NAT'L INST. OF STAND & TECH R.I.C.
A11104 907034

NIST
PUBLICATIONS

NISTIR 5788

Public Key Infrastructure Invitational Workshop

September 28, 1995
MITRE Corporation
McLean, Virginia

William E. Burr
Editor

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Security Division
Information Technology Laboratory
Gaithersburg, MD 20899

Sponsored by:
National Institue of Standards and
Technology
MITRE Corporation
Federal Information Security Infrastructure
Program Management Office

QC
100
.U56
NO.5788
1995

NIST

NISTIR 5788

Public Key Infrastructure Invitational Workshop

September 28, 1995
MITRE Corporation
McLean, Virginia

William E. Burr
Editor

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Security Division
Information Technology Laboratory
Gaithersburg, MD 20899

Sponsored by:
National Institue of Standards and
Technology
MITRE Corporation
Federal Information Security Infrastructure
Program Management Office

November 1995

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

Abstract

This is the report of the Invitational Workshop on Public Key Infrastructure, which was jointly sponsored by the National Institute of Standards and Technology (NIST), the Security Infrastructure Program Management Office (SI-PMO) and the MITRE Corporation. A public key infrastructure provides a means for issuing and managing public key certificates, which may be used to provide security services, such as authentication, integrity, confidentiality and non-repudiation, between strangers who have no previous knowledge of each other. Papers were presented on the current state of technology and standards for a Public Key Infrastructure, management and technical issues, escrowing keys used for confidentiality exchanges, and cost models.

Disclaimer

The views expressed by the participants of this workshop are their own and do not necessarily reflect those of the sponsoring organizations. Proprietary names and model numbers are cited for clarity and do not imply recommendation or endorsement by the sponsors.

Table of Contents

1

Introduction

Introduction

Message From the Co-Chairs

Robert Rosenthal, Co-Chair, NIST

Many government and industry organizations recognize the need for privacy and security in the National Information Infrastructure or NII. Important applications in business, medicine and other areas are likely to use public key cryptography and digital signature mechanisms to provide integrity, authentication, nonrepudiation and confidentiality key management security services. These powerful security techniques, when provided on a large national or worldwide basis, will enable secure transactions among partners who may never meet each other personally, where their transactions may be exposed to unknown parties and where untrusted communications and storage systems may be prevalent. The opportunity exists now to develop a ubiquitous, interoperable Public Key Infrastructure (PKI) meeting the widest spectrum of government, industry, and personal needs.

The federal government has begun work on the design of a Public Key Infrastructure to support the management of the public key certificates and keys it needs. Guidance for this work comes from many sources including our own internal research projects, cooperative programs with industry, other federal agencies, and, interactions in open forums like this invitational workshop. At NIST, we actively solicit inputs from designers and users alike to mold and formulate the ideas that drive our government programs in this area.

We are pleased to have worked with the MITRE Corporation and the General Services Administration to plan, execute and participate in this workshop. A clear result of the discussions was the need for and willingness of the participants to cooperate and share information in this dynamic area.

Richard J. Kemp, Co-Chair, SI-PMO

The electronic business of Government requires security services to be viable. In the traditional data security model, there are five basic security services offered to end users and their applications: Authentication, Access Control, Data Integrity, Non-Repudiation, and Confidentiality. As a practical matter, these services cannot be delivered in an internetworked world (the one we are moving towards) without a Public Key Infrastructure. PKI is the essential enabling technology for accomplishing business objectives in a secure fashion.

The purpose of the Federal Information Security Infrastructure Program Management Office (SI-PMO) is to provide government wide support and coordination of Federal activities necessary to implement an information security infrastructure for the use of the Federal Government. Since the private sector will lead in the development and implementation of end application security solutions, the SI-PMO will, in coordination with NIST, interact directly and continuously with private industry. The SI-PMO must articulate the needs of the federal government as a customer community, and also understand the commercial offerings, and how commercially available information security solutions can be applied to solving the government's problems.

Public forums and workshops such as this one, serve to ensure the interchange of information most necessary for the effective development and implementation of the security infrastructure. We look forward to continued joint efforts with NIST, the MITRE Corporation, and other activities to share information and experiences in this vital evolving technical area

Shirley Kawamoto, Co-Chair, MITRE
The MITRE Corporation has worked with public key cryptography for our customers and on our own behalf for some time. Among other efforts, we conducted the Public Key Infrastructure Study, under NIST sponsorship, and are now working toward the implementation of a digital signature capability for our own use. This work has made us aware that the creation of a Public Key Infrastructure is necessary before digital signatures and other public key applications critical to the implementation of the broader information infrastructure can be achieved.

MITRE's participation in the PKI Workshop was motivated by a desire to exchange information with others working in the area. Holding the workshop was one way to assemble the parties who could contribute different perspectives and have discussions leading to more understanding of PKI issues and solutions.

The workshop met our goals. The standards community and implementors, government and industry, users and vendors were all represented. Several different countries were represented. The papers presented, as well as the accompanying discussions, were instructive and thought provoking. I want to thank all of the participants for making the workshop a success.

Attendance List

NAME	AFFILIATION	PHONE	E-MAIL
Carl Boecher	Datakey	612-890-6850	
Dennis Branstad	Trusted Info Systems	301-854-6889	Denny@tis.com
Bill Burr	NIST	301-975-2914	wburr@nist.gov
Robert Cordery	Pitney Bowes	203-924-3067	cordery@pb.com
Steven Davis	Computer Science Corp.	410-684-6428	sdavis9@csc.com
Warwick Ford	Bell Northern Research	613-765-4924	wford@bnr.ca
Jim Galvin	Trusted Info Systems	301-854-6889	galvin@tis.com
Jisoo A. Geiter	MITRE		
Gene Hilborn	CSC	410-691-6550	ghilborn@csc.com
Larry Kilgallen	LJK Software	617-558-3275	Kilgallen@LJK.com
Shirley Kawamoto	MITRE	617-271-7689	Kawamoto@mitre.org
Richard Kemp	SI-PMO	202-708-0839	richard.kemp@gsa.gov
Jan Lovorn	Dyn Corp. I&ET	703-222-1414	lovornj@USVA8.dyncorp.com
Silvio Mcali	MIT		
Chris Mitchell	Royal Holloway University of London,	+44 1784 443423	cjm@dcs.rhbnc.ac.uk
Noel A. Nazario	NIST	301-975-2837	NNazario@nist.gov
Jim Omura	Cylink	408-735-5826	omura@cylink.com
Leon Pintsov	Pitney Bowes	203-924-3316	pintsov@pb.com
William T. Polk	NIST	301-975-3348	wpolk@nist.gov
Bill Rohland	Data Key Inc.	612-890-6850	
Ken Rose	Cylink	410-263-0260	
Robert Rosenthal	NIST	301-975-3603	Rosenthal@nist.gov
Mindy Rudell	MITRE	703-883-7939	rudell@Mitre.org
Chandra Srivastava	Tandem Computers Inc.	408-285-4247	srivastava.Chandra@tandem.com
Denis Trcek	Jozef Stefan Institute	+386 61 177 3733	denis@e5.ijs.si
Burton Tregub	Cylink	408-735-6667	Burt@CYLINK.COM
George Usher	SI-PMO	202-708-4896	george.usher@gsa.gov

Agenda
PKI Invitational Workshop
28 September 1995

8:30	Opening Remarks
8:45	Workshop Goals and Expectations
9:45	PKI Components 1
	Reference Procedures for Certification Infrastructure - Denis Trcek, Joseph Stefan Institute
	X.509 V3 - Warwick Ford, Northern Telcom
10:45	Break
11:00	Key and Certificate Management
	Alternatives to RSA Using Diffie-Hellman with DSS - Jim Omura, Cylink
	Trusted Third Party Based Key Management Allowing Warranted Interception - Chris Mitchell, Information Security Group, Royal Holloway University of London
12:00	Lunch
1:00	Key and Certificate Management (continued)
	Method for Certifying Public Keys in a Digital Signature Scheme - Silvio Micali, MIT
	Interworking Between Certification domains - Rich Ankney, Fischer International, presented by Jan Lovorn
2:00	Models
	A Certificate-Based Authorization Model - Rich Ankney, Fisher International, presented by Jan Lovorn
	Toward Secure Enterprise Management - Steven Davis, CSC
	Trust through Metering in the Public Key Infrastructure - Robert Cordery, Pitney Bowes
3:45	Break
	Token-Based Information Security for Commercial and Federal Information Networks - William Rohland, Datakey
	Communicating the Strength of Private Key Storage - Larry Kilgallen, LKJ Software
4:00	Closing Remarks

2

Compact Certification of Public Keys

Silvio Micali, MIT

Compact Certification of Public Keys[1]

by

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

Abstract. We put forward a PRACTICAL method for certifying public keys. Our method works with an arbitrary hierarchy of authorities, and always produces very short certificates, independent of the number of authorities.

1 The Problem of Public-Key Certification

PUBLIC KEYS. Ever since the Diffie-Hellman cryptosystem and the RSA, public-key cryptography has been the system of choice for a variety of tasks; most prominently, encryption and digital signatures.

In a public-key encryption scheme, each user U computes a pair of matching keys: an *encryption* key PK_U and a corresponding *decryption* key SK_U. Informally, such a pair of keys possesses the following two properties:

1. knowledge of SK_U enables one to efficiently decrypt any message encrypted with PK_U; and

2. PK_U does not "betray" SK_U (i.e., knowledge of PK_U does not help one to compute SK_U in any efficient way).

User U publicizes PK_U (which is thus also referred to as U's *public key*) and keeps secret SK_U (which is thus also referred to as U's *secret key*). Because of property (1), the publicity of PK_U enables anyone to send U an encrypted message understandable by U, without having agreed beforehand on a common code with U. Because of property (2), despite the publicity of PK_U, only U can understand the encrypted messages sent to him.

[1]The technology of this paper is protected by U.S. Patent No. 5,420,927

In a digital signature scheme, each user U computes a pair of matching keys: a *verification key*, denoted by PK_U, and *signing key*, denoted by SK_U. Again, PK_U does not betray its corresponding SK_U, U publicizes PK_U and keeps secret SK_U (which explains both why PK_U and SK_U are also called U's *public* and *secret* keys, and our choice of notation). User U's digital signatures are strings possessing the following (informally expressed) properties:

1′ The digital signature of U of a message m is easy to compute with knowledge of SK_U, but practically impossible to compute without it. On the other hand,

2′ Just knowledge of PK_U suffices to verify whether a given string σ is U's digital signature of m.

In sum, anyone can (knowing PK_U) verify U's digital signatures, but only U can (knowing SK_U) produce them.

THE NEED FOR PUBLIC-KEY CERTIFICATION. The usefulness of a public-key system crucially depends on the *publicity* of its public keys. Indeed, the more PK_U is known to be U's public verification key, the more people can verify U's digital signatures.

Equally crucial to the system, however, is the *genuinity* of its public keys (i.e., ensuring that each public key is correctly associated to his owner). Indeed, let X be an impostor who computes a pair of matching verification and signing keys, (V, S). Assume now that X and succeeds in convincing a user Y that V is the verification key of U. Then, knowing the corresponding signing key S, X can easily digitaly sign any message m relative to V. Thus, each such signature will be believed by Y to be generated by U.

To ensure both publicity and genuinity, there is a general consensus that public keys should be *certified*. But, *how should we certify public keys?*

For the sake of concreteness, let us discuss the issue of public-key certification when these public keys are the verification keys of a digital signature scheme. (The same problems and solution arise if they were public encryption keys, or any other type of public keys.)

Should it be common knowledge that PK_U is the verification key of user U, then there is no need of certifying it: PK_U can be considered *de facto* certified. It appears, however, that this common knowledge can arise only for the keys of very few individuals or institutions.

A very simple certification method may consist of having the user himself distribute in person his public verification key (e.g., to his closest friends on new-year's eve). Such a method, of course, is hardly practical on a large scale.

A more practical method consists of having an authority A give each user U a certificate for his public key PK_U. Assume that A has an already-certified public verification key (e.g., one that is common knowledge to all users in the system). Then, this certificate may consists of A's digital signature of, say, the following declaration "PK_U is the public verification key of user U." This declaration may be expressed in a suitably standard format. For concreteness, let A digitally sign (an encoding of) the pair (U, PK_U).[2] Let us denote this signature/certificate as $SIG_A(U, PK_U)$. This certificate is issued by A only after she has made sure that U is the owner of PK_U, so as to enforce genuinity.

This certificate for PK_U may be inserted in a data base widely accessible, or, better yet, be given to U himself. The latter option enables U to present the certificate for his public key whenever he presents his digital signature of some message m, $SIG_U(m)$, relative to that key. This gives the recipient of U's digital signature all he needs to verify it. Indeed, to verify that $SIG_U(m)$ really is U's digital signature of m, the recipient performs two steps. First, he verifies that $SIG_U(m)$ is a correct digital signature of m relative to the verification key PK_U (no matter to whom PK_U may belong). Second, he verifies that $SIG_A(U, PK_U)$ is the correct digital signature of A —relative to her already certified public key— of the pair (U, PK_U).

HIERARCHICAL CERTIFICATES. Unfortunately, the last envisaged public-key certification is only deceivingly practical. This is so because of the genuinity requirement. Indeed, in order to certify that a user U is the owner of verification key PK_U, authority A must perform some type of check. For instance, she may verify U's identity via an ID card. Alternatively, in order to make sure that U indeed owns PK_U (and thus knows its corresponding secret signing key SK_U), A may ask U to sign a standard message, such as "This is just a try." Better yet, she may ask U to sign "This is a try, R" where R is a number that A randomly chooses after U presents her with

[2]Alternatively, A may digitally sign the string obtained by concatenating U with PK_U (assuming that U represents both the user and his identifier, and that each identifier has the same length, so that it will be clear where U ends and PK_U begins).

PK_U.[3] In general, U may be asked to undergo a variety of checks before A digitally signs (U, PK_U), and some of these may require U to appear in person before A.

Now, while a reasonable computer may be able to produce the digital signatures of millions of certificates per year, it is inconceivable that a single authority may properly identify millions of users per year. Thus, to enforce properly the genuinity requirement, there must be not just a single authority A, but numerous such authorities: $A_1, A_2, \ldots$ For instance, each A_i may be a branch of the Post Office or of a given Bank. Indeed, such entities are already trusted to a large extent, and are well distributed over the territory so that it is not too inconvenient for a user to access one of them.

It is not advisable, however, that all authorities share the same secret signing keys. In such a case, in fact, the issuer of a certificate would be untraceable, and untraceability and corruptability tend to travel together...

Assume, therefore, that each authority A_i has her own public verification key, PK_i, and its corresponding secret signing key, SK_i. Then, each A_i may produce a certificate for the public key of a user U by signing (U, PK_U) relative to her own public key PK_i.

Now, a new problem arises: while in the case of a single authority, A, it is reasonable to assume that A's verification key needs no certification because it is universally known within the system, it is unreasonable to hypothesize that the same holds for each PK_i. Thus A_i's certificate for PK_U is not a "self-contained" document: one can verify that the certificate consists of a digital signature of (U, PK_U) relative to some key PK_i, but one has no idea whose public key PK_i may be (nor whether A_i is a proper authority).

For this reason, it has been suggested that a certificate for PK_U should not only include A_i's signature of, say, (U, PK_U) relative to PK_i, but also a certificate for PK_i. For instance, assume that authority A_i actually is the ith branch of some bank B, and that user U accesses A_i for obtaining a certificate

[3]Indeed, if a legitimate user U has already got a certificate for his public key PK_U, and an impostor X succeeds in getting hold of his signature of a fixed standard message —such as "This is a try"— then he could succeed in having PK_U be certified also as X's public key. Indeed, he may go to the authority, identify himself as X, and present PK_U, as his selected public key, together with a digital signature, relative to PK_U, of "This is a try." Though it is not immediately clear what X might do with such a certificate —since this depends on which other protocols are in place in the system— it is advisable to prevent such "cheating."

for his public key PK_U. Then, this certificate will comprise $SIG_i(U, PK_U)$, the digital signature of branch A_i relative to PK_i, as well as PK_i itself and bank B's digital signature of (A_i, PK_i), $SIG_B(i, PK_i)$, relative to its own public key, PK_B.

