion. For example, in example 2), suppose the certificate has `keyUsage` set to "non-repudiation", but the e-mail client of Carol does not indicate this unambiguously. So Carol may be misled into assuming that the signature and certificate authenticate the sender, and make the same conclusions as though the certificate is also intended for "claim" type of uses. We can call this the "sloppy application attack."

The attack against claim type use assumes that application software and application protocol designers may have made some basic mistakes. The security of the users of such applications can benefit from mandatory PoP. The attack against commitment type use can also be avoided if

1. a keypair has only one type of use, e.g., either authentication or non-repudiation[1], and
2. the software application of the relying party handles `keyUsage/extKeyUsage` restrictions correctly.

If it is difficult to ensure either of the above (e.g., it may be difficult to mandate the former, and unrealistic to expect the latter) then PoP can help limit the damage. However, attempting to provide protection against sloppy application designers is ultimately a doomed exercise.

6 Does PoP do any harm?

We saw that PoP could potentially offer some protection for users of badly designed applications and protocols. If PoP has no harmful consequences, requiring PoP is a prudent safety precaution. So, the logical next question is whether PoP does any harm. We consider the following factors.

- **Bandwidth**: PoP means that the requestor has to perform a private key operation. This in turn implies that a large message (e,g., signature or encryption) needs to be sent between the requestor and the CA/RA. If the certificate request channel is bandwidth constrained, size of messages becomes a factor. As we mentioned in Section 1, this was the context in which we started to investigate whether PoP is indeed indispensable.
- **Latency**: To prevent against replay attacks the PoP protocol must ensure freshness. As usual, this can be done using timestamps, which requires synchronized clocks. A better alternative is where the CA/RA sends a challenge nonce. But this means that the certificate request procedure will typically contain an extra request/response pair.
- **Novel applications**: It appears that the need for PoP arose from the "traditional" PKI scenarios where the certificates issued are identity certificates. In such cases, it is of course quite logical to try to prevent two persons from attempting to get certificates for the same public key. However, as limited scope PKIs are becoming more realistic and more prevalent, there may be new uses that are prevented or made harder by mandating PoP. For example, suppose certificates are used for authorizing payments from a bank account. A use case may be for Alice to obtain a certificate for Bob's public key as a surprise gift voucher (so that Bob is allowed to spend a certain amount of money which will be paid from Alice's account). If PoP is required, then obtaining such a certificate will require Bob's involvement, which will eliminate the surprise factor, and hence the point of this use case! Note that the certificates used in this case would not bind Alice's identity to a public key. Instead, they would bind some authorizations to a public key. In other words, they would be authorization certificates [4], rather than identity certificates.

  Although mandatory PoP will prevent a solution to this use case using standard certificates, it is of course possible to design solutions using other types of constructs, e.g., by defining some form of delegation tokens.

Enrollment is a relatively rare occurrence. In typical scenarios, it does not have any real-time requirements. Therefore, we conclude that in general, bandwidth and latency are not critical factors. While mandatory PoP may in fact preclude some class of applications using certificates, there are other ways of designing these applications. Thus, we conclude that PoP does no harm.

[1] Achieving non-repudiation requires a lot more than digital signatures; but discussion on the usefulness of "non-repudiation" as a `keyUsage` is beyond the scope of this paper.

7 The place of PoP in current PKI specifications

The PKCS #10 specification [9], designed by RSA Laboratories, states the following:

> "The signature on the certification request prevents an entity from requesting a certificate with another party's public key. Such an attack would give the entity the minor ability to pretend to be the originator of any message signed by the other party. This attack is significant only if the entity does not know the message being signed and the signed part of the message does not identify the signer. The entity would still not be able to decrypt messages intended for the other party, of course." (Section 3, Note 2 of [9]).

The threat described here is the sloppy application protocol attack we discussed in Section 5.2.2. Surprisingly, the wording of the above text, quoted from PKCS #10, suggests that the impact of this attack is minor. Nevertheless PKCS #10 mandates the use of PoP.

The Wireless PKI specification [13], designed by the precursor to the Open Mobile Alliance, states that PoP is necessary "in order to avoid certain substitution attacks" (Section 4.1) but it does not describe the attacks themselves.

The IETF CMP specification [2] states that PoP is necessary "in order to prevent certain attacks and to allow a CA/RA to properly check the validity of the binding between an end entity and a key pair" (Section 2.3 of [2]). It does not describe what these attacks may be or whether they are limited to the case of identity certificates only.

The e-mail archives of the IETF PKIX working group [6, 7] contain records of extensive discussions on whether PoP should be mandatory. The participants recognized both the assumptions under which PoP is useful (e.g., sloppy application protocols) and the limitations that PoP may impose (e.g., precluding novel applications).

As a result of these discussions, the working group appears to have chosen to require PoP, but *not to mandate PoP to be part of the certificate request protocol itself*. The current CMP specification contains the following explanation:

> "... it is REQUIRED that CAs/RAs MUST enforce POP by some means because there are currently many non-PKIX operational protocols in use (various electronic mail protocols are one example) that do not explicitly check the binding between the end entity and the private key. Until operational protocols that do verify the binding (for signature, encryption, and key agreement key pairs) exist, and are ubiquitous, this binding can only be assumed to have been verified by the CA/RA."[2]

8 Rationales for justifying PoP

One of the primary reasons for requiring PoP seems to be to minimize potential damage due to

- badly designed application-level protocols, or
- badly designed end entity application software, or
- carelessness of an end entity user.

The attacks described in Section 5.2.2 may become possible due to one of the factors listed above. PoP could potentially reduce the likelihood of the resulting attacks. CA operators therefore may view PoP as a way of protecting them from liability arising from damage due to these factors.

However, currently there is no easy *automated* way for a relying party application to check if PoP was done during enrollment. This is because there is no standard place in a certificate for the CA to indicate this. In order to benefit from PoP, relying parties must make sure that they never use certificates issued by CAs that do not require PoP. For example, users may have to examine that certification practice statements (CPSs) of CAs before accepting certificates issued by them. Needless to say, this is not a pragmatic solution.

Therefore, any good application developer has to assume that PoP was not done at the time of enrollment. In particular, an application developer must

- explicitly include all necessary identification and context information in the parts of application protocol messages that are cryptographically protected, (for example, a PKI-enabled e-mail client could include the name of the sender in the signed text; signature verification should fail if this address does not match the name in the certificate used to verify the signature.)
- require the use of different keys for different purposes, and
- consistently and correctly identify the purpose of a given key (e.g., by precisely defining the semantics of the `keyUsage` attributes) so that it is not used for a different purpose.

Such rules are part of the general guidelines for well-designed cryptographic protocols discussed elsewhere [1, 3] and are applicable in this context. The first rule was also repeatedly pointed out in the IETF PKIX mailing list discussions [6, 7].

9 Conclusions

Several standards allude to unspecified attacks in justifying why PoP is needed. We discussed potential threats and discussed how PoP can help reduce their impact. A well designed application protocol does not need PoP. However, many existing protocols and applications are not well designed in this sense. PoP is useful as a safeguard for users of such applications and protocols.