Similarly, however, one cannot expect the public key of every bank B to be universally known. Thus, the certificate for PK_U must also include PK_B and the signature of —say— the Federal Reserve of the string PK_B, relative to its own verification key PK_{FR}. Since this last verification key may not be universally known either (e.g., abroad), PK_{FR} and a certificate for PK_{FR} must also be included within the certificate for PK_U. And so on.

We call the resulting, nested, certificates *hierarchical certificates.*

THE COSTS OF HIERARCHICAL CERTIFICATES. A main drawback of hierachical certificates is their being very long. Indeed, a user's digital signature of a message may consist of a few hundred bits, but the certificate of his public key may easily require several thousand bits. In many applications, messages come in a standardized format (think of an electronic transfer) and can thus be represented quite compactly. In these cases, therefore, the length of a message and that of a user's digital signature of the message will be totally overpowered by the length of the user's hierarchical certificate for his public key.

The excessive length of hierarchical certificates translates in significant costs of various types.

1. *Transmission Costs.* In most electronic transactions, a user transmits his digital signature of a message over the phone or over a computer network. Thus, if the user also transmits a hierarchical certificate for his public key, then electronic transactions will be considerablly slower and more expensive. Indeed, the transmission of a hierarchical certificate requires the transmission of thousands of bits, and this requires additional time (e.g., via a modem) and additional financial costs (e.g., via a long-distance phone call).

2. *Storage Costs.* In many applications, digital signatures need to be stored for a long time, and (as long as one wishes to store self-contained documents) so need public-key certificates. Thus, the use of hierarchical certificates would force recipients to store thousands of additional bits

for each stored digital signature. This additional storage adds considerably to the recipient's overall costs and inconvenience. Indeed, storing very many bits *reliably* is quite expensive. Moreover, in certain applications (e.g., smart-card ones), the memory of the verifying/storing device may be quite limited, so that it simply cannot store more than a handful of hierarchical certificates.

3. *Verification Costs.* Verification of a digital signature also has computation- and thus time-costs that are not trivial. And hierarchical certificates cause these costs to increase by several folds. Indeed, each hierarchical certificate requires the verification of several, independent signatures. Again, these costs may translate into a financial one (because faster processors ought to be used at verification sites, and faster processors are more costly).

In sum, hierarchical certificates do provide self-contained documents, but suffer from the problem of being inconvenient and expensive. Let us thus see how to solve this problem, while retaining the original advantages.

2 Our Solution

Let us now describe a method that uses any digital signature scheme for computing short certificates for arbitrary data, and for users' public verification keys in particular. We call our certificates *compact* because they are much shorter than hierarchical certificates, and because their length stays the same no matter how many levels of authorities there may be in a particular system.

Assume again that there is some n-level hierarchy of authorities, and denote by $A_1^1, A_2^1, \ldots$ first-level authorities (i.e., those dealing directly with the users who whish to register their own public keys —e.g., bank branches), $A_1^2, A_2^2, \ldots$ second-level authorities (e.g., banks), and so on. Denote by PK_i^j the verification key of authority A_i^j. While the public keys of the authorities of the first $n-1$ levels may not be widely known, assume that the verification keys of the n-level authorities, the A_i^n's, are universally known or already certified.

COMPACT CERTIFICATES. Let U be a user who has selected (PK_U, SK_U) as his verification-signing pair and now wishes to obtain a compact certificate for PK_U. Then, U accesses a first-level authority of his choice, A_i^1, and undergoes the necessary tests to ensure that he is the legitimate owner of PK_U. If these tests are passed, A_i^1 testifies this to an authority of the second level, A_j^2. This can be done in a suitable standard format; for instance, A_i^1 sends A_j^2 her signature of (U, PK_U), relative to PK_i^1. Preferably, each first-level authority accesses a well defined second-level one in this manner. Upon receiving A_i^1's digital signature of (U, PK_U), A_j^2 can immediately verify it because she knows the public keys of all the first-levels authorities directly below her. If the signature is indeed correct, then A_j^2 sends A_k^3, the third-level authority directly above her, her own signature of (U, PK), relative to PK_k^3. And so on, until some n-level authority A_x^n receives the digital signature of (U, PK_U) from an authority of level $n-1$, whose public key is known to A_x^n. If this signature is also correct, then A_x^n digitally signs (U, PK_U). This last signature constitutes the desired certificate for PK_U. We call such a certificate *compact* because it comprises the digital signature of just one authority.

The new certificate may be given to U directly by A_x^n (indeed, his electronic address may also be sent "up the authority-path"), or may be handed "down the authority-path" so that it will be A_i^1 to give A_x^n's digital signature of (U, PK_U) to U as his certificate for PK_U. In either case, producing a new certificate will not take very long. Indeed, the all process can be totally automated. Notice, in fact, that, except for the first-level authority who must perform identification or other particular tests, each authority must just verify a signature (relative to some known public key) and produce a signature. Thus, assuming for concreteness that there are 10 different levels of authorities, a compact certificate requires the generation and the verification of 10 digital signatures, and 10 message exchanges. If properly done, this time may not exceed that taken by the interaction of the user with the first-level authority. In any case, the user will enjoy considerable benefits every time he transmits his certificate.

A compact certificate is very short because it does not need to comprise any public key (other than PK_U) nor any other signature (other than A_x^n's one) when the verification key of any n-level authority is well known. (Else, a compact certificate should contain a certificate for A_x^n's public key, but it would be shorter than a hierarchical certificate for PK_U because it would skip the public-key certification of all intermediate authorities.) Accordingly, it has much lower transmission, storage, and verification costs than a hierarchical certificate.

Enriching compact certificates. Above, we have envisioned compact certificates indicating that a given public key belonged to a given user. This information may be sufficient in some cases, but not in others. For instance, it does not indicate the type of checks performed by the first-level authority. Indeed, if U is a notary public, it would be a better idea to perform more checks than for a regular user and to indicate this additional information in the certificate itself.

This additional information may be easily included in an *enriched compact certificate*, and without increasing its length by much. For instance, each check may have a code name, and the code names of the optional checks actually performed may be incorporated in the certificate in the form of a string C. Indeed, the authorities "send upwards" their digital signatures of (U, PK_U, C), rather than just (U, PK_U), and it is the digital signature of (U, PK_U, C) by a n-level authority that constitutes the enriched compact certificate for PK_U.

A compact certificate can actually be enriched with any other additional information that is deemed valuable. For instance, such additional information, AI, may indicate that U is a notary public, that U has special duties and privileges, time information, etc. In particular, such additional information may include the name of the first-level authority who dealt with the user. This very valuable piece of information can be represented by very few bits, because the name of an authority (unlike her public key) may be quite short. For instance, if the first level authorities are Post Offices, their names can coincide with their 5-digit Zip Code. Indeed, even the full path of authorities leading from the one who deals with the user to the final n-level authority may be easily incorporated in an enriched compact certificate, if so wanted.

Of course, such additions carry additional costs, but these cost increases would have arisen also for hierarchical certificates.

Store-and-forward generation of compact certificates. Finally, let us mention one simple, low-cost, and important improvement to the generation of compact certificates. When an authority of level y, A_i^y, sends the proper authority of level $y+1$, A_j^{y+1}, her own signature of (U, PK, AI) —where AI stands for the additional information deemed appropriate— relative to her own public key PK_i^y, A_j^{y+1} may store A_i^y's signature, and then forward her own as before.

This step allows a higher degree of control over the authorities. In particular, a higher level authority cannot be corrupted so as to "initiate" a fake certification for PK_U by just sending, to the authority above her, her own signature of (U, PK_U, AI) without having received anything from an authority directly below her. If an audit about a given user/public-key certificate occurs, she must produce the digital signature of the authority below her. Now, unless this signature has indeed been received by her and kept in storage, this would be hard to do. Indeed, each authority may easily verify the signature of the authorities below her (because she knows their public keys) but not produce them (because she lacks their signing keys).

It should be noted that the cost of this extra storage is borne by the certification authorities, and not by the users (and it is to be expected that the formers are properly equipped for this task). Notice also that the cumulative storage requirement of the store-and-forward generation of compact certificates is reasonably small. Indeed, if there are ten levels of authorities, then for each public key ten digital signatures must be stored: one for each authority along a 10-long path. Therefore, this storage is roughly that required to store a single hierarchical certificate in a system with the same level of authorites. However, in the latter case (if one wishes to keep self-contained documents), this storage price must be paid by the users, and for each digital signature rather than for each public key![4]

IN SUM. Compact certificates are just as informative, secure, and easily implementable as hierarchical ones, but they are much shorter and more convenient all around.

[4]The latter storage requirement can be slightly improved, but at an extra cost. For instance, if a user V realizes that he has already received two signatures relative to the same public key PK_U, then he can store just one certificate for PK_U.

3

X.509 Version 3: Policies and Constraints

Warwick Ford, BNR

NORTEL BNR

X.509 Version 3: Policies and Constraints

Warwick Ford
September, 1995

Requirements

- One key pair and certificate suitable for many different applications
- Scalability
- Interoperability of separately-administered infrastructures
- Support for multiple different policies in one infrastructure
- Users and domain administrators can manage trust relationships, including multi-domain chains
- Automated key-pair life cycle maintenance

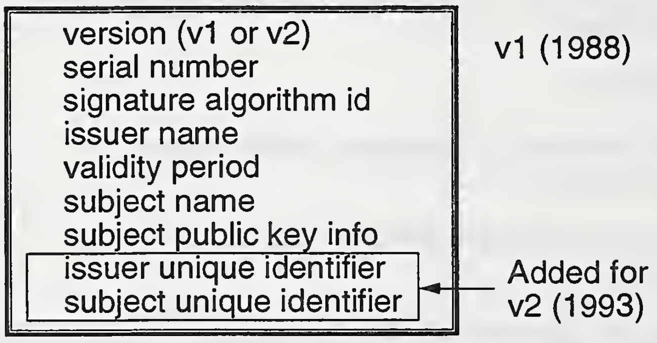
2

X.509 Evolution

- 1988: X.509 v1 certificate
- 1993: PEM - X.509 infrastructure with restrictions
- 1993: X.509 v2 certificate adds UniqueId fields; CRL revised to align with PEM CRL
- July 1994: ISO/IEC/ITU proposes X.509 corrigendum, to add extension mechanism to certificate and CRL; working draft on standard extensions issued
- October 1994: ANSI X9 (Financial Services) adopts v3 certificate format for draft standard X9.30-3
- July 1995: ISO/IEC/ITU corrigenda defining X.509 v3 certificate and v2 CRL adopted, Draft Amendment on standard extensions issued for final ballot
- April 1996: Final ballot resolution on DAM

3

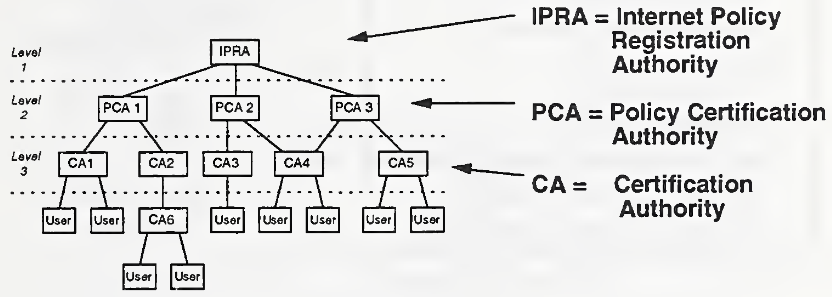
The X.509 Certificate (v1 and v2)



Signed by issuer (CA)

4

PEM – Certification Authorities


Level 1
Level 2
Level 3
IPRA
PCA 1
PCA 2
PCA 3
CA1
CA2
CA3
CA4
CA5
CA6
User
IPRA = Internet Policy Registration Authority
PCA = Policy Certification Authority
CA = Certification Authority

A CA is certified by another CA or by one or more PCAs

A CA below the PCA level can only issue certificates for X.500 names which are subordinate to that CA's name

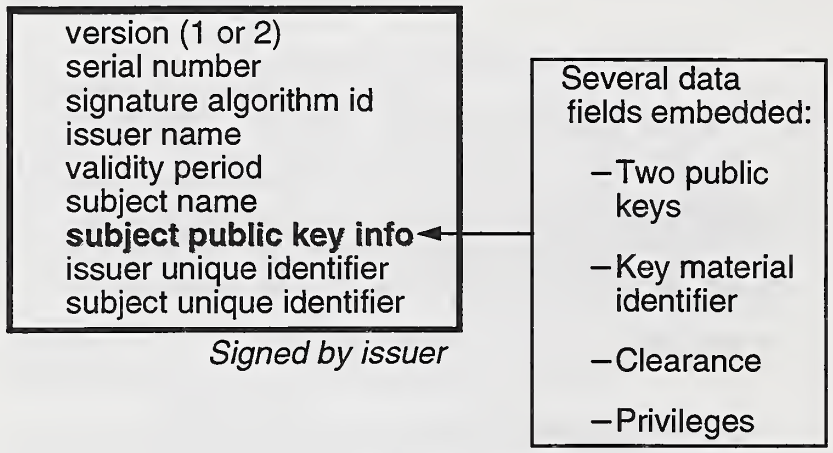
5

PEM/X.509 Shortcomings

- An X.500 name does not adequately identify a subject to a certificate user
- PEM CA hierarchy is too rigid for many applications
- Certificate chains should start at point of most trust — often close to certificate user, not at top of hierarchy
- PCAs are an unwieldy way of managing policies
- Name subordination rule does not fit with regular X.500 naming
- No support for key life cycles
- Revocation lists (CRLs) can grow excessively large

6

DoD MISSI Certificate Needs


version (1 or 2)
serial number
signature algorithm id
issuer name
validity period
subject name
subject public key info
issuer unique identifier
subject unique identifier

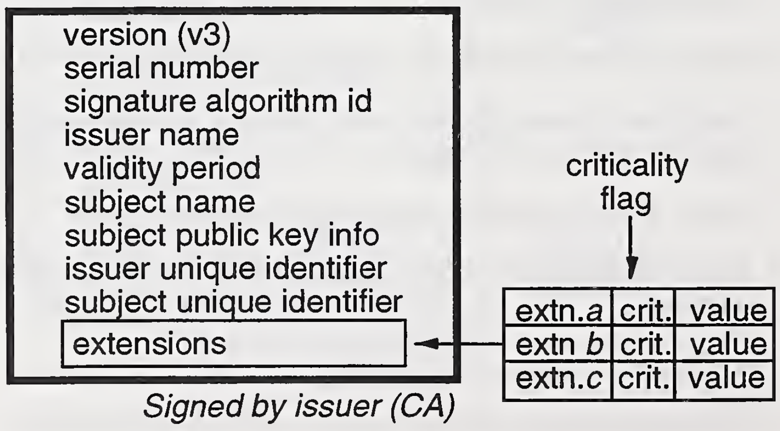
Signed by issuer

Several data fields embedded:

- Two public keys
- Key material identifier
- Clearance
- Privileges

7

The X.509 Certificate (v3)


version (v3)
serial number
signature algorithm id
issuer name
validity period
subject name
subject public key info
issuer unique identifier
subject unique identifier
extensions

Signed by issuer (CA)

extn.*a*	crit.	value
extn *b*	crit.	value
extn.*c*	crit.	value

8

Extension Criticality

- An extension field comprises:
  - extension type identifier (ASN.1 object identifier)
  - criticality flag
  - extension field value
- Criticality flag indicates whether an implementation which does not understand an extension may or may not safely ignore it
- Noncritical extensions do not inhibit interoperability
- Public-key-using systems/applications and specific policies have their own requirements as to what extensions must be present — this is NOT related to criticality

9

Key/Policy Information Extensions

- Authority Key Identifier: Identifies CA signing key by key-id or issuer/serial-no. pair
- Key Attributes: Subfields for:
  - key-id
  - key usage indicator
  - private key usage period
- Certificate Policies: Carries policy identifier(s), with optional qualifier(s)
- Key Usage Restriction: Critical extension restricting use to specific policy(s)
- Policy Mappings: Mapping of policy id at domain boundary

10

Subject/Issuer Attribute Extensions

- Subject Alternative Name
  - Internet RFC822 Name
  - Internet DNS name
  - X.400 Address
  - Another X.500 name
  - Private name form
- Issuer Alternative Name
- Subject Directory Attributes: Any other X.500 attributes that may help in identifying a subject, e.g., postal address, corporate title, photograph

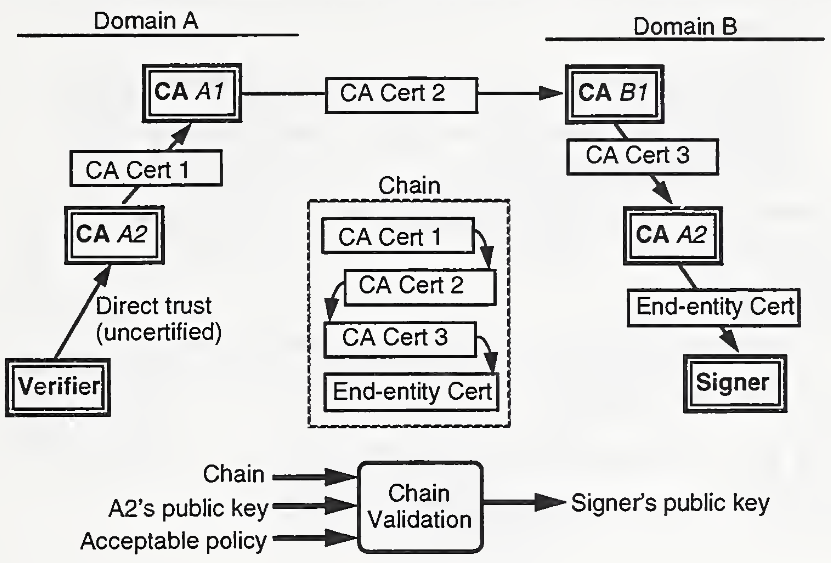
11

Chain Constraint Extensions

- When one CA certifies another, the issuer CA may constrain what certificates the subject CA can issue
- Basic Constraints
  - CA-or-End-Entity indicator
  - For a CA, can also limit chain length
  - Subtrees constraint
- Name Constraints:
  - Can constrain CA to a specified name space or set of name spaces
  - Can stipulate name subordination to apply immediately or further down the chain (including *subordinate-to-CAs-superior option*)
- Policy Constraints:
  - Can require explicit policy identifiers
  - Can inhibit policy mapping

12

Example Chain Structure



13

Policy-related Extensions

- A certificate policy is a published statement about CA policies and procedures, which a certificate user can use to decide if a given certificate is suitable for its purposes
- A certificate policy is registered and assigned a unique ASN.1 object identifier
- Certificate Policies extension: Carries policy identifier(s), with optional qualifier(s)
- Policy Mappings extension: Mapping of policy id at domain boundary
- Key Usage Restriction extension: Critical extension restricting use to specific policy(s)

14

Example Certificate Policies

- U.S. Government:
  - IRS-Taxpayer-Identification policy (defined by IRS)
  - Govt-Employee-Identification policy (defined by Govt. Policy Registration Authority)
  - Govt-Public-Identification policy (defined by Govt. Policy Registration Authority)
- Financial Industry:
  - Retail Low-value Transaction policy (defined by ABA)
- Network Service Provider:
  - AT&T Subscriber policy (defined by AT&T)
  - AT&T Operations policy (defined by AT&T)

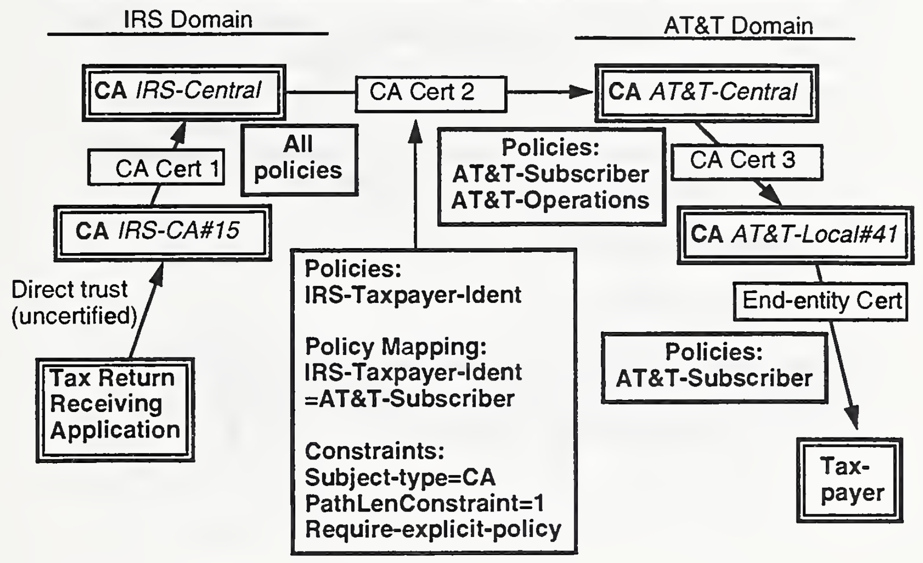
15

Example Multi-domain Scenario

- For its tax-filing application, the IRS is prepared to accept tax returns signed using digital signatures certified by a range of different service providers, e.g., AT&T
- The IRS tax return receiving application (which verifies the signatures) is supplied with the public key of some IRS CA (e.g., IRS-CA#15), and is told to accept only chains with certificate policy “IRS-Taxpayer-Identification”

16

Example Chain Structure


IRS Domain
AT&T Domain
CA *IRS-Central*
CA Cert 2
CA *AT&T-Central*
CA Cert 1
All policies
Policies: AT&T-Subscriber AT&T-Operations
CA Cert 3
CA *IRS-CA#15*
CA *AT&T-Local#41*
Policies: IRS-Taxpayer-Ident
Policy Mapping: IRS-Taxpayer-Ident =AT&T-Subscriber
Constraints: Subject-type=CA PathLenConstraint=1 Require-explicit-policy
Direct trust (uncertified)
End-entity Cert
Policies: AT&T-Subscriber
Tax Return Receiving Application
Tax-payer

17

Naming Constraints

- Subtrees constraint
  - in Basic Constraints extension
  - added in PDAM ballot resolution
  - may obsolete Name Constraints extension
- Name space constraint, Name subordination constraint
  - in Name Constraints extension

18

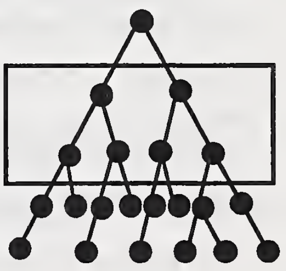
Subtrees Constraint

- Restricts all subsequent names in a certificate chain to being subordinate to one of a set of specified subtree roots
- Example: Organization Acme's primary CA is certified to only issue certificates in the subtrees:
  - c=US, o=Acme Inc.
  - c=UK, o=Acme Ltd.
- The same constraint is inherited in subsequent certificates, e.g., in certificates issued by CAs subordinate to Acme's primary CA

19

Name Constraints

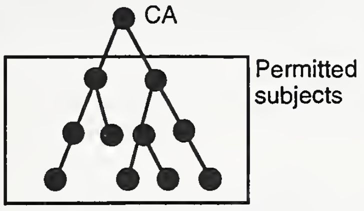
- Name space constraint
  - restricts name space for which subject CA can issue certificates (not necessarily inherited by subsequent CAs in a certificate chain)
  - can be specified as a set of subtrees, each restricted to a certain set of levels



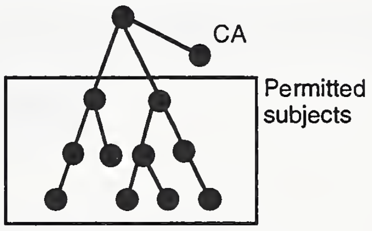
20

Name Constraints

- Name subordination constraint
  - indicates that from some point in the chain onwards, subject names must be subordinate to issuer names
  - subordinate-to-CAs-superior alternative eliminates major objections to subordinate-to-CA



Subordinate-to-CA



Subordinate-to-CA's-superior

21

Summary

- X.509 v3 certificate addresses several problems apparent in PEM/X.509 design
- X.509 v3 format now adopted; standard extensions in final ballot round
- Certificate Policies provide means for PKI to support different applications and span different domains
- Naming constraints provide means for controlling trust further along certificate chain
- Subtrees Constraint may make Name Constraints obsolete

22

4

Alternatives to RSA Using Diffie-Hellman with DSS

Jim Omura, Cylink Corp.

ALTERNATIVES TO RSA USING DIFFIE-HELLMAN WITH DSS

Jim Omura
CYLINK Corporation
910 Hermosa Court
Sunnyvale, CA 94086

ABSTRACT:

Many vendors who need security for their networking applications often assume that RSA is the only public-key technique available. Just as secure and easier to use than RSA are the techniques based on extensions to the original public-key paper by Diffie and Hellman [1]. In this note we describe the various applications of public-key cryptography and show how these Diffie-Hellman (DH) based techniques perform the same functions as RSA.

INTRODUCTION:

In 1976, Diffie and Hellman [1] started an explosion of open research in cryptology when they first introduced the notion of public-key cryptography which allows for new electronic means to handle key distribution in conventional cryptographic systems and for digital signatures in electronic messages. In this original paper Diffie and Hellman gave a limited example of a public-key system which is known today as the Diffie-Hellman key exchange. Later in 1978 Rivest, Shamir, and Adleman [2] gave a complete example of a public-key system that is popularly known as RSA. This RSA system can perform both key distribution and digital signature functions. RSA can also be used for encryption but here it has no practical advantages over conventional encryption techniques which are generally much faster.

It turns out that the original example given by Diffie and Hellman had the elements of a complete public-key system. This was discovered by El Gamal [3] who added the digital signature feature to the original Diffie-Hellman key exchange ideas. In 1994, the National Institute of Standards and Technology (NIST) adopted the Digital Signature Standard (DSS) based on a variation of this El Gamal digital signature [4]. Thus the Diffie-Hellman key exchange together with its extension to digital signatures in the form of DSS can do the same public-key functions that RSA can perform.

We first present a general discussion of the public-key method for creating digital signatures and the use of certificates. This is followed by a general discussion of the key distribution problem in conventional cryptographic systems. The remaining sections describe how the Diffie-Hellman key exchange with DSS can accomplish all the same functions as RSA in all the important applications of public-key cryptography.

DIGITAL SIGNATURES AND CERTIFICATES:

With both RSA and DSS, a person's digital signature is based on a unique pair of numbers; one that is private and another that is public. Although mathematically related, knowing a person's public number does not reveal the corresponding private number. Alice, for example, can use her private number to create a digital signature attached to her electronic message. Later another person, say Bob, can easily authenticate Alice's digital signature by only using Alice's public number. Bob can also verify the integrity of the message that Alice signed. As long as Alice keeps her private number secret, nobody can counterfeit her digital signatures while anyone can both authenticate her digital signatures and verify the integrity of her signed messages by using only her public number.

To make practical use of public-key digital signatures there needs to be established a trusted certification authority which also has a private number and public number. It is assumed that everyone in the system has knowledge of the public number of the certification authority. Thus everyone can verify the digital signatures and the integrity of any signed message of the certification authority.

Alice must first identify herself to the certification authority and submit her public number to be certified. Once the certification authority is satisfied it has properly identified Alice, it can create a message that consists of Alice's data (it may include, for example, her name, address, social security number, unique privileges, time of expiration, and her public number) which is then digitally signed by the certification authority using its private number. This electronic message that is signed by the certification authority is Alice's certificate. It is assumed that everyone in the system obtains such a unique personal certificate from the certification authority. It is also assumed that everyone in the system can verify the integrity of the data in any certificate issued by the certification authority.

The United States Postal Service will be offering the first government certification authority in the summer of 1996. A complete network of such trusted certification authorities will be needed for widespread use of digital signatures. The government's planned system of certification authorities is called the Public-Key Infrastructure (PKI).

Once Alice has her certificate, she can attach it to her signed messages. To authenticate Alice's signature, Bob can first authenticate her certificate and recover Alice's public number. He then uses what he knows is Alice's public number to authenticate her digital signature and verify the integrity of the message she signed.

CRYPTOGRAPHIC STRENGTHS OF RSA AND DIFFIE-HELLMAN BASED SYSTEMS:

The cryptographic strength of these public-key systems depend on how difficult it is for anyone to compute a person's private number given only the person's corresponding public number. For RSA this is based on the difficultly of finding the prime factors of a large integer while the Diffie-Hellman based systems depend on the difficulty of computing discrete logarithms in a finite field generated by a large prime number. Both of these are well known "hard to solve" mathematical problems. Although the discrete logarithm problem is believed to be more difficult to solve than the factoring problem, in practical terms the differences are not important [5,6,7].

In terms of ease of computations, there is also not much of a difference between the Diffie-Hellman based systems and RSA. Depending on the circumstances, there may be a computational advantage with one method over the other but with today's high speed processors and custom chips these differences are not significant for numbers from 512 bits to 1024 bits in length.

Debates have been going for some time comparing various properties of the RSA and DSS public-key digital signature schemes. Although there are some differences, the bottom line is that from a practical point of view these two public-key digital signature schemes are roughly the same in strength and computational requirements.

In a recent study, Odlyzko [8] concludes that the 512 bit numbers are considered marginally safe today while 1024 bit numbers are expected to be safe for a decade in both RSA and Diffie-Hellman based systems. Because eventually the numbers may exceed 1024 bits in length, there is now interest in elliptic curve public-key cryptosystems that were first proposed independently by N. Koblitz [9] and V.S. Miller [10]. These are not new public-key systems but are basically the Diffie-Hellman based systems using elliptic curves over finite fields. Elliptic curves over the finite field GF(2^n) are the most interesting and specific implementations have been proposed that provide a high degree of security with small numbers where n is less than 200 bits [11,12]. The RSA system does not extend to these elliptic curve cryptosystems.

There is still some reluctance to use elliptic curve cryptosystems since they have not been scrutinized as carefully as integer factorization (attack on RSA) and ordinary discrete logarithms for GF(p) where "p" is a prime number (attack on conventional Diffie-Hellman and DSS systems).

CONVENTIONAL ENCRYPTION AND KEY DISTRIBUTION:

Because public-key algorithms are computationally intensive, in practice they are generally used for creating digital signatures in electronic messages and for handling key distribution in systems using symmetric encryption algorithms. Symmetric encryption algorithms are used primarily for maintaining the privacy of information. The best known conventional encryption algorithm is the Data Encryption Algorithm which is in NIST's 1976 Data Encryption Standard (DES) [13]. DES has been around for almost 15 years and now a replacement is being discussed by several organizations. NIST has proposed the Clipper chip and the general concept of an Escrowed Key

System where the FBI has the capability (with a valid court order) to decrypt traffic using this system. The banking standards group, ANSI, is currently considering an extension of DES to Triple-DES. IDEA (International Data Encryption Algorithm), an algorithm invented by Professor James Massey and his students at the ETH in Zurich, has been used by Phil Zimmermann in his Pretty Good Privacy (PGP) security software package that was distributed on the Internet [14]. IDEA, however, is patented by ASCOM, a Swiss company that funded the work by Massey and his students. Massey has recently developed a new conventional encryption algorithm called SAFER which is not patented and is available license free [15].

Conventional encryption algorithms require a single secret key for both encryption and decryption of messages. Before the invention of public-key cryptography, key distribution required a trusted secure channel. Traditionally this channel was a trusted person who installed secret keys into the various encryption algorithms in a secure network.

With the use of public-key cryptography key distribution can be done electronically at much less cost and risk than using trusted couriers. This requires the use of digital signatures and a certification authority. In the following sections we describe how this public-key approach for key distribution can be handled just as easily with the Diffie-Hellman based system as with RSA.

NOTATIONS:

The following are the notations we will use throughout this note.

{ }	Braces indicate the Secure Hash Algorithm (SHA) which is required by the NIST Digital Signature Standard (DSS) as input to the Digital Signature Algorithm (DSA). Here {a, b} is the result when the SHA is applied to "a" concatenated with "b."
[ ]	Brackets are for any function of the Diffie-Hellman shared secret number, Z, which is used to create a conventional L bit secret key for a symmetric encryption algorithm. Here [Z] may be, for example, the L least significant bits of Z or perhaps some hash of Z which is L bits long.
I_A	This is Alice's identification which is typically attached to any message sent by Alice. It is generally a short identifier with no cryptographic elements in it.
C_A	This is a random number generated by Alice to be used in a challenge response protocol.
$S_A\{\ \}$	$S_A\{a, b\}$ is Alice's DSS signature on the hashed version of the message which is "a" concatenated with "b." There will be several protocols presented here where the DSS signature is sent without the message for which the signature is applied. Here Alice uses her permanent private number X_A to create her DSS signatures.