Mandating PoP has some drawbacks. It will preclude the use of standard certificates to achieve one class of use cases where Alice is allowed to delegate authority to Bob by obtaining a certificate for Bob's public key without Bob's involvement. Also, if the communication channel used for enrollment is resource constrained, it is necessary to check if PoP is really needed for the application under consideration. But none of these drawbacks is substantial.

It is becoming increasingly clear that the successful uses of PKI tend to be for specific applications [5]. Designers of application-specific PKIs can and should check if PoP is really needed for the applications of interest to them.

Thus we conclude that by and large, requiring PoP during enrollment is a useful safety precaution because of the shortcomings in applications that are already widely deployed. Designers of new PKIs should require it, especially if there is any likelihood that their PKI will be used with legacy applications.

However, as there is no simple automated way for a relying party application to check whether PoP was done during enrollment, we argue that designers of new security protocols and applications *must not* assume that CAs require PoP during enrollment. They must follow the well known rules of secure protocol design referred to in Section 8.

10 Acknowledgments

We thank the anonymous referees, Antti Vähä-Sipilä, Jukka Virtanen, Kaisa Nyberg, Michael Waidner, Pasi Eronen, Philip Ginzboorg, Olli Immonen, and the members of the 3GPP SA3 working group for their valuable feedback on previous versions of this paper.

References

[1] Martín Abadi and Roget Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[2] C. Adams and S. Farrel. Internet X.509 public key infrastructure: Certificate management protocols. Internet Engineering Task Force, RFC 2510, March 1999.

[3] Ross Anderson and Roger Needham. Robustness principles for public key protocols. In Don Coppersmith, editor, *Advances in Cryptology; CRYPTO '95*, number 963 in Lecture Notes in Computer Science, pages 236–247. Springer-Verlag, 1995.

[4] C. Ellison et al. SPKI Certificate Theory. Internet Engineering Task Force, RFC 2693, September 1999.

[5] Peter Gutmann. PKI: It's Not Dead, Just Resting. *IEEE Computer*, 35(8):41–49, Auguist 2002.

[6] IETF PKIX Mailing list discussions. IETF PKIX mailing list archive, February 1997. `http://www.imc.org/ietf-pkix/old-archive-97/thrd3.html#00081`.

[7] IETF PKIX Mailing list discussions. IETF PKIX mailing list archive, October 1997. `http://www.imc.org/ietf-pkix/old-archive-97/threads.html#01062`.

[8] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Avaliable from `http://www.cacr.math.uwaterloo.ca/hac`.

[9] RSA Laboratories. PKCS #10 v1.7: Certification Request Syntax Standard. RSA Laboratories, May 2000. Also published as IETF RFC 2986.

[10] 3GPP security group. Support for subscriber certificates. Third generation partnership project (3GPP), Security working group work item description, February 2002. Available from `http://www.3gpp.org/ftp/tsg_sa/WG3_Security/TSGS3_22_Bristol/Docs/PDF/S3-020162.pdf`.

[11] Y. Sheffer, H. Krawczyk, and Bernard Aboba. PIC, A Pre-IKE Credential Provisioning Protocol, October 2002. IETF *ipsra* working group draft `draft-ietf-ipsra-pic-06.txt`.

[12] Michael Waidner. Personal communication, August 2002.

[13] WAP Forum. Wireless application protocol, public key infrastrcutre definition. WAP forum, WAP-217-WPKI, April 2001.

Keyjacking: Risks of the Current Client-side Infrastructure

John Marchesini, S.W. Smith, Meiyuan Zhao
Department of Computer Science
Dartmouth College

{carlo,sws,zhaom}@cs.dartmouth.edu

April 21, 2003

Abstract

In theory, PKI can provide a flexible and strong way to authenticate users in distributed information systems. In practice, much is being invested in realizing this vision via tools such as client-side SSL and browser-based keystores. Exploring this vision, we demonstrate that browsers will use personal certificates to authenticate requests that the person neither knew of nor approved (in some scenarios, direct migration from password-based systems to client-side SSL makes things worse). We also demonstrate the easy permeability of these keystores, including new attacks on medium and high-security IE/XP keys. We suggest some short-term countermeasures. However, against this background, it is not clear that the current client-side infrastructure can achieve the PKI vision. A fundamental rethinking of the trust, usage, and storage model might result in more effective tools for building a PKI.

1 Introduction

Because public-key cryptography can enable secure information exchange between parties that do not share secrets a priori, PKI has long promised the vision of enabling secure information services in large, distributed populations.

In the last decade, the Web has become the dominant paradigm for electronic access to information services. The *Secure Sockets Layer* is the dominant paradigm for securing Web interaction. For a long time, SSL with *server-side authentication*—where, during the handshake, the server presents a public-key certificate and demonstrates knowledge of the corresponding private key—was perhaps the most accessible use of PKI in the lives of ordinary users.

However, in the full vision of PKI, all users have key pairs—not just the server operators. Within the SSL specification, a server can request *client-side authentication*—where, during the handshake, the client also presents a public-key certificate and demonstrates knowledge of the corresponding private key. The server can then use this information for identification, authentication, and access control on the services it provides to this client.

An emerging client-side PKI exploits the natural synergy between these two scenarios. Because the Web is the way we do business and client-side SSL permits servers to authenticate clients:

- modern browsers[1] now include personal keystores, for a user's key pairs;
- enterprises (and other distributed populations) are arranging for users to obtain certified key pairs to live in these keystores;
- providers of Web information services are starting to use client-side SSL as a better alternative than passwords or to authenticate users;
- and even non-Web applications may typically expect to find and use the key pair resident in the browser keystore.

[1] Admittedly, due to Microsoft's contention that the browser is part of the operating system, one might argue that the Internet Explorer (IE) keystore on Windows is really part of Windows.

In previous work, we have examined the effectiveness of server-side SSL [33] and of digital signatures on documents [15]. In this paper, we examine the question: *does this client-side PKI work*?

- When browser-based keystores are used in contemporary desktop environments, is it reasonable for the user at the client to assume that his private key is used only to authenticate services he was aware of, and intended?
- Is it reasonable for the user at the server to assume that, if a request is authenticated via client-side SSL, that client was aware of and approved that request?

Our Agenda We wish to stress that we believe that PKI is a much better way than the alternatives to carry out authentication and authorization in distributed, multi-organizational settings, for many reasons:

- PKI does not require shared secrets.
- PKI does not require a previously-established direct trust relationship between the two parties.
- PKI permits many parties to make assertions.
- PKI permits non-repudiation of assertions—Bob can prove to Cathy that Alice authorized this request to Bob.

In particular, we are *not* advocating password schemes.

However, rolling out client-side PKI and migrating existing information services to use it requires considerable resources and effort. Weaknesses in the underlying technology risk undermining this effort. We provide a critical examination of the current client-side PKI approach precisely because we want the PKI vision to succeed.