$S_A(\ )$	$S_A(a, b)$ is the concatenation of "a" and "b" followed by Alice's DSS signature on message "a, b." Here again Alice uses her permanent private number X_A to create her DSS signatures. Note that $S_A(a, b) = a, b, S_A\{a, b\}$.
$Cert_A$	This is Alice's certificate which contains Alice's name, Alice's privileges (and possibly other information), her permanent public number Y_A, and the trusted Certification Authority's DSS signature for this personal data.
$E_K(\ )$	This is encryption using a symmetric cryptosystem (such as DES) with key K.
$a \oplus b$	This is the bit-by-bit modulo-2 addition of two equal length binary sequences "a" and "b."

THE DIFFIE-HELLMAN KEY EXCHANGE WITH DSS:

For the Digital Signature Standard (DSS) we have common system parameters "g", "q", and "p" where "p" is the modulus prime number. For this DSS system Alice has a permanent private number X_A and the corresponding permanent public number Y_A given by

$$Y_A = g^{X_A} \bmod p .$$

Bob also has his own permanent private number X_B and corresponding permanent public number Y_B given by

$$Y_B = g^{X_B} \bmod p .$$

We assume that both Alice and Bob have certificates issued by a certification authority that includes their public numbers. Denote these certificates as $Cert_A$ and $Cert_B$ where Y_A and Y_B are Alice and Bob's permanent public numbers that are included in their certificates. Thus when Alice receives Bob's certificate, $Cert_B$, she knows with certainty that Y_B is Bob's public number. These do not change as long as the certificates are valid.

<u>Assume throughout this note that Alice and Bob have each other's certificates. Thus Alice knows with certainty Bob's permanent public number Y_B and Bob similarly knows Alice's permanent public number Y_A.</u> Certificates could be exchanged prior to a protocol or obtained from a directory service. If certificates are not distributed prior to the protocols given here then they must be included in the initial exchanges of these protocols.

The Standard Diffie-Hellman Key Exchange

In the conventional Diffie-Hellman key exchange Alice and Bob can generate a shared secret key by conducting the following transaction:

1. Alice randomly generates a secret private number R_A and computes the corresponding public number W_A given by

$$W_A = g^{R_A} \mod p$$

while Bob randomly generates his own secret private number R_B and computes his corresponding public number W_B by

$$W_B = g^{R_B} \mod p \ .$$

2. When Alice and Bob want to establish a key exchange they first exchange their public numbers where Alice sends W_A to Bob and Bob sends W_B to Alice.

$$\begin{tikzcd}\textbf{Alice} \arrow[r, shift left, "W_A"] & \textbf{Bob} \arrow[l, shift left, "W_B"]\end{tikzcd}$$

3. Alice computes the common shared secret number Z by using only Bob's public number W_B and her secret number R_A by

$$Z = W_B^{R_A} \mod p \ .$$

Bob is able to compute the same shared secret number Z by using Alice's public number W_A and his own secret number R_B by

$$Z = W_A^{R_B} \mod p \ .$$

Note that in each new transaction between Alice and Bob new private and public numbers can be used with the resulting newly computed common shared secret number Z. This type of conventional Diffie-Hellman key exchange works well in many end-to-end secure communication applications. It weakness is that there is no authentication of the public numbers that are exchanged. For example, how does Alice know that W_B is truly Bob's public number?

Challenge Response Authentication

Before continuing with authenticated Diffie-Hellman key exchanges we first exam how Alice and Bob can authenticate each other. For mutual authentication the obvious challenge response protocol using DSS is as follows:

1. Alice generates a random number C_A and sends this to Bob.

$$\begin{tikzcd}\textbf{Alice} \arrow[r, "C_A"] & \textbf{Bob}\end{tikzcd}$$

2. Bob generates a random number C_B and sends to Alice the DSS signed message C_B, $S_B\{C_A, C_B\}$.

$$\begin{tikzcd}\textbf{Alice} & \textbf{Bob} \arrow[l, "{C_B, S_B\{C_A, C_B\}}"']\end{tikzcd}$$

3. Alice sends back to Bob the DSS signature $S_A\{C_A, C_B\}$.

$$\begin{tikzcd}\textbf{Alice} \arrow[r, "{S_A\{C_A, C_B\}}"] & \textbf{Bob}\end{tikzcd}$$

Note that Eve could intercept Alice's second transmission to Bob and replace it with her DSS signed message $S_E\{C_A, C_B\}$. Bob would then believe that it is Eve that is at the other end of the link instead of Alice. Alice, on the other hand, knows that she is linked to Bob and conducts the session without knowing that Bob thinks she is Eve.

This problem can be avoided by linking some identification to each transmission in the protocol. For example in Step 1 Alice sends I_A along with C_A. In Step 2 Bob can send the DSS signed message I_B, C_B, $S_B\{I_A, C_A, I_B, C_B\}$ to Alice and in Step 3 Alice can send the DSS signed message I_A, $S_A\{I_A, C_A, I_B, C_B\}$ to Bob. In this case Eve could replace I_E for I_A in both of Alice's transmissions to Bob and also use her own DSS signature on the second transmission to Bob. Although Bob would be deceived into believing he is in communication with Eve, Alice would know that Eve is doing this and terminate this session.

This modified mutual authentication protocol with the normal practice of sending an identification with each transmission is as follows:

1. Alice generates a random number C_A and sends this to Bob.

$$\begin{tikzcd} \textbf{Alice} \arrow[r, "{I_A, C_A}"] & \textbf{Bob} \end{tikzcd}$$

2. Bob generates a random number C_B and sends to Alice the DSS signed message $I_B, C_B, S_B\{I_A, C_A, I_B, C_B\}$.

$$\begin{tikzcd} \textbf{Alice} & \textbf{Bob} \arrow[l, "{I_B, C_B, S_B\{I_A, C_A, I_B, C_B\}}"'] \end{tikzcd}$$

3. Alice sends back to Bob the DSS signature $S_A\{I_A, C_A, I_B, C_B\}$.

$$\begin{tikzcd} \textbf{Alice} \arrow[r, "{I_A, S_A\{I_A, C_A, I_B, C_B\}}"] & \textbf{Bob} \end{tikzcd}$$

In the remainder of this note we always assume that an identification of the sender is attach to any message.

Authenticated Diffie-Hellman Key Exchanges

If Alice and Bob have permanent public numbers that are certified then they can conduct the Diffie-Hellman scheme with authenticated public numbers. The problem here is that the computed shared secret number would always be the same as long as the permanent public numbers are unchanged. In general it is not a good idea to use the same secret numbers too long.

Method #1:

One way to get around this is to generate new secret and public numbers for this conventional Diffie-Hellman key exchange but send the public numbers signed with DSS signatures. This problem of authenticated key exchange has been examined by Diffie, Van Oorschot, and Wiener [16] who recommends the following authenticated key exchange protocol:

1. Alice randomly generates a secret private number R_A and computes the corresponding public number W_A which she sends to Bob.

$$\begin{tikzcd} \textbf{Alice} \arrow[r, "{I_A, W_A}"] & \textbf{Bob} \end{tikzcd}$$

2. Bob randomly generates a secret private number R_B and computes the corresponding public number W_B . He computes the shared secret number Z from Alice's public number W_A and his private number R_B. He then sends to Alice W_B together with $E_K(S_B\{I_A, W_A,$

I_B, $W_B\}$) where K = [Z].

$$\textbf{Alice} \xleftarrow{I_B,\ W_B,\ E_K(S_B\{I_A,\ W_A,\ I_B,\ W_B\})} \textbf{Bob}$$

3. Alice computes Z from Bob's public number W_B and her secret private number R_A. She then obtains the key K = [Z] and next decrypts and verifies Bob's signature on the hashed versions of the certificates and public numbers W_A and W_B. She also sends to Bob $E_K(S_A\{I_A, W_A, I_B, W_B\})$.

$$\textbf{Alice} \xrightarrow{I_A,\ E_K(S_A\{I_A,\ W_A,\ I_B,\ W_B\}).} \textbf{Bob}$$

4. Bob then uses the key K and decrypts this message and verifies Alice's signature on the hashed certificates and public numbers.

The use of the shared secret number in this protocol is necessary to prevent a person in the middle, say Eve, causing Bob to believe he is conducting the DH key exchange with her rather than Alice [16]. There are many other ways to securely bind the shared secret number Z to the signed public numbers W_A and W_B. We consider another approach next.

Method #2:

1. Alice randomly generates a secret private number R_A and computes the corresponding public number W_A which she sends to Bob.

$$\textbf{Alice} \xrightarrow{I_A,\ W_A} \textbf{Bob}$$

2. Bob randomly generates a secret private number R_B and computes the corresponding public number W_B. He also computes the shared secret number Z from W_A and R_B. Next he randomly generates a random L bit sequence C_B and computes $[Z] \oplus C_B$. He then sends to Alice W_B, $[Z] \oplus C_B$, $S_B\{I_A, W_A, I_B, W_B, C_B\}$.

$$\textbf{Alice} \xleftarrow{I_B,\ W_B,\ [Z] \oplus C_B,\ S_B\{I_A,\ W_A,\ I_B,\ W_B,\ C_B\}} \textbf{Bob}$$

3. Alice computes the shared secret number Z from Bob's public number W_B and her secret private number R_A. She can then obtain C_B and verify the signature on the SHA of I_A, W_A, I_B, W_B, C_B.

   Next she then randomly generates a random bit sequence C_A and computes $[Z] \oplus C_A$ and sends to Bob $[Z] \oplus C_A$, $S_A\{I_A, W_A, I_B, W_B, C_A\}$.

$$\begin{tikzcd}[column sep=huge] \textbf{Alice} \arrow[r, "{I_A, [Z] \oplus C_A, S_A\{I_A, W_A, I_B, W_B, C_A\}}"] & \textbf{Bob} \end{tikzcd}$$

4. Bob uses Z to find C_A and can then verify Alice's signature on the SHA of I_A, W_A, I_B, W_B, C_A .

Note that since C_B is a random bit sequence, $[Z] \oplus C_B$ is a bit level "one time pad" encryption of [Z] which is known to be absolutely secure. The key issue here is the amount of information about C_B that is revealed in $S_B\{I_A, W_A, I_B, W_B, C_B\}$. This also applies to how much $S_A\{I_A, W_A, I_B, W_B, C_A\}$ reveals about C_A which is used to encrypt Z in $[Z] \oplus C_A$.

This second method has the advantage that it does not use a particular symmetric key encryption algorithm in the protocol except for the bit level one time pad which is very simple and secure.

A Store and Forward Version of Diffie-Hellman

In the conventional Diffie-Hellman key exchange we assume that Alice and Bob are engaged in a two way protocol. Suppose Alice wants to send an encrypted message by e-mail when Bob is not available to conduct an authenticated Diffie-Hellman key exchange. We describe a store and forward Diffie-Hellman scheme which is a natural extension of the original scheme for key agreement.

Assume that Alice has Bob's certificate, $Cert_B$, and therefore has his authenticated permanent public number Y_B. Alice can generate a random secret number R_A and the corresponding public number W_A and also generate a shared secret key Z by

$$Z = Y_B^{R_A} \mod p \ .$$

She can then use the shared secret number Z to make an encryption key K = [Z] and encrypt (using DES for example) her DSS signed message M to get the encrypted message $E_K(M, S_A\{M, I_A, W_A\})$. She can then send to Bob together with this encrypted message, her session public number W_A. This can be sent in a store and forward manner system.

$$\begin{tikzcd}[column sep=huge] \textbf{Alice} \arrow[r, "{I_A, W_A, E_K(M, S_A\{M, I_A, W_A\})}"] & \textbf{Bob} \end{tikzcd}$$

Later when Bob picks up this message he can compute the shared secret number by

$$Z = W_A^{X_B} \mod p$$

where X_B is Bob's permanent secret number. $K = [Z]$ is the message encryption key that is then used to decrypt the Alice's signed message from $E_K(M, S_A\{M, I_A, W_A\})$. Here the encryption of $S_A\{M, I_A, W_A\}$ binds W_A to Alice and the shared secret key.

Key Escrow For File Encryption

Another form of the Diffie-Hellman key exchange can be used with Key Escrow Services. Suppose that Bob is a trusted Key Escrow Service with the certified public number Y_B that is known to Alice. Alice may want to encrypt her files but also wants to allow the Key Escrow Service to generate her file encryption key in the event that she misplaces it. Alice can generate a private random number R_A and encrypt her files with a key $K = [Z]$ where Z is obtained from

$$Z = Y_B^{R_A} \bmod p \ .$$

To allow the Key Escrow Service to generate her file key if she misplaces it Alice generates the public number

$$W_A = g^{R_A} \bmod p$$

which she must send to Bob, the Key Escrow Service. This sending of W_A to Bob must be done so that Bob knows that W_A belongs to Alice.

If Alice were to send $S_A(W_A)$ to Bob then Eve could intercept this and send to Bob $S_E(W_A)$. Bob would then believe that W_A belongs to Eve and later Eve could claim she lost her file encryption key and ask Bob to generate it for her. Thus Eve would obtain Alice's file encryption key.

One way to avoid this problem is for Alice to send to Bob her public number W_A, I_A, and $E_K(S_A\{I_A, W_A\})$ thus binding her public number with her identity and the shared secret number. The encryption key used here might be based on another hash of the shared secret number Z that is different from the hash of Z used to encrypt her file.

$$\begin{tikzcd}\textbf{Alice} \arrow[r, "{I_A, W_A, E_K(S_A\{I_A, W_A\})}"] & \textbf{Bob}\end{tikzcd}$$

The Key Escrow Service, Bob, can send back acknowledgment of the reception of this public number by sending back $S_B\{I_A, I_B, W_A\}$.

$$\begin{tikzcd}\textbf{Alice} & \textbf{Bob} \arrow[l, "{I_B, S_B\{I_A, I_B, W_A\}}"']\end{tikzcd}$$

An approach that is independent of the symmetric encryption algorithm $E_K(\ )$ is for Alice to randomly generate a binary sequence C_A and send to the Key Escrow Service I_A, W_A, $C_A \oplus [Z]$, $S_A\{I_A, W_A, C_A\}$.

$$\begin{tikzcd}\textbf{Alice} \arrow[r, "{I_A,\ W_A,\ C_A \oplus [Z],\ S_A\{I_A,\ W_A,\ C_A\}}"] & \textbf{Bob}\end{tikzcd}$$

The Key Escrow Service can send to Alice the same acknowledgment as given above.

Again the security of this rests on the difficulty of obtaining Z from this transmission from Alice to the Key Escrow Service which in turn depends on how much Alice's DSS signature on the SHA of I_A, W_A, C_A will reveal anything about C_A.

Key Escrow For Communication Encryption

Next suppose we want a Key Escrow System for all of Alice's Diffie-Hellman key exchange in a communication session with another person.

Following the same procedure outlined in the above Key Escrow for File Encryption, Alice now has the secret number Z which the Key Escrow Service can also compute. Suppose that the Key Escrow Service uses Z to next compute a public number

$$P_A = g^Z \mod p$$

which it then signs as $S_B(I_A, I_B, P_A)$ and returns to Alice.

$$\begin{tikzcd}\textbf{Alice} & \textbf{Bob} \arrow[l, "{I_B,\ S_B(I_A,\ I_B,\ P_A)}"']\end{tikzcd}$$

Note that there is no exchange of secret numbers between Alice and the Key Escrow Service for this Key Escrow System.

Alice now uses P_A as the her public number in an authenticated Diffie-Hellman key exchange with anyone to establish secure communications. Here Alice would send $S_B(I_A, I_B, P_A)$ in place of P_A alone in the usual authenticated Diffie-Hellman key exchange described earlier. Thus anyone can observe the exchange of public numbers and verify that the exchanged public numbers will result in a shared key that is escrowed in the Key Escrow Service that is identified.

Note that by observing the Diffie-Hellman key exchange between Alice and anyone else, the Key Escrow Service will be identified and the DSS signature of this service will be attached to the public number of Alice. If this is not the case then it will also be clear that this key exchange is not done according to the protocol established for the Escrowed Key System.

As suggested by S. Micali [17], we can use several Key Escrow Services (KESs) where all or some subset of these KESs will be needed to recover a key used in the Diffie-Hellman key exchange. Suppose there are L KESs. Assume Alice conducts the above protocol with each of the L KESs. For example, she randomly generates L independent private numbers $R_1, R_2, ..., R_L$ and computes corresponding public numbers $W_1, W_2, ..., W_L$. With R_k and W_k Alice conducts the above key escrow exchange with the kth KES to obtain a signed public number P_k for which she has the corresponding shared private number Z_k. This KES returns to Alice the signed public number $S_k(I_A, I_k, P_k)$ where

$$P_k = g^{Z_k} \mod p \ .$$

For the Diffie-Hellman key exchange Alice then uses the private number

$$Z = Z_1 + Z_2 + ... + Z_L$$

and the corresponding public number

$$P_A = g^Z \mod p = P_1 P_2 ... P_L \mod p.$$

In the Diffie-Hellman key exchange using P_A, Alice must sent all the signed public numbers of the L KESs in place of P_A.

To recover the shared secret number in the Diffie-Hellman key exchange, one would need to obtain the shared secret number that Alice has with each of the L KESs.

Another option might be that Alice can choose any of the 2^L-1 non-trivial subsets of the public numbers $P_1, P_2, ..., P_L$ and use only their products to form a public number for a particular key exchange. She would then send the corresponding subset of the public numbers signed by the KESs that are used.

Key Exchange for Broadcast Applications

Suppose Alice wants to send the same message M encrypted to many people. She can do this by generating a secret broadcast key K for the encrypted message $E_K(M)$. If she wants to send this same encrypted message to L people with fixed public numbers $Y_1, Y_2, ..., Y_L$ (which are known to Alice because she has their certificates) she can use a form of the "Store and Forward" version of Diffie-Hellman many times.

First Alice randomly generates a secret private number R_A and computes the corresponding public number W_A. For the jth person she can compute the shared secret key

$$Z_j = Y_j^{R_A} \bmod p$$

and use this shared key to make as a secret master key K(j) = [Z] to encrypt the broadcast key K as $E_{K(j)}(K)$. She can do this with each of the L intended receivers of this encrypted broadcast message.

Alice can broadcast W_A and $E_K(M)$ if the receivers do not need to authenticate the source of this broadcast message. Otherwise Alice would broadcast $E_K(M, S_A\{M, I_A, W_A\})$ with I_A and W_A. To the jth receiver she can also send separately the encrypted key $E_{K(j)}(K)$.

Note that Alice can be sending a continuously encrypted signal such as a pay TV channel that is encrypted by the secret broadcast key K. The jth subscriber that has paid a fee can then get the secret key K by receiving $E_{K(j)}(K)$ from Alice.

SUMMARY:

The important practical applications of public-key cryptography are for digital signatures in electronic messages and for key distribution in conventional encryption systems that maintain privacy in secure networks. For these applications we have shown that the public-key example introduced by Diffie and Hellman and the extension of this to the Digital Signature Standard can perform the same functions as the RSA public-key system. Although there are some detail differences, in practical terms there is little difference in performance between the two systems. **The Diffie-Hellman based systems can be used in place of RSA in any application requiring public-key cryptography.**

REFERENCES:

[1] W. Diffie and M.E. Hellman, "New Directions in Cryptography," **IEEE Trans. on Info. Theory**, vol. IT-22, pp. 644-654, Nov. 1976.

[2] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," **Commun. of the Assoc. of Comp. Mach.**, vol. 21, pp. 120-126, Feb. 1978.

[3] T. El Gamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," **IEEE Trans. on Info. Theory**, vol. IT-31, pp. 469-472, July 1985.

[4] Federal Information Processing Standards Publication (FIPS PUB) 186, "Digital Signature Standard (DSS)," National Institute of Standards and Technology, May 1994.

[5] S.C. Pohlig and M. Hellman, "A Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance," **IEEE Transactions On Information Theory**, IT-24, pp. 106-110.

[6] B.A. LaMacchia and A.M. Odlyzko, "Computation of Discrete Logarithms in Prime Fields," **Designs, Codes and Cryptography**, Kluwer Academic Publishers, 1991, pp. 47-62.

[7] G.J. Simmons, **Contemporary Cryptology: The Science of Information Integrity**, IEEE Press, New York, NY, 1992. See Chapter 10, "Cryptanalysis: A Survey of Recent Results," by E.F. Brickell and A.M. Odlyzko.

[8] A.M. Odlyzko, "The Future of Integer Factorization and Discrete Logarithms," preliminary draft paper, AT&T Bell Laboratories, Murray Hill, New Jersey, May 10, 1995.

[9] N. Koblitz, "Elliptic Curve Cryptosystems," **Mathematics of Computation**, vol. 48, 1987, pp.203-209.

[10] V.S. Miller, "Use of Elliptic Curves in Cryptography," **Advances in Cryptology - CRYPTO '85 Proceedings**, Berlin: Springer-Verlag, 1986, pp. 417-426.

[11] A.J. Menezes and S. Vanstone, "The Implementation of Elliptic Curve Cryptosystems," **Advances in Cryptology - AUSCRYPT '90 Proceedings**, Berlin: Springer-Verlag, 1990, pp. 2-13.

[12] A.J. Menezes, **Elliptic Curve Public-key Cryptosystems**, Kluwer, 1993.

[13] Federal Information Processing Standards Publication (FIPS PUB) 46-2, “Data Encryption Standard (Reaffirmed until 1998),” National Institute of Standards and Technology, December 1993.

[14] S. Garfinkel, **PGP: Pretty Good Privacy**, O’Reilly & Associates, Sebastopol, CA, 1995.

[15] J.L. Massey, “SAFER K-64: A Byte-Oriented Block Ciphering Algorithm,” pp. 1-17 in Fast Software Encryption (Ed. R. Anderson), **Proceedings of the Cambridge Security Workshop**, Cambridge, U.K., December 1993.

[16] W. Diffie, P.C. Van Oorschot, and M.J. Wiener, “Authentication and Authenticated Key Exchanges,” in **Designs, Codes and Cryptography**, Kluwer Academic Publishers, 1992, pp. 107.

[17] S. Micali, “Fair Public-Key Cryptosystems,” preliminary draft paper, MIT, Cambridge, MA, March 25, 1993.

ALTERNATIVES TO RSA USING DIFFIE-HELLMAN WITH DSS

Jim Omura
CYLINK Corporation
910 Hermosa Court
Sunnyvale, CA 94086

ABSTRACT:

Many vendors who need security for their networking applications often assume that RSA is the only public-key technique available. Just as secure and easier to use than RSA are the techniques based on extensions to the original public-key paper by Diffie and Hellman [1]. In this note we describe the various applications of public-key cryptography and show how these Diffie-Hellman (DH) based techniques perform the same functions as RSA.

INTRODUCTION:

In 1976, Diffie and Hellman [1] started an explosion of open research in cryptology when they first introduced the notion of public-key cryptography which allows for new electronic means to handle key distribution in conventional cryptographic systems and for digital signatures in electronic messages. In this original paper Diffie and Hellman gave a limited example of a public-key system which is known today as the Diffie-Hellman key exchange. Later in 1978 Rivest, Shamir, and Adleman [2] gave a complete example of a public-key system that is popularly known as RSA. This RSA system can perform both key distribution and digital signature functions. RSA can also be used for encryption but here it has no practical advantages over conventional encryption techniques which are generally much faster.

It turns out that the original example given by Diffie and Hellman had the elements of a complete public-key system. This was discovered by El Gamal [3] who added the digital signature feature to the original Diffie-Hellman key exchange ideas. In 1994, the National Institute of Standards and Technology (NIST) adopted the Digital Signature Standard (DSS) based on a variation of this El Gamal digital signature [4]. Thus the Diffie-Hellman key exchange together with its extension to digital signatures in the form of DSS can do the same public-key functions that RSA can perform.

We first present a general discussion of the public-key method for creating digital signatures and the use of certificates. This is followed by a general discussion of the key distribution problem in conventional cryptographic systems. The remaining sections describe how the Diffie-Hellman key exchange with DSS can accomplish all the same functions as RSA in all the important applications of public-key cryptography.

DIGITAL SIGNATURES AND CERTIFICATES:

With both RSA and DSS, a person's digital signature is based on a unique pair of numbers; one that is private and another that is public. Although mathematically related, knowing a person's public number does not reveal the corresponding private number. Alice, for example, can use her private number to create a digital signature attached to her electronic message. Later another person, say Bob, can easily authenticate Alice's digital signature by only using Alice's public number. Bob can also verify the integrity of the message that Alice signed. As long as Alice keeps her private number secret, nobody can counterfeit her digital signatures while anyone can both authenticate her digital signatures and verify the integrity of her signed messages by using only her public number.

To make practical use of public-key digital signatures there needs to be established a trusted certification authority which also has a private number and public number. It is assumed that everyone in the system has knowledge of the public number of the certification authority. Thus everyone can verify the digital signatures and the integrity of any signed message of the certification authority.

Alice must first identify herself to the certification authority and submit her public number to be certified. Once the certification authority is satisfied it has properly identified Alice, it can create a message that consists of Alice's data (it may include, for example, her name, address, social security number, unique privileges, time of expiration, and her public number) which is then digitally signed by the certification authority using its private number. This electronic message that is signed by the certification authority is Alice's certificate. It is assumed that everyone in the system obtains such a unique personal certificate from the certification authority. It is also assumed that everyone in the system can verify the integrity of the data in any certificate issued by the certification authority.

The United States Postal Service will be offering the first government certification authority in the summer of 1996. A complete network of such trusted certification authorities will be needed for widespread use of digital signatures. The government's planned system of certification authorities is called the Public-Key Infrastructure (PKI).

Once Alice has her certificate, she can attach it to her signed messages. To authenticate Alice's signature, Bob can first authenticate her certificate and recover Alice's public number. He then uses what he knows is Alice's public number to authenticate her digital signature and verify the integrity of the message she signed.

CRYPTOGRAPHIC STRENGTHS OF RSA AND DIFFIE-HELLMAN BASED SYSTEMS:

The cryptographic strength of these public-key systems depend on how difficult it is for anyone to compute a person's private number given only the person's corresponding public number. For RSA this is based on the difficultly of finding the prime factors of a large integer while the Diffie-Hellman based systems depend on the difficulty of computing discrete logarithms in a finite field generated by a large prime number. Both of these are well known "hard to solve" mathematical problems. Although the discrete logarithm problem is believed to be more difficult to solve than the factoring problem, in practical terms the differences are not important [5,6,7].

In terms of ease of computations, there is also not much of a difference between the Diffie-Hellman based systems and RSA. Depending on the circumstances, there may be a computational advantage with one method over the other but with today's high speed processors and custom chips these differences are not significant for numbers from 512 bits to 1024 bits in length.

Debates have been going for some time comparing various properties of the RSA and DSS public-key digital signature schemes. Although there are some differences, the bottom line is that from a practical point of view these two public-key digital signature schemes are roughly the same in strength and computational requirements.

In a recent study, Odlyzko [8] concludes that the 512 bit numbers are considered marginally safe today while 1024 bit numbers are expected to be safe for a decade in both RSA and Diffie-Hellman based systems. Because eventually the numbers may exceed 1024 bits in length, there is now interest in elliptic curve public-key cryptosystems that were first proposed independently by N. Koblitz [9] and V.S. Miller [10]. These are not new public-key systems but are basically the Diffie-Hellman based systems using elliptic curves over finite fields. Elliptic curves over the finite field GF(2^n) are the most interesting and specific implementations have been proposed that provide a high degree of security with small numbers where n is less than 200 bits [11,12]. The RSA system does not extend to these elliptic curve cryptosystems.

There is still some reluctance to use elliptic curve cryptosystems since they have not been scrutinized as carefully as integer factorization (attack on RSA) and ordinary discrete logarithms for GF(p) where "p" is a prime number (attack on conventional Diffie-Hellman and DSS systems).

CONVENTIONAL ENCRYPTION AND KEY DISTRIBUTION:

Because public-key algorithms are computationally intensive, in practice they are generally used for creating digital signatures in electronic messages and for handling key distribution in systems using symmetric encryption algorithms. Symmetric encryption algorithms are used primarily for maintaining the privacy of information. The best known conventional encryption algorithm is the Data Encryption Algorithm which is in NIST's 1976 Data Encryption Standard (DES) [13]. DES has been around for almost 15 years and now a replacement is being discussed by several organizations. NIST has proposed the Clipper chip and the general concept of an Escrowed Key

System where the FBI has the capability (with a valid court order) to decrypt traffic using this system. The banking standards group, ANSI, is currently considering an extension of DES to Triple-DES. IDEA (International Data Encryption Algorithm), an algorithm invented by Professor James Massey and his students at the ETH in Zurich, has been used by Phil Zimmermann in his Pretty Good Privacy (PGP) security software package that was distributed on the Internet [14]. IDEA, however, is patented by ASCOM, a Swiss company that funded the work by Massey and his students. Massey has recently developed a new conventional encryption algorithm called SAFER which is not patented and is available license free [15].

Conventional encryption algorithms require a single secret key for both encryption and decryption of messages. Before the invention of public-key cryptography, key distribution required a trusted secure channel. Traditionally this channel was a trusted person who installed secret keys into the various encryption algorithms in a secure network.

With the use of public-key cryptography key distribution can be done electronically at much less cost and risk than using trusted couriers. This requires the use of digital signatures and a certification authority. In the following sections we describe how this public-key approach for key distribution can be handled just as easily with the Diffie-Hellman based system as with RSA.

NOTATIONS:

The following are the notations we will use throughout this note.

{ }	Braces indicate the Secure Hash Algorithm (SHA) which is required by the NIST Digital Signature Standard (DSS) as input to the Digital Signature Algorithm (DSA). Here {a, b} is the result when the SHA is applied to "a" concatenated with "b."
[ ]	Brackets are for any function of the Diffie-Hellman shared secret number, Z, which is used to create a conventional L bit secret key for a symmetric encryption algorithm. Here [Z] may be, for example, the L least significant bits of Z or perhaps some hash of Z which is L bits long.
I_A	This is Alice's identification which is typically attached to any message sent by Alice. It is generally a short identifier with no cryptographic elements in it.
C_A	This is a random number generated by Alice to be used in a challenge response protocol.
$S_A\{\ \}$	$S_A\{a, b\}$ is Alice's DSS signature on the hashed version of the message which is "a" concatenated with "b." There will be several protocols presented here where the DSS signature is sent without the message for which the signature is applied. Here Alice uses her permanent private number X_A to create her DSS signatures.

$S_A(\ )$	$S_A(a, b)$ is the concatenation of "a" and "b" followed by Alice's DSS signature on message "a, b." Here again Alice uses her permanent private number X_A to create her DSS signatures. Note that $S_A(a, b) = a, b, S_A\{a, b\}$.
Cert_A	This is Alice's certificate which contains Alice's name, Alice's privileges (and possibly other information), her permanent public number Y_A, and the trusted Certification Authority's DSS signature for this personal data.
$E_K(\ )$	This is encryption using a symmetric cryptosystem (such as DES) with key K.
$a \oplus b$	This is the bit-by-bit modulo-2 addition of two equal length binary sequences "a" and "b."

THE DIFFIE-HELLMAN KEY EXCHANGE WITH DSS:

For the Digital Signature Standard (DSS) we have common system parameters "g", "q", and "p" where "p" is the modulus prime number. For this DSS system Alice has a permanent private number X_A and the corresponding permanent public number Y_A given by

$$Y_A = g^{X_A} \bmod p \ .$$

Bob also has his own permanent private number X_B and corresponding permanent public number Y_B given by

$$Y_B = g^{X_B} \bmod p \ .$$

We assume that both Alice and Bob have certificates issued by a certification authority that includes their public numbers. Denote these certificates as Cert_A and Cert_B where Y_A and Y_B are Alice and Bob's permanent public numbers that are included in their certificates. Thus when Alice receives Bob's certificate, Cert_B, she knows with certainty that Y_B is Bob's public number. These do not change as long as the certificates are valid.

Assume throughout this note that Alice and Bob have each other's certificates. Thus Alice knows with certainty Bob's permanent public number Y_B and Bob similarly knows Alice's permanent public number Y_A. Certificates could be exchanged prior to a protocol or obtained from a directory service. If certificates are not distributed prior to the protocols given here then they must be included in the initial exchanges of these protocols.