This Paper First, we lay out the background. Section 2 introduces how Web services work; Section 3 discusses (pre-PKI) user authentication; Section 4 discusses the push to use SSL client-side PKI. Then, we discuss our exploration. Section 5 frames the basic questions; Section 6 and Section 7 report the experiments. Finally, in Section 8 and Section 9, we consider the implications.

2 Web Information Services

Currently, the Web is the dominant paradigm for information services. Typically, the browser issues a request to a server and the server responds with material the browser renders.

Language of the Interaction From the initial perspective of a browser user (or the crafter of a home page), these "requests" correspond to explicit user actions, such as clicking a link or typing a URL; these "responses" consist of HTML files.

However, the language of the interaction is richer than this, and not necessarily well-defined. The HTML content a server provides can include references to other HTML content at other servers. Depending on the tastes of the server operator and the browser, the content can also include executable code; Java and Javascript are fairly universal. This richer content language provides many ways for the browser to issue requests that are more complex than a user might expect, and not necessarily correlated to user actions like "clicking on a link."

As part of a request, the browser will quietly provide parameters such as the browser platform and the REFERER (sic)—the URL of the page which contained the link that generated this request.

Issues such as caching at the browser site or an intermediate firewall can complicate this model further. [7]

In the current computing paradigm, we also see a continual bleeding between Web interaction and other applications. For example, in many desktop configurations, a server can send a file in an application format (such as PDF or Word), which the browser happily hands off to the appropriate application; non-Web content (such as PDF or Word) can contain Web links, and cause the application to happily issue Web requests.

Web Information Services Surfing through hypertext documents constituted the initial vision for the Web—and, for many users, its initial use. However, in current enterprise settings, the interaction is typically much richer: users (both of the browser and server) want to map non-electronic processes into the Web, by having client users fill out forms that engender personalized responses (e.g., a list of links matching a search term, or the user's current medical history) and perhaps have non-Web consequences (such as registering for classes or placing an Amazon order).

In the standard way of doing this, the server provides an HTML `form` element which the browser user fills out and returns to a *common gateway interface (CGI)* script (e.g., see Chapter 15 in [22]).

This `form` element can contain `input` tags that (when rendered by the browser) produce the familiar elements of a Web form: boxes to enter text; boxes (with a "browse") tag to enter file names for upload; radio buttons; checkboxes; etc. For each of these tags, the server may specify a `name` (which names the parameter being collected from the user) and a default `value`. The server content associates this form with a `submit` action (typically triggered by the user pressing a button labeled "Submit"), which transforms the parameters and their values into a request to specific URL. (If the `submit` action specified the `GET` method, the parameters are pasted onto the end of the URL; if the `POST` method, the parameters are sent back in a second request part.)

However, this submit URL specifies an executable script, not a passive HTML file, in the "Web directory" at the server. When a server receives a request for such a script, it invokes the script; the script can interrogate request parameters, such as the form responses, interact with other software at the server side, and also dynamically craft content to return to the browser.

3 Authentication and Security

3.1 Authenticating the User

In enterprise settings, the server operator may wish to restrict content only to browser users that are authorized. In a situation where the browser user is requesting a service via a `form`, the server operator may wish to authenticate specific attributes about the user, such as identity and the fact that the user authorizes this request. The Web paradigm provides several standard avenues to do this.

Client Address For one example, the server may restrict requests to client machines with specific hostname or IP address properties.

Passwords With *basic authentication* (or the *digest authentication* variant), the server can require that the user present a userid and password, which the browser collects via a special user interface channel and returns to the server. The server requesting the authentication can provide some text that the browser will display in the password-prompt box. Alternatively, the server may also collect such authenticators as part of the `form` responses from the user.

With these various forms of password-based authentication, the server operator would be wise to take steps to ensure the passwords and other sensitive data are not exposed in transit, such as:

- by offering the entire service over an SSL channel;
- by having the form submitted by the `POST` method, so the responses are not cataloged in histories, logs, `REFERER` fields, etc..

Indeed, if neither the user nor server otherwise expose a user's password, and if the user has authenticated that he is talking to the intended server, then a strong case can be made that a properly authenticated request requires the user's awareness and approval. The password had to come from somewhere!

Weaknesses Depending on the configuration of such a server, it is possible that the authentication happens only once. In such a scheme, once a user has authenticated, subsequent requests may never require re-authentication.

Password-based systems also have other risks. Users may pick bad passwords or share them across services; the authentication is not bound to the actual service (that is, we have no non-repudiation); the adversary may mount online guessing attacks (Pinkus et al has recently considered some interesting countermeasures here [26]); users may not check that they are connected to correct server, making them vulnerable to bogus sites that look similar (i.e. Spoofing [6, 33, 34]).

Cookies The server can establish longer state at a browser by saving a *cookie* at the browser. The server can choose the contents, expiration date, and access policy for this cookie; a properly functioning browser will automatically provide this cookie along with any request to a server that satisfies the policy. Many distributed Web systems—such as PubCookies [28]—use one of the above mechanisms to initially authenticate the browser user, and then use a cookie to amplify this authentication to a longer session at that browser, for a wider set of servers.

Cookie-based authentication can also be risky. Fu et al [8] discuss many design flaws in Cookie-based authentication schemes; PivX [31] discusses many implementation flaws in IE which allows an adversarial site to read other sites' cookies.

3.2 Validating User Input

Besides authenticating the user, another critical security aspect of providing Web services is ensuring that the input is correct.

Issues here can occur on two levels:

- An adversarial user can exploit server-side script vulnerabilities by carefully crafting *escape sequences* that cause the server to behave in unintended ways. The canonical example here is a server using user input as an argument in a shell command; devious input can cause the server to execute a command of the user's choosing.
- On an application level, an adversarial user can change the request data, such as form fields or cookie values. The canonical example here is a commerce server that collects items and prices via a form.

Standard good advice is that the script writer thoroughly vet any *tainted* user input [10], and also verify that critical data being returned has not been modified [27].

4 Client-Side PKI

4.1 Overview

When prodded, PKI researchers (such as ourselves) will recite a litany of reasons why PKI is a much better way than the alternatives to carry out authentication and authorization in distributed, multi-organizational settings. As we mentioned in the introduction, browser-based keystores and client-side SSL are a dominant emerging paradigm for bringing PKI to large populations. Some organizations currently using client-side SSL include Dartmouth College, MIT, the Globus Grid project, IBM WebSphere, and many suppliers of VPN software.

On the application end, numerous players preach the client-side SSL is a better way to authenticate users than passwords. We cite a few examples culled from the Web:

- The W3C: "SSL can also be used to verify the users' identity to the server, providing more reliable authentication than the common password-based authentication schemes." [30]
- Verisign: "Digital IDs (digital certificates) give web sites the only control mechanism available today that implements easily, provides enhanced security over passwords, and enables a better user experience." [14]
- Thawte: "Most modern Web browsers allow you to use a Personal Email Certificate from Thawte to authenticate yourself to a Web server. Certificate-based authentication is much stronger and more secure than password-based authentication." [24]

- Entrust: "... identify or authenticate users to a Web site using digital certificates as opposed to username/password authentication where passwords are stored on the server and open to attacks." [5]

Recent research on user authentication issues also cite client-side SSL as the desired (but impractical) solution. [8, 26]

The clear message is that Web services using password-based authentication would be much stronger if they used client-side SSL instead.