The Standard Diffie-Hellman Key Exchange

In the conventional Diffie-Hellman key exchange Alice and Bob can generate a shared secret key by conducting the following transaction:

1. Alice randomly generates a secret private number R_A and computes the corresponding public number W_A given by

$$W_A = g^{R_A} \mod p$$

while Bob randomly generates his own secret private number R_B and computes his corresponding public number W_B by

$$W_B = g^{R_B} \mod p \ .$$

2. When Alice and Bob want to establish a key exchange they first exchange their public numbers where Alice sends W_A to Bob and Bob sends W_B to Alice.

$$\begin{tikzcd} \textbf{Alice} \arrow[r, shift left, "W_A"] & \textbf{Bob} \arrow[l, shift left, "W_B"'] \end{tikzcd}$$

3. Alice computes the common shared secret number Z by using only Bob's public number W_B and her secret number R_A by

$$Z = W_B^{R_A} \mod p \ .$$

Bob is able to compute the same shared secret number Z by using Alice's public number W_A and his own secret number R_B by

$$Z = W_A^{R_B} \mod p \ .$$

Note that in each new transaction between Alice and Bob new private and public numbers can be used with the resulting newly computed common shared secret number Z. This type of conventional Diffie-Hellman key exchange works well in many end-to-end secure communication applications. It weakness is that there is no authentication of the public numbers that are exchanged. For example, how does Alice know that W_B is truly Bob's public number?

Challenge Response Authentication

Before continuing with authenticated Diffie-Hellman key exchanges we first exam how Alice and Bob can authenticate each other. For mutual authentication the obvious challenge response protocol using DSS is as follows:

1. Alice generates a random number C_A and sends this to Bob.

$$\begin{tikzcd} \textbf{Alice} \arrow[r, "C_A"] & \textbf{Bob} \end{tikzcd}$$

2. Bob generates a random number C_B and sends to Alice the DSS signed message C_B, $S_B\{C_A, C_B\}$.

$$\begin{tikzcd} \textbf{Alice} & \textbf{Bob} \arrow[l, "{C_B,\ S_B\{C_A, C_B\}}"'] \end{tikzcd}$$

3. Alice sends back to Bob the DSS signature $S_A\{C_A, C_B\}$.

$$\begin{tikzcd} \textbf{Alice} \arrow[r, "{S_A\{C_A, C_B\}}"] & \textbf{Bob} \end{tikzcd}$$

Note that Eve could intercept Alice's second transmission to Bob and replace it with her DSS signed message $S_E\{C_A, C_B\}$. Bob would then believe that it is Eve that is at the other end of the link instead of Alice. Alice, on the other hand, knows that she is linked to Bob and conducts the session without knowing that Bob thinks she is Eve.

This problem can be avoided by linking some identification to each transmission in the protocol. For example in Step 1 Alice sends I_A along with C_A. In Step 2 Bob can send the DSS signed message I_B, C_B, $S_B\{I_A, C_A, I_B, C_B\}$ to Alice and in Step 3 Alice can send the DSS signed message I_A, $S_A\{I_A, C_A, I_B, C_B\}$ to Bob. In this case Eve could replace I_E for I_A in both of Alice's transmissions to Bob and also use her own DSS signature on the second transmission to Bob. Although Bob would be deceived into believing he is in communication with Eve, Alice would know that Eve is doing this and terminate this session.

This modified mutual authentication protocol with the normal practice of sending an identification with each transmission is as follows:

1. Alice generates a random number C_A and sends this to Bob.

$$\begin{tikzcd}\textbf{Alice} \arrow[rrrr, "{I_A, C_A}"] & & & & \textbf{Bob}\end{tikzcd}$$

2. Bob generates a random number C_B and sends to Alice the DSS signed message I_B, C_B, $S_B\{I_A, C_A, I_B, C_B\}$.

$$\begin{tikzcd}\textbf{Alice} & & & & \textbf{Bob} \arrow[llll, "{I_B, C_B, S_B\{I_A, C_A, I_B, C_B\}}"']\end{tikzcd}$$

3. Alice sends back to Bob the DSS signature $S_A\{I_A, C_A, I_B, C_B\}$.

$$\begin{tikzcd}\textbf{Alice} \arrow[rrrr, "{I_A, S_A\{I_A, C_A, I_B, C_B\}}"] & & & & \textbf{Bob}\end{tikzcd}$$

In the remainder of this note we always assume that an identification of the sender is attach to any message.

Authenticated Diffie-Hellman Key Exchanges

If Alice and Bob have permanent public numbers that are certified then they can conduct the Diffie-Hellman scheme with authenticated public numbers. The problem here is that the computed shared secret number would always be the same as long as the permanent public numbers are unchanged. In general it is not a good idea to use the same secret numbers too long.

Method #1:

One way to get around this is to generate new secret and public numbers for this conventional Diffie-Hellman key exchange but send the public numbers signed with DSS signatures. This problem of authenticated key exchange has been examined by Diffie, Van Oorschot, and Wiener [16] who recommends the following authenticated key exchange protocol:

1. Alice randomly generates a secret private number R_A and computes the corresponding public number W_A which she sends to Bob.

$$\begin{tikzcd}\textbf{Alice} \arrow[rrrr, "{I_A, W_A}"] & & & & \textbf{Bob}\end{tikzcd}$$

2. Bob randomly generates a secret private number R_B and computes the corresponding public number W_B . He computes the shared secret number Z from Alice's public number W_A and his private number R_B. He then sends to Alice W_B together with $E_K(S_B\{I_A, W_A,$

I_B, W_B}) where K = [Z].

$$\textbf{Alice} \xleftarrow{\quad I_B,\ W_B,\ E_K(S_B\{I_A,\ W_A,\ I_B,\ W_B\})\quad} \textbf{Bob}$$

3. Alice computes Z from Bob's public number W_B and her secret private number R_A. She then obtains the key K = [Z] and next decrypts and verifies Bob's signature on the hashed versions of the certificates and public numbers W_A and W_B . She also sends to Bob $E_K(S_A\{I_A, W_A, I_B, W_B\})$.

$$\textbf{Alice} \xrightarrow{\quad I_A,\ E_K(S_A\{I_A,\ W_A,\ I_B,\ W_B\}).\quad} \textbf{Bob}$$

4. Bob then uses the key K and decrypts this message and verifies Alice's signature on the hashed certificates and public numbers.

The use of the shared secret number in this protocol is necessary to prevent a person in the middle, say Eve, causing Bob to believe he is conducting the DH key exchange with her rather than Alice [16]. There are many other ways to securely bind the shared secret number Z to the signed public numbers W_A and W_B. We consider another approach next.

Method #2:

1. Alice randomly generates a secret private number R_A and computes the corresponding public number W_A which she sends to Bob.

$$\textbf{Alice} \xrightarrow{\quad I_A,\ W_A\quad} \textbf{Bob}$$

2. Bob randomly generates a secret private number R_B and computes the corresponding public number W_B . He also computes the shared secret number Z from W_A and R_B . Next he randomly generates a random L bit sequence C_B and computes $[Z] \oplus C_B$. He then sends to Alice W_B, $[Z] \oplus C_B$, $S_B\{I_A, W_A, I_B, W_B, C_B\}$.

$$\textbf{Alice} \xleftarrow{\quad I_B,\ W_B,\ [Z] \oplus C_B,\ S_B\{I_A,\ W_A,\ I_B,\ W_B,\ C_B\}\quad} \textbf{Bob}$$

3. Alice computes the shared secret number Z from Bob's public number W_B and her secret private number R_A. She can then obtain C_B and verify the signature on the SHA of I_A, W_A, I_B, W_B, C_B.

   Next she then randomly generates a random bit sequence C_A and computes $[Z] \oplus C_A$ and sends to Bob $[Z] \oplus C_A$, $S_A\{I_A, W_A, I_B, W_B, C_A\}$.

$$\textbf{Alice} \xrightarrow{I_A,\ [Z] \oplus C_A,\ S_A\{I_A,\ W_A, I_B,\ W_B,\ C_A\}} \textbf{Bob}$$

4. Bob uses Z to find C_A and can then verify Alice's signature on the SHA of I_A, W_A, I_B, W_B, C_A .

Note that since C_B is a random bit sequence, $[Z] \oplus C_B$ is a bit level "one time pad" encryption of [Z] which is known to be absolutely secure. The key issue here is the amount of information about C_B that is revealed in $S_B\{I_A, W_A, I_B, W_B, C_B\}$. This also applies to how much $S_A\{I_A, W_A, I_B, W_B, C_A\}$ reveals about C_A which is used to encrypt Z in $[Z] \oplus C_A$.

This second method has the advantage that it does not use a particular symmetric key encryption algorithm in the protocol except for the bit level one time pad which is very simple and secure.

A Store and Forward Version of Diffie-Hellman

In the conventional Diffie-Hellman key exchange we assume that Alice and Bob are engaged in a two way protocol. Suppose Alice wants to send an encrypted message by e-mail when Bob is not available to conduct an authenticated Diffie-Hellman key exchange. We describe a store and forward Diffie-Hellman scheme which is a natural extension of the original scheme for key agreement.

Assume that Alice has Bob's certificate, $Cert_B$, and therefore has his authenticated permanent public number Y_B. Alice can generate a random secret number R_A and the corresponding public number W_A and also generate a shared secret key Z by

$$Z = Y_B^{R_A} \bmod p \ .$$

She can then use the shared secret number Z to make an encryption key K = [Z] and encrypt (using DES for example) her DSS signed message M to get the encrypted message $E_K(M, S_A\{M, I_A, W_A\})$. She can then send to Bob together with this encrypted message, her session public number W_A. This can be sent in a store and forward manner system.

$$\textbf{Alice} \xrightarrow{I_A,\ W_A,\ E_K(M,\ S_A\{M,\ I_A,\ W_A\})} \textbf{Bob}$$

Later when Bob picks up this message he can compute the shared secret number by

$$Z = W_A^{X_B} \bmod p$$

where X_B is Bob's permanent secret number. $K = [Z]$ is the message encryption key that is then used to decrypt the Alice's signed message from $E_K(M, S_A\{M, I_A, W_A\})$. Here the encryption of $S_A\{M, I_A, W_A\}$ binds W_A to Alice and the shared secret key.

Key Escrow For File Encryption

Another form of the Diffie-Hellman key exchange can be used with Key Escrow Services. Suppose that Bob is a trusted Key Escrow Service with the certified public number Y_B that is known to Alice. Alice may want to encrypt her files but also wants to allow the Key Escrow Service to generate her file encryption key in the event that she misplaces it. Alice can generate a private random number R_A and encrypt her files with a key $K = [Z]$ where Z is obtained from

$$Z = Y_B^{R_A} \mod p \ .$$

To allow the Key Escrow Service to generate her file key if she misplaces it Alice generates the public number

$$W_A = g^{R_A} \mod p$$

which she must send to Bob, the Key Escrow Service. This sending of W_A to Bob must be done so that Bob knows that W_A belongs to Alice.

If Alice were to send $S_A(W_A)$ to Bob then Eve could intercept this and send to Bob $S_E(W_A)$. Bob would then believe that W_A belongs to Eve and later Eve could claim she lost her file encryption key and ask Bob to generate it for her. Thus Eve would obtain Alice's file encryption key.

One way to avoid this problem is for Alice to send to Bob her public number W_A, I_A, and $E_K(S_A\{I_A, W_A\})$ thus binding her public number with her identity and the shared secret number. The encryption key used here might be based on another hash of the shared secret number Z that is different from the hash of Z used to encrypt her file.

$$\begin{tikzcd}\textbf{Alice} \arrow[r, "{I_A, W_A, E_K(S_A\{I_A, W_A\})}"] & \textbf{Bob}\end{tikzcd}$$

The Key Escrow Service, Bob, can send back acknowledgment of the reception of this public number by sending back $S_B\{I_A, I_B, W_A\}$.

$$\begin{tikzcd}\textbf{Alice} & \textbf{Bob} \arrow[l, "{I_B, S_B\{I_A, I_B, W_A\}}"']\end{tikzcd}$$

An approach that is independent of the symmetric encryption algorithm $E_K(\ )$ is for Alice to randomly generate a binary sequence C_A and send to the Key Escrow Service I_A, W_A, $C_A \oplus [Z]$, $S_A\{I_A, W_A, C_A\}$.

$$\begin{tikzcd} \textbf{Alice} \arrow[r, "{I_A, W_A, C_A \oplus [Z], S_A\{I_A, W_A, C_A\}}"] & \textbf{Bob} \end{tikzcd}$$

The Key Escrow Service can send to Alice the same acknowledgment as given above.

Again the security of this rests on the difficulty of obtaining Z from this transmission from Alice to the Key Escrow Service which in turn depends on how much Alice's DSS signature on the SHA of I_A, W_A, C_A will reveal anything about C_A.

Key Escrow For Communication Encryption

Next suppose we want a Key Escrow System for all of Alice's Diffie-Hellman key exchange in a communication session with another person.

Following the same procedure outlined in the above Key Escrow for File Encryption, Alice now has the secret number Z which the Key Escrow Service can also compute. Suppose that the Key Escrow Service uses Z to next compute a public number

$$P_A = g^Z \mod p$$

which it then signs as $S_B(I_A, I_B, P_A)$ and returns to Alice.

$$\begin{tikzcd} \textbf{Alice} & \textbf{Bob} \arrow[l, "{I_B, S_B(I_A, I_B, P_A)}"'] \end{tikzcd}$$

Note that there is no exchange of secret numbers between Alice and the Key Escrow Service for this Key Escrow System.

Alice now uses P_A as the her public number in an authenticated Diffie-Hellman key exchange with anyone to establish secure communications. Here Alice would send $S_B(I_A, I_B, P_A)$ in place of P_A alone in the usual authenticated Diffie-Hellman key exchange described earlier. Thus anyone can observe the exchange of public numbers and verify that the exchanged public numbers will result in a shared key that is escrowed in the Key Escrow Service that is identified.

Note that by observing the Diffie-Hellman key exchange between Alice and anyone else, the Key Escrow Service will be identified and the DSS signature of this service will be attached to the public number of Alice. If this is not the case then it will also be clear that this key exchange is not done according to the protocol established for the Escrowed Key System.

As suggested by S. Micali [17], we can use several Key Escrow Services (KESs) where all or some subset of these KESs will be needed to recover a key used in the Diffie-Hellman key exchange. Suppose there are L KESs. Assume Alice conducts the above protocol with each of the L KESs. For example, she randomly generates L independent private numbers $R_1, R_2, \ldots, R_L$ and computes corresponding public numbers $W_1, W_2, \ldots, W_L$. With R_k and W_k Alice conducts the above key escrow exchange with the kth KES to obtain a signed public number P_k for which she has the corresponding shared private number Z_k. This KES returns to Alice the signed public number $S_k(I_A, I_k, P_k)$ where

$$P_k = g^{Z_k} \mod p \ .$$

For the Diffie-Hellman key exchange Alice then uses the private number

$$Z = Z_1 + Z_2 + \ldots + Z_L$$

and the corresponding public number

$$P_A = g^Z \mod p = P_1 P_2 \ldots P_L \mod p.$$

In the Diffie-Hellman key exchange using P_A, Alice must sent all the signed public numbers of the L KESs in place of P_A.

To recover the shared secret number in the Diffie-Hellman key exchange, one would need to obtain the shared secret number that Alice has with each of the L KESs.

Another option might be that Alice can choose any of the 2^L-1 non-trivial subsets of the public numbers $P_1, P_2, \ldots, P_L$ and use only their products to form a public number for a particular key exchange. She would then send the corresponding subset of the public numbers signed by the KESs that are used.

Key Exchange for Broadcast Applications

Suppose Alice wants to send the same message M encrypted to many people. She can do this by generating a secret broadcast key K for the encrypted message $E_K(M)$. If she wants to send this same encrypted message to L people with fixed public numbers $Y_1, Y_2, \ldots, Y_L$ (which are known to Alice because she has their certificates) she can use a form of the "Store and Forward" version of Diffie-Hellman many times.

First Alice randomly generates a secret private number R_A and computes the corresponding public number W_A. For the jth person she can compute the shared secret key

$$Z_j = Y_j^{R_A} \bmod p$$

and use this shared key to make as a secret master key K(j) = [Z] to encrypt the broadcast key K as $E_{K(j)}(K)$. She can do this with each of the L intended receivers of this encrypted broadcast message.

Alice can broadcast W_A and $E_K(M)$ if the receivers do not need to authenticate the source of this broadcast message. Otherwise Alice would broadcast $E_K(M, S_A\{M, I_A, W_A\})$ with I_A and W_A. To the jth receiver she can also send separately the encrypted key $E_{K(j)}(K)$.

Note that Alice can be sending a continuously encrypted signal such as a pay TV channel that is encrypted by the secret broadcast key K. The jth subscriber that has paid a fee can then get the secret key K by receiving $E_{K(j)}(K)$ from Alice.

SUMMARY:

The important practical applications of public-key cryptography are for digital signatures in electronic messages and for key distribution in conventional encryption systems that maintain privacy in secure networks. For these applications we have shown that the public-key example introduced by Diffie and Hellman and the extension of this to the Digital Signature Standard can perform the same functions as the RSA public-key system. Although there are some detail differences, in practical terms there is little difference in performance between the two systems. **The Diffie-Hellman based systems can be used in place of RSA in any application requiring public-key cryptography.**

REFERENCES:

[1] W. Diffie and M.E. Hellman, "New Directions in Cryptography," **IEEE Trans. on Info. Theory**, vol. IT-22, pp. 644-654, Nov. 1976.

[2] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," **Commun. of the Assoc. of Comp. Mach.**, vol. 21, pp. 120-126, Feb. 1978.

[3] T. El Gamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," **IEEE Trans. on Info. Theory**, vol. IT-31, pp. 469-472, July 1985.

[4] Federal Information Processing Standards Publication (FIPS PUB) 186, "Digital Signature Standard (DSS)," National Institute of Standards and Technology, May 1994.

[5] S.C. Pohlig and M. Hellman, "A Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance," **IEEE Transactions On Information Theory**, IT-24, pp. 106-110.

[6] B.A. LaMacchia and A.M. Odlyzko, "Computation of Discrete Logarithms in Prime Fields," **Designs, Codes and Cryptography**, Kluwer Academic Publishers, 1991, pp. 47-62.

[7] G.J. Simmons, **Contemporary Cryptology: The Science of Information Integrity**, IEEE Press, New York, NY, 1992. See Chapter 10, "Cryptanalysis: A Survey of Recent Results," by E.F. Brickell and A.M. Odlyzko.

[8] A.M. Odlyzko, "The Future of Integer Factorization and Discrete Logarithms," preliminary draft paper, AT&T Bell Laboratories, Murray Hill, New Jersey, May 10, 1995.

[9] N. Koblitz, "Elliptic Curve Cryptosystems," **Mathematics of Computation**, vol. 48, 1987, pp.203-209.

[10] V.S. Miller, "Use of Elliptic Curves in Cryptography," **Advances in Cryptology - CRYPTO '85 Proceedings**, Berlin: Springer-Verlag, 1986, pp. 417-426.

[11] A.J. Menezes and S. Vanstone, "The Implementation of Elliptic Curve Cryptosystems," **Advances in Cryptology - AUSCRYPT '90 Proceedings**, Berlin: Springer-Verlag, 1990, pp. 2-13.

[12] A.J. Menezes, **Elliptic Curve Public-key Cryptosystems**, Kluwer, 1993.

[13] Federal Information Processing Standards Publication (FIPS PUB) 46-2, "Data Encryption Standard (Reaffirmed until 1998)," National Institute of Standards and Technology, December 1993.

[14] S. Garfinkel, **PGP: Pretty Good Privacy**, O'Reilly & Associates, Sebastopol, CA, 1995.

[15] J.L. Massey, "SAFER K-64: A Byte-Oriented Block Ciphering Algorithm," pp. 1-17 in Fast Software Encryption (Ed. R. Anderson), **Proceedings of the Cambridge Security Workshop**, Cambridge, U.K., December 1993.

[16] W. Diffie, P.C. Van Oorschot, and M.J. Wiener, "Authentication and Authenticated Key Exchanges," in **Designs, Codes and Cryptography**, Kluwer Academic Publishers, 1992, pp. 107.

[17] S. Micali, "Fair Public-Key Cryptosystems," preliminary draft paper, MIT, Cambridge, MA, March 25, 1993.

5

Communicating the Strength of Private Key Storage

Larry Kilgallen, LJK Software

Communicating the Strength of Private Key Storage

by Larry Kilgallen
LJK Software
Kilgallen@LJK.com

A single organization can theoretically mandate particular levels of effort to safeguard private keys, limited only by budget. But interoperation between organizations or interoperation involving unaffiliated individuals will typically not be subject to a single policy.

The latest (1995) extensions to the X.509 recommendations can provide cryptographic authentication of statements regarding safety of private keys, but only for some of the possible risks and circumstances.

It may be that for other risks and circumstances only a less-trusted mechanism for communicating key storage strength is possible. However, in all cases it is important that interchange of such information does not increase the exposure to attack.

In all cases standardized syntax for communicating information regarding strength of key storage is preferable for interoperation. Any "security by obscurity" provided by disparate syntaxes is so weak as to provide no security benefit (and not even significant competitive advantage for vendors).

Who need Protection Information ?

When public key cryptography is being used to safeguard information, certain data custodians may need to authorize access based not only identity but based on the degree of certainty with which that identity is protected.

A trust hierarchy of Certification Authorities is typically used by a data custodian to ensure that a particular public key really corresponds to the private key held by a particular individual. But the degree to which that private key is protected might vary from a hardware token with a built-in keypad for passwords down to a PKCS #5-encrypted key stored in a public directory using as a password the name of the local sports team.

One option for communicating the strength of private key storage, of course, would be to include standards for such storage in the rules of a particular certificate hierarchy. For interoperation between disparate organizations, however, hierarchy rules are already overloaded, and creating more rather than fewer hierarchies is generally viewed as not a goal by those who seek interoperation between organizations.

Password Quality Issues

PKCS #5 password based encryption of private keys obviously brings concerns about password quality. But even hardware tokens which contain private keys typically require passwords to enable beneficial use of the key, so password quality is also of concern in those cases. Possible metrics for password quality include:

- Minimum total length
  a

- Minimum number of runs of alphabetic characters
  634-5789

- Minimum number of runs of numeric characters
  qwerty.

- Minimum number of runs of punctuations characters
  b2c3d4

- Longest run of identical characters
  bbb-222

- Longest forward run of sequential characters
  Lotus-123

- Longest reverse run of sequential characters
  edcba-432

- Longest run of characters which is repeated
  451/a/451

- Longest run of characters which is repeated in reverse
  abc4/3cba

- Maximum password lifetime

When I mention enforced maximum password lifetime, it is because I feel such measures actually decrease security for many locally held private keys. Others may have different opinions. Thus, the information which must be conveyed to a data custodian is not some rating scale of private key protection quality but rather the details of protection mechanisms so that a data custodian can make a decision regarding whether a particular mechanism meets local standards. Certainly there may be cases where a common set of criteria are shared by multiple data custodians, but there will always be cases where such agreement is not possible, so the raw data regarding password storage must be available for use by data custodians who do not find themselves in agreement with some common criteria.

Notably missing from the proposed list above are two elements commonly used centralized password quality assessment.

- Password dictionary lookup
- Password history

Both of those are difficult to fully implement with hardware tokens, and may be impossible for PKCS #5 storage of private keys.

Other Protection Mechanisms

Beyond the password used to unlock a private key for beneficial use, there are other aspects of protection including:

- PKCS #5 token stored on removable media
- PKCS #5 token stored on non-removable media
- Hardware token using untrusted keyboard for password
- Hardware token with an integral password keypad
- Hardware token which commits amnesia after 10 bad tries
- Hardware token which commits amnesia after 2 bad tries
- Hardware token with fingerprint reader
- Hardware token with DNA tester
- Hardware token with DNA tester and audit trail

Particular classes of hardware tokens can be grouped for easy decisionmaking (PCMCIA cards, Smart Diskettes, ISO Smart Cards), but details of brands and models should also be available in cases where a data custodian has particular rules regarding the degree of physical hardening required, etc.

Applicability of X.509 Extensions

The 1995 X.509 revisions provide for arbitrary extensibility of public key certificates, and as such can be used to convey cryptographically signed information which is known to the CA at certificate issue time. Depending on the exact manner in which hardware tokens are deployed, it may be that a CA is in a position to certify that the private key which corresponds is stored in a particular brand and model of hardware token. In some of those cases it may even be possible for the CA to make statements regarding password strength, because the token is constructed to enforce particular rules of password strength.

For many private key protection situations, including virtually all those associated with PKCS #5 protected keys, no such CA assurance is possible.

Alternative "Soft" Mechanisms

It is conceivable that one might trust software which makes use of a private key to report information regarding security of that key. This would be vulnerable to a sophisticated attack by the keyholder, but could usefully be employed to reject access from users who had chosen an inadequate password or were storing PKCS #5 keys in a fashion which was unacceptable to the data custodian.

Even with standardized representation of unsigned information regarding private key security, generalized "underground" tools to help end users misrepresent the degree of protection provided for their public keys would be difficult to build since non-standardized protocols for various applications would each have to be addressed separately. The lack of direct usability of such tools by an attacker should further dampen any tendency of such mechanisms to spread.

Avoiding Increased Exposure

Of course network transmission of information regarding the degree of protection afforded a private key could be used as a tool by attackers in some situations. One defense against that is to require the service to which access will be attempted to transmit protection requirements to would-be accessors before the access attempt. Then accessors would not transmit information about key protection except in cases where their key protection was adequate for the standards of the object of their connection. They could, in fact, be programmed to transmit only an indication of the standards being met, rather than a full description of strengths. The resultant traffic thus becomes an indication not of the strength of key storage being used, but the minimum strength requirements being met.

One aspect in which mere minimum standard transmission is not sufficient is in specification of exact manufacturers and models being used. While transmitting such information may be important to some data custodians, in the event a major vulnerability becomes known, the information as to who is using particular models could be exploited by attackers. Encrypting that information in transmissions is a possibility but one can then descend a twisty path regarding less-than honorable services which demand information for access and then provide the information to attackers.

Standards are Crucial

With multiple certificate hierarchies, multiple private key storage methods and multiple administrative domains, information about the strength of private key storage can only be effectively shared with standard expressions for the various aspects of private key protection.

X.509 V3 provides a framework for those aspects of private key protection information which can be attested to by the Certification Authority. For other aspects which can only be handled via "soft" controls, there is no obvious framework, although a structure similar to that used withing X.509 V3 might be reasonable, encapsulated within particular applications in a protocol-specific fashion.

6

Certification Infrastructure Reference Procedures

Denis Trcek and Borka Jerman Blazic, "Jozef Stefan" Institute

CERTIFICATION INFRASTRUCTURE REFERENCE PROCEDURES

Denis Trček, Borka Jerman Blažič
"Jožef Stefan" Institute,
Ljubljana, SLOVENIA
denis@e5.ijs.si, borka@e5.ijs.si

Abstract

Implementations of security services in a global network are strongly based on asymmetric cryptography, which in turn depends on a public key certification infrastructure. Its properties and behaviour should prevent threats. Moreover, the infrastructure should be efficiently manageable and it should meet users' expectations regarding trust. Flexible guidelines and procedures are given in this paper, where all certification infrastructure security issues are logically structured. Thus it is easier to understand and to find possible solutions in an easier way.

Keywords: certification infrastructure, CA structure requirements, certificate management.

I. Introduction

Asymmetric cryptography is of immense importance for provision of security services in a global computer network. Proper binding between a user and his/her public key is required for the correct operation of protocols using public key cryptography. A certificate [4] has been introduced as an object providing that binding, which is issued by a trusted authority, called Certification Authority (CA). To serve the whole Internet community, a global system of CAs has to be established and it has to be supported by appropriate database.

The aim of this paper is to address a logical structure on all security issues of a certification infrastructure and to identify their inter-relationships. The starting point is prevention of threats that are present in every security infrastructure. Although the basic idea of a CA structure (and a security infrastructure in general) seemed clear and easy to implement, it turned out to be a hard problem. It' s been already many years since first standards [6, 7] in this field have been launched, but the only operational certification infrastructure has been established within PASSWORD [8, 9]. Even this one was not widely used, as experiments with Privacy Enhanced Mail (PEM, [7]) have shown that setting up a CA infrastructure is a complex task. Prevention of numerous threats is interleaved with policy and both depend on technical matters. Besides, corresponding standards include many implicit assumptions.

In the following sections questions about prevention of threats, management issues and user expectations along with scalability will be addressed. This paper presents a logical structure on security issues related to a public key certification infrastructure:

- naming is a separate issue;
- certification structure is a separate issue;
- key generation is a separate issue;
- creation and revocation of certificates is a separate issue;
- distribution and storage is a separate issue (database requirements);
- effective verification is a separate issue.

The other focus of this paper is based on the policy issue as one of the most important things for setting up a security infrastructure. Although policy is not formally defined in standards [5, 6, 7], it is a vital point for every certification infrastructure. This topic will be covered in more details in the following subsections.

And finally, solutions are tried to be found that would not require complex certificate modifications. Version 2 of certificate is a target of the paper.

The paper is organised as follows. In section 2 current experiences with certification infrastructure are described. Next, reference procedures and requirements are defined along with new concepts. In section 3 future work to be done in this field is outlined. It is assumed that the reader is familiar with concepts of public key cryptography, security services, certificates, CAs and X.500 directory. Detailed description of these concepts can be found in [4, 5, 6, 7].

II. Elements of Security Infrastructure

Basic security infrastructure elements are:

- ordinary users and processes that communicate over the global network;
- certification authorities or CAs, which are trusted entities for issuing certificates;

- naming authorities, which manage the name space and allocates unique names to users;
- global-timing authorities, which are responsible for consistent and authenticated time services through the global network;
- a public distributed database for storage and distribution of new and revoked certificates.

These elements should be logically separated into different categories, although a same physical entity will often play roles of more than one logical entity.

Generally, local procedures will be out of the scope of this analysis. Therefore time authorities and time-related objects will not be considered (e.g., attributes for validity in certificates or time-stamps in messages). These are required for local verification operations and secure protocols. The same holds true for digital signatures. Next, procedures for getting sequences of certificates and assuring their existence are also considered to be a local matter.

The scope of this analysis is to provide a public key certification infrastructure for proper operations of local verification procedures and secure protocols. *The requirements (set of rules) will be logically derived from a threats analysis and from additional issues, which are logically structured into different categories. However, these categories are of equal importance, and they are partially interrelated.*

II.A. Experiences With Existing CA Structure

As already mentioned, a certification infrastructure has been established within the PASSWORD project. It has followed PEM system requirements [7] with deployment of a European X.500 system. The following problem has been identified - assume that an organisational unit "a" provides a CA service for an organisation "b" in a country "c". The related certificates (and Certificate Revocation Lists - CRLs [4]) have to be stored in an entry "C=c;O=b;OU=a" in the Directory Information Base (DIB [4]). If this part of the Directory Information Tree (DIT [4]) is managed by another organisation, conformance with PEM requirements causes the following:

- Entity "a" is responsible for the data, which it can not fully control within X.500 system, because they are stored in a DSA, which is effectively managed by another organisation. Therefore every change of the data, required by the CA, could be subject to delays and effective service is not possible. Besides, there exists a possibility of attacks by an organisation that manages the DSA.

To solve the problem mentioned above, every CA should be the master of its own entry. Would managing its own DSA solve the problem? No - because of the nature of the X.500 system, new drawbacks are introduced:

- CA's DSA entry is still managed by an organisation, which manages the name space down to the level "C=c;O=b". Thus effective management of a certificate of its own DSA is not possible. Even worse, CRLs can not be accurately managed.
- If a superior DSA does not support strong authentication, it can not be issued a certificate and a security chain is broken.

Unfortunately, it is impossible to avoid the first problem mentioned above due to the nature of the X.500 system. The second problem, however, would be solved (and the first one mitigated), if the CA could arbitrarily choose a superior DSA. In this case also the requirement for a CA' s own DSA would not be so strongly demanded. To achieve this, *a naming hierarchy should be separated from a certification hierarchy.*

It is a fact that X.500 infrastructure is already widely established and that the certification structure will be incorporated in the former one. Therefore it could be thought that the organisations, managing DSAs, should also take care for certification service. Thus, an entity, which has got a right to manage a part of a naming space, automatically gains control over the corresponding certification hierarchy. This is unacceptable, as managing a global data base is completely different from providing users with certificates. If nothing else, this scheme is very inflexible.

Finally, who will provide a global TLCA (Top Level Certification Authority [7]) service to the whole Internet community? It is very likely that due to strategic reasons every country will want to be independent regarding this matter. Conclusion:

- *There will be more TLCAs in the global network.*
- *Certification hierarchy should be separated from naming hierarchy, at least in case of CAs.*

R.2.1

In the approach that follows, flexibility is preserved, which is very important for Internet growth. It is worth to mention that flexibility of Internet had a strong influence on its evolution. For example, by deployment of domain name system, routing of e-mail is independent of naming hierarchy and the latter is independent of physical address space.