4.2 At the Server

How does this work?

As noted earlier, the *secure sockets layer* permits the browser and user to establish an encrypted, integrity-protected channel over which to carry out their Web interaction: request, cookies, form responses, basic authentication data, etc. The typical SSL use includes server authentication; newer SSL uses permit the browser to authenticate as well. The server operator can require that a client authenticate via SSL, can restrict access based on how it chooses to validate the client certificate; server-side CGI scripts can interrogate client-certificate information, along with the other parameters available.

4.3 At the Browser

Typically, browser-based storage relies on some form of database system, such as Berkeley DB 1.85 [4], to store both certificates and private key material in a "secure" manner.

Netscape/Mozilla Netscape stores its security information in a subdirectory of the application named `.netscape` (Mozilla uses `.mozilla`). There are two files of primary interest: `key3.db` which stores the user's private key, and `cert7.db`[2] which stores the certificates recognized by the browser's security module.

Both of these files are binary data, stored in the Berkeley DB 1.85 format.Additionally, these files are password protected so that any application capable of reading the Berkeley DB format is still required to provide a password to read the plaintext or to modify the files without detection.

A detailed description of the techniques used to securely store users' keys is beyond the scope of this paper, but we point readers to [11, 12, 13, 19, 21] for details.

Internet Explorer/Windows IE stores the private key and certificate as a binary "blob" in the registry by default [3]. This absolves the key pair creator from having to worry about key management issues on the machine. This approach makes the private key and certificate as secure as the underlying operating system, in that the operating system is responsible for allowing/denying access to the registry.

Microsoft recommends against this behavior, noting that there is no password protection on the private key by default, and that the key is only as secure as the user's account [18]. This implies that if an attacker were to gain access to a user's account or convince the user to execute code with the user's privileges, the attacker would be able to use the private key at will, without having to go through any protections on the key (such as a password challenge).

One way to remedy the lack of password protection is to "export" the private key, placing it in a password protected `.pwl` file (for IE 3 and earlier) or a `.pfx` file which stores the key in PKCS#12 (for IE 4 to current versions).

Additionally, there are two independent developments in Microsoft's key store technology which are relevant to our work.

First, all versions of the CryptoAPI since the version which shipped with IE 4 provide a means for displaying a warning or a password prompt when the private key is being used. We refer to a key which displays a warning only as a *medium-security key* and a key which asks for a password as a *high-security key*

[2] In December 2002, NSS 3.7 introduced `cert8.db`, but it is nothing radically different.

Second, the latest versions of Microsoft Windows (Win2000 and XP) give applications an interface for protecting data called the *Data Protection API (DPAPI)*. In short, DPAPI provides OS-level data protection services to applications, allowing them to use the OS to store things like private keys and passwords. Applications use DPAPI via two functions which are part of the CryptoAPI.

4.4 Historical Vulnerabilities

Netscape/Mozilla's keystore has remained fairly static, and to the best of our knowledge, historical vulnerabilities are also current vulnerabilities.

Microsoft's IE, on the other hand, has gone through a number of revisions. Perhaps the most comprehensive list of problems with Microsoft's key storage system over the years comes from Peter Gutmann.

The first vulnerability applies to situations where the private key is stored in the registry. With a tool such as the "Offline NT Password & Registry Editor" [23], it is possible for an attacker to access a user's account given physical access to the computer on which the account resides in a few minutes. Since registry-stored keys are not password protected, an attacker can use the private key of the account's owner at will for as long as they are logged on. Additionally, an attacker could export the key to a floppy disk (password protecting it with a password that the attacker chooses), and then use tools like Peter Gutmann's or our modified version of OpenSSL to retrieve the key offline.

The second vulnerability comes from the format in which the private key is stored on disk once it has been exported (in a `.pwl` or `.pfx` file). There is a tool named *breakms*, available from Gutmann's web site [9], which performs a dictionary attack to discover the password used to protect the file and outputs the private key.

Prior to our work, we have not seen attacks against medium-security or high-security keys, nor have we seen vulnerabilities demonstrated in DPAPI.

Recent anonymous postings[2] discuss potential vulnerabilities in Microsoft's Digital Rights Management Scheme (MS-DRM), but the private keys used in this scheme are included with the application (e.g. the Windows Media Player) and shipped with the core system (i.e. `blackbox.dll`). Although interesting, this discussion is distinct from the browser-based storage of personal private keys.

5 The Question

We believe PKI is valuable and that secure Web information services are important. We also realize that any deployment will require considerable effort and user education (as we participate in such a deployment here at Dartmouth). Hence, we believe that it's important to ask: *Does it work?*

If we encourage user populations to enroll in client-side PKI, and encourage service providers to migrate current services to use client-side SSL authentication and to roll out new services this way, have we achieved the desired goals: that service requests are authenticated from user A only when user A consciously issued that request?

To this end, we carried out a series of experiments in order to evaluate the effectiveness of using the browser and client-SSL as a component of a client-side PKI. (However, some of our attacks have a wide range of applications, and could potentially be used to subvert other authentication schemes as well. We focus on PKI because it is claimed to be the strongest—and in theory, it could be).

Discussions of usability and security stress the importance of the system behaving as the user expects [35], and the dangers in creating systems whose proper use is too complex [1, 32]. In the case of client-side PKI, we have two classes of users to consider:

- The user of the client browser, who requests services
- The user of the server, who sets up and deploys the Web application that provides these services.

As a consequence, we weren't focused on bizarre bugs (or extremely carefully constructed applications), but on general usability. If users on either end follow the "path of least resistance"—standard out-of-the-box configurations and

advice—does it work?

Section 6 and Section 7 describe our experiments. Section 8 will consider countermeasures and implications.

6 Our Experiments: Usage of Keys

A basic assumption underlying client-side SSL is that the client's certificate and private key are used only for SSL requests that the client user was actually aware of and approved.

Is this true?

6.1 GET Requests

The language of Web interaction—even when restricted to HTML only, and no Javascript—makes it very easy for a server S_A to send content to a browser B, that causes the browser to issue an arbitrary request r to an arbitrary server.

If one wants this request r to be issued over SSL, we've found that a reliable technique is to use the HTML `frameset` construction, itself offered over server-side SSL. Figure 1 sketches this scenario; Figure 2 shows some sample HTML.

Basic Techniques A `frameset` enables a server S_A to specify that the browser should divide the screen into a number of frames, and to load a specified URL into each frame. The adversarial server can specify *any* URL for these frames. If the server is careful with frame options, only one of these frames will be visible at the browser. However, the browser will issue all the specified requests.

This behavior appears to violate the well-known security model that "an applet can only talk back to the server that sent it" because this material is not an applet.

We stress that this is different from full-blown cross-site scripting. S_A is not is using a subtle bug to inject code into pages that are (or appear to be from) other servers. Rather, S_A is using the standard rules of HTML to ask the browser to itself load another page.

Framesets and SSL In previous work [33], we noticed that if server S_A offers a frameset over server-side SSL, but specifies that the browser load an SSL page from S_B in the hidden frame, then many browser configurations will happily negotiate SSL handshakes with both servers—but (in the cases we tried) the browser will only report the S_A certificate.

So, we wondered what would happen if S_B requested client-side authentication.

- In Mozilla 1.0.1/Linux (RedHat 7.3 with 2.4.18-5 kernel), using default options, the browser will happily use a client key to authenticate, without informing the user.
- In IE 6.0/WindowsXP, using default options and any level key, the browser will happily use a client key to authenticate, without informing the user, if the user has already client-side authenticated to S_B.

  If the user has not, a window will pop-up saying that the server with a specified hostname has requested client-side authentication; which key, and is it OK? (Potentially, server keep-alive configurations could also force this behavior.)
- In Netscape 4.79/Linux (RedHat 7.3 with 2.4.18-5 kernel), using default options, the browser will pop-up a window saying that the server with a specified hostname has requested client-side authentication; which key, and is it OK? Then the browser will authenticate.

The request to S_B can easily be a `GET` request, forging response of a user to a Web form.

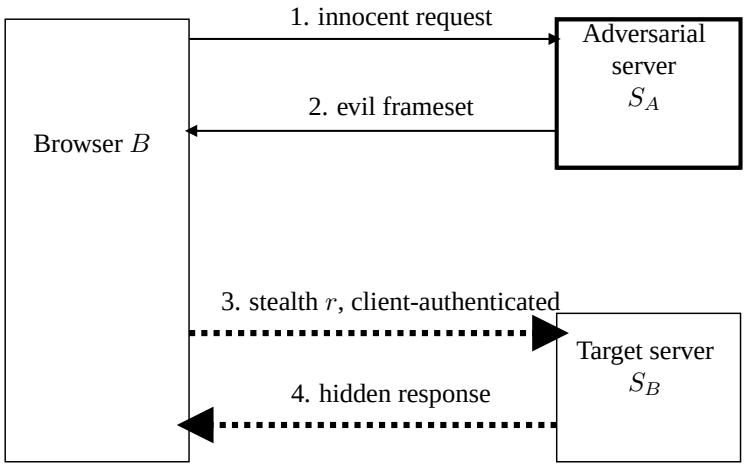

Figure 1: To borrow client-side authentication, the adversary needs to convince the browser's user to visit an SSL page at the evil server. Using the ordinary rules of Web interaction, the evil server can provide content that causes the browser to quietly issue a SSL request, authenticated with the user's personal certificate, to the victim server.

```
<html>
<frameset rows="*,1" cols="*,1" frameborder="no">

<frame src="f0.html" name="f0" scrolling="no">
<frame src="blank" name="b0" scrolling="no">
<frame src="blank" name="b1" scrolling="no">
<frame src="https://cobweb.dartmouth.edu:8443/cgi-bin/test.pl?
    debit=1000&
    major=None%3B%20I%27m%20withdrawing%20from%20the%20college"
   name="f1" scrolling="no">
</frameset>
<noframes> no frames </noframes>
</html>
```

Figure 2: HTML permits an adversarial server to send a frameset to a browser. The browser will then issue requests to obtain the material to be loaded into each frame. A deviously crafted frameset (such as the one above) appear to be an ordinary page. If an adversarial server includes a form response in the hidden frame, the browser will submit an SSL request to an arbitrary target server via GET. In many scenarios, browsers will use client-side authentication for the GET; with the devious frameset, the user may remain unaware of the request, the use of his personal certificate, and the response from the target.

```
<html>
<head>
<SCRIPT LANGUAGE=javascript>
  function fnTemp()
   {
     document.myform.submit();
   }
</script>
</head>
<body onload="fnTemp()">

<form name="myform" method="post"
  action="https://cobweb.dartmouth.edu:8443/cgi-bin/test.pl">
<input name="debit" value="1000">
<input name="major" value="Hockey">
<input type="submit" value="Submit Form">
</form>
</body>
</html>
```

Figure 3: A web page such as this uses Javascript to cause the browser submit an SSL request to an arbitrary target server via POST. In many scenarios, browsers will use client-side authentication for the POST. If an adversarial server specifies that this page be loaded into a hidden frame, then the user may remain unaware of the request, the use of his personal certificate, and the response from the target.

6.2 POST Requests

Some implementors preach that no sane Web service should accept `GET` response to Web forms. However, services that use `POST` responses are also vulnerable. If we extend the adversary's tools to include Javascript, then the adversarial page can easily include a `form` element with default values, and an `onload` function that submits it, via an SSL `POST` request, to S_B. Figure 3 sketches this code.

Sending this page via a hidden frame further hides the request and the response.

Again, browsers will use the user's personal certificate to authenticate this request.

6.3 Implications

As we noted earlier, it is continually touted that client-side SSL is superior to password-based authentication.

Suppose the operator of an honest server S_B offers a service where authorization or authentication are important. For example:

- Perhaps S_B wanted to prove that its content was served to particular authorized parties (and perhaps to prove that those parties requested it—one thinks of Pete Townshend or a patent challenge).
- Perhaps S_B is offering email or class registration services, via `form` elements, to a campus population.

If S_B had set up their site with server-side SSL, and required basic authentication or some other password scheme, then one might argue that a service can be carried out in a user's name only if that user authorized it, or shared their password.

However, suppose S_B uses "stronger" client-side SSL. With Mozilla and default options, a user's request to S_B can be forged by a visit to an adversarial site S_A. With IE and default options, a user's request can be forged if the user has already visited S_B.

With Netscape or IE, a user's request to S_B can probably be forged without the user noticing if the adversarial site simply claims that S_A *also* requires client-side authentication. With this bogus claim—which sounds quite reasonable

in a campus enterprise environment—the user will expect to see the password prompts, etc. (and probably won't notice any fine print). A further concern is that the fine print expresses host name, which users may not necessarily be able to correlate to URL or the content in specific browser real estate.

We note that this authentication-borrowing differs from the standard single-sign-on risk that, once a user arms their credential, their browser may silently authenticate to any site the user consciously visits. In our scenario, the user's browser silently authenticates to any site of the adversarial site's choosing.

Limits We could not demonstrate a way for the adversary, using the tools of sending standard HTML and Javascript to users with standard browsers, to forge a response to a file upload `input` tag (see further discussion below) or to forge `REFERER` fields (although `telnet` links look promising).

7 Our Experiments: Storage of Keys

In Section 6, the adversary can only borrow use of the client's private key, under constrained circumstances. It would be much more interesting to steal the key outright.

7.1 Stealing Netscape/Mozilla Keys from Foolish Users

In Netscape and Mozilla, the private key is stored in the `key3.db` file, as discussed in Section 4.3. Knowledge of this file and the user's keystore password enables easy extraction of the private keys.

There is a significant amount of information available describing the algorithms used to store the private keys and certificates in the Netscape/Mozilla browser. As a result, there are a number of tools which can be used to view and modify Netscape's and Mozilla's key stores.

The NSS Security Tools (available from Mozilla [19]) allow users to add and delete the key and certificate databases directly. While it is possible to vandalize key databases, none of the tools seemed to give direct access to the private key. NDBS 2.0, a free tool available from Carnegie Mellon University allows programmatic access to Netscape's key and certificate databases [13]. The tool is extremely powerful in that it enables applications to capture the private key in a number of formats and do what they wish with it—store it to a file, use it to generate signatures, post it on a web site, etc. NDBS is really a set of Java classes which enable programmers to write code which accesses the key store.

A simple social engineering attack can allow us to capture users' private keys. The attack relies on a misleading interface presented on a web page to trick users into sending us the file containing their private keys, the file containing their certificate, and the password needed to access those files. Once collected, the items are given to a program which uses NDBS to discover the private keys and forge the digital signatures.

The attack begins by users pointing their web browser to an official looking site that claims to be the Dartmouth Authentic Really Secure Service (DARSS), which is advertised as some special service that requires PKI. The user is told that the DARSS must verify the key pair, and ensure that the user is actually authorized to use the key. This is done by filling out a form which points the browser to the files containing the private key and certificate [3] and prompts the user for the password protecting the files. Note that the password `input` tag can have the `type=password` option, so that the characters don't echo on the string, in order to increase the user's feeling of security.

This is a flat-out lie. The DARSS is not verifying anything, and the password is not used to check authorization. In actuality, the files and password are shipped to a directory and archived, so that another program can discover and collect the private keys and forge digital signatures.

The system has two main components, the front end which is the SSL site responsible for advertising the DARSS and the form tricking the user into sending us their files and password, and the back end which is the system which uses either NDBS (for Netscape keys) or OpenSSL (for *exported* IE keys) to open the files and capture the private key.

[3] In Netscape, the keys are in a default place relative to the user's home directory, and the home directory is implicit if the user types in a relative path.

7.2 Stealing Netscape Keys from Wiser Users

The attack of Section 7.1 above requires that the users are vulnerable to the social engineering technique of simply asking for their password and keystore paths. Can we modify this attack to be effective against more cautious users?

Our lab has some Web spoofing experience [33, 34]. Drawing on these tricks, we easily constructed a server page that opens a window that looks and acts very much like the standard Netscape classic-skin keystore password prompt. (Our initial foray into Mozilla left a "Javascript" warning at the top; using the browser's own `alert` pop-up did not disable echoing of the password prompt.) Thus, for many Netscape users, the adversary can easily obtain the keystore password.

The next step is to get the encrypted password files. The HTML spec permits the server to specify a default value for `input` tags of `type=file`; and Netscape leaves keys in a known place, so that's a start. A bit of experimentation revealed a way to create submittable forms with `type=file` fields whose text boxes and "Browse" buttons were not visible to the user (even without Javascript). However, then we ran into a stumbling block: the standard browsers deliberately disregard the HTML spec, and do *not* actually render default values for `type=file` upload tags. Except with explicitly buggy browser versions that were quickly patched, the user must actually type something. (It appears we have hints of a trusted path from user to server!)

To get around this, we need to upgrade the adversary's toolkit to include an executable running on the user's platform. This executable locates and sends back the encrypted key file. (It has been rumored that the password itself can be found by careful inspection of the browser process's address space, but we did not try that.)

The permeability of modern computing environments, plus the general lack of a trusted path from browser to user, makes Netscape a risky place to leave client private keys.

7.3 Stealing IE Low-Security Keys

In older Windows platforms, obtaining client private keys was fairly straightforward, as discussed in Section 4.4. Modern versions of Windows and IE have added features which make obtaining the client's private key a bit more challenging. Examples include:

- providing medium and high security options to the key generation functions, resulting in either a warning or a password prompt when the private key is accessed by an application.
- giving all applications a means to request that the OS securely store data (such as a private key or password) via the DPAPI.

However, these are just features; some applications (e.g. legacy applications) do not use these newer security features. Of particular interest to us is the ability to generate a "low-security key"—a key which can be used by any application running with the user's privileges without warning the user that the key is in use. Of further interest is the fact that this is the CryptoAPI's default behavior.

Peter Gutmann raised serious concern over the `CryptExportKey()` function found in the CryptoAPI back in 1998[9]. Specifically, with the default key generation, any program running under the user's privileges may call this function and obtain a copy of the user's private key.

We were curious to see if the latest versions of the CryptoAPI have remedied this issue. Our conclusion: "no". We were able to construct a small executable which, when run on a low-security (default) key, quietly exports the user's private key with no warning.

7.4 Stealing IE Medium-Security and High-Security Keys

The Windows CryptoAPI does permit users to bring in keypairs at "medium" or "high" security levels. With both of these levels, use of the private key will trigger a warning window; in the high-security option, the warning window requests a password.

Consequently, the attack of Section 7.3 may not work; when the executable asks the API to export the private key, the user may notice an unexpected warning window. So our attack strategy has to improve.

7.4.1 API Hijacking

Before we discuss the specifics of stealing private keys, a brief introduction to the general method of *API Hijacking* is in order. The goal of this attack is to intercept (hijack) calls from some process (such as IE) to system APIs (such as the CryptoAPI).

Delay Loading API Hijacking uses a feature of Microsoft's linker called "Delay Loading". Typically, when a process calls a function from an imported *Dynamically Link Library (DLL)*, the linker adds information about the DLL into the process (in what is referred to as the "imports section"). This topic is discussed in depth by Matt Pietrik [25], but we present a very brief overview.

When a process is loaded, the Windows loader reads the imports section of the process, and dynamically loads each DLL required. As each DLL is loaded, the loader finds the address of each function in the DLL and writes this information into a data structure maintained in the process's imports section known as the *Import Address Table (IAT)*. As the name suggests, the IAT is essentially a table of function pointers.

When a DLL has the "DelayLoad" feature enabled, the linker generates a small stub containing the DLL and function name. This stub is placed into the imports section of the calling process instead of the function's address. Now, when a function in the DLL is called by a process for the first time, the stub in the process's IAT dynamically loads the DLL (using `LoadLibrary()` and `GetProcAddress()`). This way, the DLL is not loaded until a function it provides is actually called—i.e. its loading is delayed until it is needed.

For delay loading to be used, the application must specify which DLLs it would like to delay load via a linker option during the build phase of the application. So, how does one use delay loading on a program for which they can not build (possibly because they don't have the source code—i.e. IE)?

DLL Injection The answer is to redirect the IAT of the victim process (e.g. IE) to point to a stub which implements the delay loading *while the process is running.*

The strategy is to get the stub code as well as the IAT redirection code into an attack DLL, and *inject* this DLL into the address space of the victim process. Once the attack DLL is in the process, the IAT redirection code changes the victim's IAT to point to the stub code. At that point, all of the victim process's calls to certain imported DLLs will pass through the attack DLL (which imported DLLs are targeted and which functions within those DLLs are specified by the attack DLL—i.e. the attacker gets to choose which DLLs to intercept). This implements a software man-in-the-middle attack between an application and certain DLLs on which it depends.

The Windows OS provides a number of methods for injecting a DLL into an process's address space (a technique commonly referred to as "DLL Injection"). The preferred method is via a "Windows Hook", which is a point in the Windows message handling system where an application can install a routine which intercepts messages to a window.

7.4.2 Hijacking the CryptoAPI

Using the techniques above, we were able to construct a couple of programs which allowed us to intercept function calls from IE to the CryptoAPI. This is particularly useful for stealing medium or high security private keys which display warning messages when used (in a client-side SSL negotiation, for example).

The idea is to wait for IE to use the key (hence, displaying the warning or prompting for a password), and then get a copy of the private key for ourselves—*without triggering an extra window that might alert the user.*

The Attack Essentially, the attack code is two programs: an attack DLL with the IAT redirection code and the delay loading stubs, and one executable to register a hook which is used to inject the attack DLL into IE's address space.

The strategy is:

- Get the attack DLL and executable onto the victim's machine (perhaps through a virus or a remote code execution vulnerability in IE).
- Get the executable running. This installs a Windows hook which gets the attack DLL injected into IE's address space.
- Change IE's IAT so that calls to desired functions in the CryptoAPI (crypt32.dll) are redirected to our attack DLL.
- At this point, we have complete control and are aware of what IE is trying to do. For example, if we specify `CryptSignMessage()` to be redirected in our attack DLL, then every time IE calls this function (e.g. to do an SSL client-side authentication), control will pass to our code.
- We know that the user is expecting to see a warning in this case, so we take advantage of the opportunity to do something nefarious—like export the private key. In our current demo, the adversarial code exports the private key, so the warning window will say "exporting" instead of "signing" at the top[4].

  This could be remedied by hijacking the call which displays the warning. In fact, this would allow us to disable all such warnings, but we did not implement this.

In sum, API Hijacking can lead to a number of quite effective attacks that can undermine many underlying security mechanisms. In addition to getting the private key (as in our demo), it would be possible (and easier) to simply "use" the private key to forge signatures.

8 Analysis

8.1 Short-Term Countermeasures

Many institutions, including our own, are working on trying to deploy client-side PKIs and Web information services that use them.

As we have demonstrated, using standard browser-based keystores and believing the literature's claims that client-SSL along authenticates the user can be dangerous.

- Devious HTML can create scenarios where client-side authentication with a personal certificate does not imply the person was aware of or approved this request.
- A single adversarial executable can remove the private key.
- Current user interfaces and default options governing a browser's use of personal keys can be murky.

Those using client-side PKI should take note—particularly if they are migrating from password-based schemes.

Borrowing Authentication Certainly, competent and careful programming can increase the security of the system. However, we have found that the "right thing to do" in many situations is not always so easy to find in the standard literature. For example, none of the sources we cited touting the advantages of client-side SSL over passwords pointed out that while a password in an SSL-protected form response can authenticate the user (because the requester knew the password), client-side SSL provides no such guarantees.

One colleague suggested the use of hidden form fields. This is a good idea. At a minimum, we recommend that providers of services intending to use personal certificate authentication do a two-step process: first, authenticate the

[4] In our demo, we fail the IE request, so the user sees a "404" error; however, more polite adversarial code might carry out the requested cryptographic operation and return the response to the user.

requester and construct a form that includes hidden field containing the name of requester, a timestamp, all signed (or MAC'd); then, check for this field on the form submission. So far, we have not been able to figure out how an adversarial server can fool such a service (without using an implementation bug, such as code-injection).

Forcing browsers to periodically re-negotiate with the server via the server's `Keep-Alive` can be used to kill SSL sessions. This would make it a little trickier for the adversary to borrow authentication—e.g., more likely to trigger warnings.

However, these approaches are still unsatisfactory in the long run: the person owning this personal certificate as at the mercy of the server for correct and reasonable use of the private key. Further, history shows adding security after the fact, rather than designing it in from the beginning, is not always effective.

Borrowing Keys To combat key theft, we recommend that deployers insist on medium-security and high-security keys for IE (since this complicates the attack), and consider making keys non-exportable. Moving private keys to a separate device (e.g., a token) can provide further protections against OS weaknesses.

As news continues to show [17], code injection and other malicious executables remain a constant risk in contemporary networked desktop environments. Using long RSA moduli does not help if the underlying platform is easily permeable. We recommend that deployers not overlook the basics of aggressively maintaining system security on machines housing private keys.

User Interfaces We also recommend that deployers very carefully explore the certificate use options in the browser platforms in their user population, and educate users to choose wise configurations and pay attention to the implications. As noted earlier, many of the browsers we saw, when configured to warn of key use, only warned with the hostname—so deployers would be wise not to mix different security domains on the same host.

8.2 Long-Term

On a deeper level, one might argue that the term "personal certificate" is a misnomer. In the best case, using a browser-stored key for client authentication (via SSL) authenticates two HTTP endpoints. No person need be involved at all. Even though much conventional wisdom implies client-side SSL can replace weaker authentication, SSL designers state that client-side SSL is simply not intended to replace application-level security [29]. With client-side SSL (and perhaps many single sign-on schemes), "what the user knows" has been replaced by "what some complex software on the user's desktop does in response to complex stimuli."

A number of research directions arise here.

Usability Consequently, it would be interesting to re-consider the client Web authentication system, from the perspective of security and usability. To cite just a few design principles: [35]

- "The path of least resistance" for users should result in a secure configuration.
- The interface should expose "appropriate boundaries" between objects and actions.
- Things should be authenticated in the user's name only as the "result of an explicit user action that is understood to imply granting."

One might quip that it has hard to find a principle here that the current paradigm does *not* violate.

We stress again that these principles should apply to both the client user, as well as the IT staffer setting up a Web page.

Trusted Paths One natural area for further attention is a *trusted path.* Our earlier work [33] built trusted paths from the browser to the user. We also need trusted paths in the other direction (e.g., a Web equivalent of the "secure attention

key") and an easy way for Web service writers to invoke that. This may not be as much of a stretch as one might think; already, the standard browsers depart from the HTML specification and require that a user type a value into a `file` input tag. Wouldn't an `authenticate` input tag be much easier than trying to work through cryptographic hidden fields? Adding another level of personal certificate that only was invokable via such a tag (and perhaps even signed something) would help. Indeed, in the online literature, we see that Netscape provides a *signed form* facility in Java that forces some user involvement [20] (but IE does not provide native support); we also see some brief discussion for authentication tags [16].

Until then, ongoing work [26] in using *reverse Turing tests* to defeat robotic probing could assist a server in setting up a trusted path.

Another area for further attention is the user's mental model of Web interaction. For this new `authenticate` tag (or even current warning windows) to be effective, the screen material to which it applies should be clearly perceivable by the user. Even adopting the "Basic Authentication" model of letting the server demanding the authentication provide some descriptive freetext might help—instead of "hostname wants you to authenticate," the browser window might say "...in order to change your class registration—are you sure?". (Netscape's Signed Forms goes in this direction, but it permits the server to provide HTML content that can enable some types of signature spoofing.)

To rephrase a point from our earlier work [33], the community insists on strict access controls protecting the client file system from server content, but neglects access controls protecting the user's perception of the client user interface.

Tokens with UI On a system level, we recommend that further examination be given to the module that stores and wields private keys: perhaps a trustable subsystem with a trusted path to the user. As a device which has a very rich and complex interaction with the rest of the world, browsers can often behave in unexpected and unclear ways. Such a device should not be the cornerstone of a secure system.

Many researchers have long advocated that private keys are too important to be left exposed on a general-purpose desktop. We concur. However, we also go further that the user interface governing the use of the private key is too important to be left on the desktop—and too important to be left to the sole determination of the server programmer, through a content language not designed to resist spoofing.

9 Conclusions

One might look at this work from a narrow or broad perspective.

From a narrow perspective, here's what we do:

- We explicitly demonstrate that client-side authentication with personal certificates does not necessarily authenticate the person. (In the hoopla surrounding client-side PKI, we had not seen this issue raised. Indeed, the opposite is preached.)
- We explicitly demonstrate that browser-based keystores—even IE medium-security ad high-security keys—are easily permeable in modern desktop environments. (We had not seen attacks on these keys, nor had we seen use of our approach applied to cryptographic APIs.)

However, we intend this paper to have a broader perspective as well. We believe in PKI. Much work is being done in many places to try to bring PKI to users; considerable investment of effort is being focused on the client-side browser paradigm. We humbly suggest that some of this investment might be better spent rethinking the basic model.

Acknowledgments

The authors are grateful to our many colleagues—both in the Dartmouth PKI Lab and in the greater Internet2 community—for their help and advice.

The authors have also received support from the Mellon Foundation, the NSF, AT&T/Internet2, and the U.S. Department of Justice (contract 2000-DT-CX-K001). The views and conclusions do not necessarily reflect the sponsors.

References

[1] R. Anderson. Why Cryptosystems Fail. *Communications of the ACM*, pages 32–40, November 1994.

[2] Anonymous. Microsoft's digital rights management scheme - technical details. `http://cryptome.org/ms-drm.htm`.

[3] AspEcrypt. Opening a Certificate Store. `http://www.aspencrypt.com/task_certs.html`.

[4] Berkeley DB. `http://www.sleepycat.com`.

[5] Buyer's guide - web portal security solution. `http://www.entrust.com/resources/pdf/buyers_guide.pdf`, November 2001.

[6] E. Felten, D. Balfanz, D. Dean, and D. Wallach. Web spoofing: An internet con game. In *20th National Information Systems Security Conference*, 1997.

[7] E. Felten and M. Schneider. Timing Attacks on Web Privacy. In *ACM Computer and Communications Security*, 2000.

[8] K. Fu, E. Sit, K. Smith, and N. Feamster. Dos and Don'ts of Client Authentication on the Web. In *USENIX Security*, 2001.

[9] Peter Gutmann. How to recover private keys for Microsoft Internet Explorer, Internet Information Server, Outlook Express, and many others - or - Where do your encryption keys want to go today? `http://www.cs.auckland.ac.nz/~pgut001/pubs/breakms.txt`.

[10] J. Hamilton. *CGI Programming 101*. CGI101.com, 1999.

[11] S. Henson. Netscape Certificate Database Info. `http://www.drh-consultancy.demon.co.uk/cert7.html`.

[12] S. Henson. Netscape Key Database Format. `http://www.drh-consultancy.demon.co.uk/key3.html`.

[13] Pisey Huy, Grace A. Lewis, and Ming-hsun Liu. Beyond the Black Box: A Case Study in C to Java Conversion and Product Extensibility. Technical report, Carnegie Mellon Software Engineering Institute, 2001.

[14] Implementing web site client authentication using digital ids and the netscape enterprise server 2.0. `http://www.verisign.com/repository/clientauth/ent_ig.htm#clientauth`.

[15] K. Kain, S.W. Smith, and R. Asokan. Digital Signatures and Electronic Documents: A Cautionary Tale. In *Advanced Communications and Multimedia Security*. Kluwer Academic Publishers, 2002.

[16] S. Lawrence and P. Leach. User agent authentication forms. `http://www.w3.org/TR/NOTE-authentform`, February 1999.

[17] Microsoft security bulletin ms03-004. `http://www.microsoft.com/technet/treeview/default.asp?url=/technet/secu%rity/bulletin/ms03-004.asp`, February 2003.

[18] Microsoft Authenticode Developer Certificates. `http://www.thawte.com/getinfo/products/devel/msauthenticode.html`.

[19] Mozilla. NSS Security Tools. `http://www.mozilla.org`.

[20] Netscape form signing. `http://developer.netscape.com/tech/security/formsign/formsign.html`, 1999.

[21] Netscape Communications Corp. *Command-Line Tools Guide : Netscape Certificate Management System*, 4.5 edition, October 2001.

[22] J. Niederst. *Web Design in a Nutshell (2/E)*. O'Reilly, 2001.

[23] Offline NT Password & Registry Editor. `http://home.eunet.no/~pnordahl/ntpasswd`.

[24] Personal certificates. `http://www.thawte.com/html/COMMUNITY/personal/`.

[25] Matt Pietrek. Under the hood. *Miscrosoft Systems Journal*, February 2000.

[26] B. Pinkas and T. Sander. Securing Passwords Against Dictionary Attacks. In *ACM Computer and Communications Security*, 2002.

[27] Preventing html form tampering. `http://advosys.ca/papers/form-tampering.html`, August 2001.

[28] Pubcookie: open-source software for intra-institutional web authentication. `http://www.washington.edu/pubcookie/`.

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

[30] L. Stein and J. Stewart. The world wide web security faq. `http://www.w3.org/Security/Faq/`, February 2002.

[31] Unpatched ie security holes. `http://www.pivx.com/larholm/unpatched/`.

[32] A. Whitten and J.D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *USENIX Security*, 1999.

[33] E. Ye and S.W. Smith. Trusted Paths for Browsers. In *USENIX Security*, 2002.

[34] E. Ye, Y. Yuan, and Smith S.W. Web Spoofing Revisited: SSL and Beyond. Technical Report TR2002-417, Department of Computer Science, Dartmouth College., 2002.

[35] K.-P. Yee. User Interaction Design for Secure Systems. `http://zesty.ca/sid/uidss-may-28.pdf`.