II.B. CA Structure Model Requirements

Basic CA requirements are the following:

1. *Prevention of threats - a system has to be resistant against possible attacks.*
2. *Trust - this means that a user expects to find a sequence of certificates that are issued by CAs that he/she trusts in a certain context.*
3. *Management:*
   - *all the certificates and certificate revocation lists (CRLs [6]) should be easily accessible;*
   - *to be able to use the infrastructure a user should have at least an initial information, i.e., a public key of only one CA;*
   - *the structure itself should be simple - effective certificate issuing and CRL management.*

R.2.2

The reason for the first requirement is obvious. The second requirement is based on a fact that a user trusts those entities that he/she wants to trust, and not those he/she is supposed to trust. The third requirement is based again on the nature of human reasoning - if getting certificates and their verification is a complex and time consuming process, nobody will use them.

II.B.1 Trust issues

Despite the fact that a certificate basically provides binding between an entity and a corresponding public key, it is unrealistic to expect that every user will have only one certificate for all his/her activities. Certificates can be seen as an analogy to an ordinary document and every person has been issued more such documents in real life. Thus certificates will enable the user to act somehow on its basis in line with a certain *policy*, determined by the issuer of certificates.

To assure users with various sequences of certificates, a global CA structure has to be hierarchical, where *hierarchical means that only certain CAs will have a right to issue certificates to certain users*. This is one issue of a *security policy*, which can be informally defined as *a set some of rules*. Therefore security policy actually dictates every hierarchy and should be included as a field in a certificate (e.g., by using issuerUniqueIdentifier and/or subjectUniqueIdentifier in X.509(93) certificate). It is not the intention of this paper to introduce security policy with formal definitions (see e.g. [10]), but to point out its importance.

This conclusion is stated below along with additional requirements:

1. *The global CA structure should be hierarchical, where the hierarchy is defined by a policy.*
2. *Users have to have unique names, which describe their role(s), where naming is a responsibility of a naming authority. This is achieved with an introduction of unique names (UN), consisting of unique basic names (UBN) and unique aliased names (UAN). As implied, UBN is the user' s basic name - all other names are aliases and all of them are unique.*
3. *All users have to be able to generate their own key pair, where only the public component is (physically) brought to a CA.*
4. *Users have to have entries in a global, electronic and public database. A certificate is effective once it is stored in a global, public (electronic), database.*
5. *The global public database system has to support strong authentication and access control.*
6. *A CA can revoke only those certificates that were issued by this CA (CRLs must have ISSUER attributes of the same value as ISSUER fields in the sequence of certificates, contained in the CRL).*
7. *A certificate may be issued only when a prototype certificate with new public key is submitted.*
8. *Any CA is allowed to sign another user or CA if the latter conforms to the policy of this CA.*
9. *Every user has to have an access to the global timing reference and every packet should have a time stamp.*

R.2.3

Unique names (UNs) are introduced to point out the importance of treating naming as a separate issue and to prevent implications that the rules are intended for an X.500 environment only. Reasons for the requirements above are:

- Requirement R.2.3.1 - the reasons have already been given at the beginning of this subsection.
- Requirement R.2.3.2 - to prevent identification ambiguities in a global name space.
- Requirement R.2.3.3 - a secret key remains secret only if a user generates it. As keys will not be subject to frequent changes and as nowadays everybody has enough processing power at hands, this requirement is easy to fulfil.
- Requirement R.2.3.4 - it is included to prevent threats as described in subsection

II.B.2. and to fulfil the third requirement of R.2.2.

- Requirement R.2.3.5 - it is included to prevent threats, so only a legitimate user can modify his/her own entry, while others can access it using read and/or compare operations (in case of certificate this means that the SUBJECT field must be the same as entry' s unique name). Detailed reasons are given in subsection II.B.2.
- Requirement R.2.3.6 - the reason is the third requirement in R.2.2. Although every hierarchically superior CA is in a position to revoke any certificate below it by simply revoking the first subordinated certificate, this is not practical. Next, if the lowest-level certificate has to be revoked, it could be put on any CRL under the CA in question, which is in charge for a certain policy. But this would result in complicated CRL checks. Instead of checking only CRL list at the ISSUER, the whole branch of CRLs below the top one would then have to be checked.
- Requirement R.2.3.7 - it is included to prevent threats as it will be described in subsection II.B.2. Along with requirement R.2.3.4 it makes possible issuing of new certificates without a need for a user to physically appear at a CA, except for the first time.
- Requirement R.2.3.8 - it is a natural consequence of R.2.3.1. Entities, which are known directly, are trusted more than those, which are known indirectly. Therefore sequences of certificates needed for public key verification should be as short as possible.
- Requirement R.2.3.9 - it is hard to prevent reply without an incremental counter or time-stamps, where time stamps are more general solution. It is also hard to reconstruct successful attacks without a time frame.

II.B.2. Prevention of threats

The basis for the issues given in this subsection is taken from the threats analysis written by M. Roe [8].

Loss of Confidentiality

Loss of confidentiality may occur:

a. when the user' s local key storage is compromised;
b. when a private key is intercepted at transmission between a User Agent and a local storage unit;
c. when the key generation process is compromised (bad keys are generated);
d. when a malicious CA disables security mechanisms.

The prevention of the threats **a, c** and **d** above is achieved, if the requirement R.2.3.3 in the subsection II.B.1 is met. Threat **b** is a local matter and thus out of the scope of this paper.

Modification of Data

Possible threats are:

a. modification of certificate contents during transmission;
b. modification of stored certificates;
c. modification of security attributes prior to being packaged in a certificate (made by user, attacker, or malicious CA).

Threat **a** is prevented by the nature of a certificate (signature) itself. It is very easy to check the digital signature, still it is computationally very hard to forge it. The same holds true for threat **b**, where R.2.3.4 additionally prevents this threat. Regarding threat **c**, a user is able to modify only his/her public or private key, which a CA would immediately note because of R.2.3.7. An attack during transmission is prevented by R.2.3.3. And finally, malicious CA operation can be easily detected by R.2.3.4. If the ISSUER field is forged then this can be noted as the certificate is made public via an electronic database and everybody can check that the public component of a falsified UN does not verify digital signature (the same holds true for algorithm identifier fields). If the subject field is forged, such certificate cannot be entered into an appropriate UN entry of a global database (R.2.3.5). If version and serial number fields are forged then this is identified, as all the certificates issued by a CA are publicly accessible via an electronic database. Therefore automatic checking procedures can be applied.

Masquerade

Masquerade occurs, when an entity pretends to be a different entity:

a. masquerade of a user requesting a certificate,
b. masquerade of a CA issuing a certificate,
c. masquerade of a CA during cross-certification.

Threat **a** is prevented by R.2.3.3, because a user has to physically appear at a CA (at least for the first time) and can be identified. Threat **b** is prevented by R.2.3.4, as the digital signature can be checked with a public key of an entity that a rogue CA pretends to be. The same holds true for threat **c**.

False Repudiation

This occurs when one entity denies sending or receiving information in one of the following cases:

a. repudiation by a user of having requested a certificate,
b. repudiation by a user of having received a secret key,
c. repudiation by a user of having requested the revocation of a certificate,
d. repudiation by a CA of having issued a certificate,
e. repudiation by a CA of having revoked a certificate,
f. repudiation by a CA of having requested a cross-certificate.

Threats **a** and **f** are prevented by R.2.3.5 - if the user (CA) has not requested a certificate (cross-certificate), he/she will not store it in the database. Threat **b** is prevented by R.2.3.3. Regarding threat **c**, assume that a private key has been compromised. There will be definitely no reactions, as long as a legitimate user does not recognise this fact. Then the user simply sends an e-mail request to a CA and asks for revocation. The CA is always in a position to check the digital signature of the message. It is very unlikely that such an action will be undertaken by another person who is not entitled to use the compromised private component. But if this person sent a request, it would be even better - a CA could again check the digital signature and revoke a certificate. This way, further false use of a compromised key is stopped. The legitimate user is informed about this and generates another key pair. Then the user encrypts a message with a CA' s public key and sends a prototype certificate to the CA(R.2.3.7). The CA signs the key and sends a certificate back to the user, who puts it in a directory. If the attacker also sends a prototype certificate with another public key, the CA will immediately recognise that two certificates are issued for the same UN with different public keys and would stop the process. If attacker does this before a legitimate user, he/she will have to put it in a database, which is very hard because of R.2.3.4. He/she should also know the password and the additional security attributes for the strong authentication with a data processor.

And this is a very interesting feature of the system that is compliant with requirements in this paper. Namely, such a system is self-regenerative in a sense that, when issuing a new certificate, a physical contact between a CA and a user is required only once, and that is when issuing certificate for the first time. All subsequent certificates can be issued over the network without a need for a physical contact.

Regarding threat **d**, repudiation by a CA of having issued a certificate is hindered by a fact that the certificate contains its UN and its digital signature, which can be verified by CA' s public key that is found in the CA' s entry in the database. The same holds true for repudiation of revoking a certificate, mentioned in threat **e**. The problem is, however, when a CA does not revoke a certificate according to the owner's request. To solve this problem, user has to be provided with a proof of delivery of his/her request to the CA.

Misuse of Privilege

Misuse of privilege means that a CA performs actions it was trusted never to do in one of the following cases:

a. a CA uses a user' s private key to forge signatures,
b. a CA uses a user' s private key to decrypt confidential information,
c. a CA uses incorrect certificates to subvert security mechanism,
d. a CA uses incorrect CRL to cause denial of service.

Threats **a** and **b** are prevented by R.2.3.3. Threat **c** is prevented by R.2.3.4, which makes the results of CA operations public and everybody is in a position to check these results. The only problems that occur in this case are time delays that are shortened to the minimum due to the public availability of results of CA' s operations. For instance, incorrect certificates cannot be harmful, as they are not effective until users put them in a database; users are always in a position to verify the certificate before storing them in the database.

Similar reasoning applies to **d**. As soon as a user is informed about the fact that a CA has revoked his/her certificate, he/she can insist on showing a revocation request to a third party. As a CA is not able to forge the digital signature (R.2.3.3), it has no proof of a revocation request (R.2.3.7) and can thus be charged. Time delays are certainly introduced this way, but they can not be avoided.

Exceeding Authority

It occurs when CA performs unauthorised actions:

a. a CA issues a certificate for a member of an organisation over which it has no jurisdiction;
b. a CA revokes a certificate issued by another authority over which it has no jurisdiction.

Threat **a** is prevented by R.2.3.8, and threat **b** by R.2.3.6.

II.C. Database Requirements

To prevent masquerade, a requirement for a database is straightforward:

- *Every data processor has to have its own certificate to prevent masquerade, which should be signed by the TLCA, as a global and public database requires general confidence.*

R.2.4

So every communication with a data processor should be done using strong authentication to prevent masquerade of the system. Applying this requirement to X.500 a drawback, already mentioned in II.A, is introduced.

Requirement R.2.4 should be fulfilled also by alternatives, e.g., WHOIS++ [2] and front-end systems, e.g., SOLO [3].

II.D. Guidelines for a CA Architecture

Users will certainly have to have multiple key pairs because of conclusion from subsection II.B.1 (R.2.3.2). It is a common reasoning that when one public component is signed by a certain CA, it means that this CA allows the user to use the key according to the set of rules that express the CA's policy. So if the user is allowed to have signed the same public component by another CA with a different set of rules, this means that the key pair effectively embodies the union of two sets of rules. This is not a serious problem, as the user' s entry would contain two certificates and according to R.2.3.4 both superior CAs could see if CA in question violates any of their policies.

It should be emphasised that *it is a certificate, which embodies a CA policy and not a key*. Again, policy means that a certain person, whose role is identified by a UN, is allowed to do certain tasks, identified by CA' s rules. Therefore, if a CA has not put restrictions on a public key (its length, for example), user may have the same key in multiple certificates, signed by different CAs.

A practice of having many certificates with equal ISSUER fields and equal SUBJECT fields (be it with the same or different key pairs) should be avoided. It is obvious that the certificates with same ISSUER/SUBJECT fields and same keys are completely obsolete. In case of different keys the following should be considered. When a CA signs a key, this means that it allows the user to sign the data with that key according to its policy, which means that one key pair is usually enough and others just add complexity to certificate management. However, if one entity insists on having multiple key-pairs per UN, this is acceptable as long as both are signed by the same CA, because of the threat described below.

Assume a case of a CA having two key pairs. Suppose a CA signs certificates with one key and the CRL with another. This means that one key (used to sign user' s key) has to be verified against one sequence of certificates, and the other one for CRLs against another sequence of certificates. When verifying each of these sequences, the same problem with two additional sequences for each starting sequence of certificates may appear again. Even worse, this would be in a hierarchy, where ISSUER and SUBJECT can be on the same hierarchical level, where this could result in loops and infinite procedures. Thus we can conclude:

- *Multiple key pairs per UN are allowed only if their certificates have the same ISSUER field OR if each CRL is signed by all keys.*

R.2.5

Note that requirements, given so far, support incorporation of non-hierarchical systems (like PGP[11]) in existing global hierarchy. When such community will be incorporated, it should preserve all internal functionality, otherwise users might refuse to use it. This means that in a global context certificates in question will be of the same hierarchical level. Corollary:

1. *It is a certificate that embodies a CA' s policy. Therefore it should be possible to get information about the certificate' s issuing policy.*
2. *Users may have many certificates because of the different roles they have in their life, where the various roles will be identified by different UN in the SUBJECT field of certificates.*
3. *Except in case of R.2.5, users (CAs) may have only one key pair per UN. However, they are allowed to have multiple certificates for the same SUBJECT UN, but with different ISSUER UNs. This means that users may have the same key pair signed by different CAs, if CAs have not put specific restrictions on the keys.*

R.2.6

It is important to notice that R.2.6 rules do not prohibit prototype certificates, where one entity creates a self-signed certificate. Thus revocation procedures as described in II.B.2. IV can be carried out via a database. Moreover, the top-level CA should have its self-signed certificate stored in a database by default. This is far more convenient than publishing its public key in newspapers, for example. A prototype certificate in an electronic database that fulfils R.2.3.4, R.2.3.5 and R.2.4 has the same level of trust as those published in a newspaper, but it is already in electronic form and can be used directly without manual retyping, which is time consuming and error-prone.

Careful readers might have noticed that in this paper we have assumed properties, which can easily be adopted by the 84 version of X.509 certificate. Thus building a certification infrastructure can be accomplished by the most widespread systems that actually support only 84 certificate. *To specify a policy, a separate field can be used in a secured database entry.*

To avoid situations where more certificates with the same ISSUER and SUBJECT field would appear and would thus complicate verification processes (as all certificates in such an entry should be checked), the following could be required:

- *Users (and CAs) may not have two or more certificates with the same ISSUER and SUBJECT fields, which means that one CA is allowed to sign only one certificate per UN.* R.2.7

In case of a 93 certificate, the policy could be put into a new attribute, while the other additional field could be used for global unique identification of certificate. This would make R.2.7 obsolete. As we are catering for the set of rules, which could be immediately used with existing systems, R.2.7 is proposed. Note that it does not contradict with R.2.5, but it merely restricts it to easier use the oldest version of X.509(88) specification.

At the end it should be emphasised that all other issues, which are not explicitly explained or addressed, should be interpreted according existing standards [4 , 5, 6, 7].

III. Conclusions

In this paper the importance of general properties of a public key certification infrastructure is pointed out and a general set of rules is given. The approach is based on a policy as a starting point to define hierarchies. Reference requirements are given that are suitable for small and global environments to preserve their openness and scalability. Local procedures are not imposed, which can be therefore completely defined locally and can thus accommodate any policy requirements. Furthermore, the requirements for a global database are very loose and can be met by various implementations.

It is almost a fact that certificates will be heavily extended with additional data fields [1]. The approach in this paper is general enough to accommodate these additional possible attributes in another way. It can be seen also as a complementary part of version 3. For example, one of the advantages of V3 is provision of users with more user-friendly data (like RFC 822 e-mail address of the SUBJECT), which is in line with UBN/UAN concept in the paper. However, it should be noted that automatic verification procedures could be developed without these extensions, using just 93 extensions. Two additional fields in a 93 certificate give a possibility for global unique identification and a verification path of a certificate and a policy specification.

The main open problem is the formalisation of *security policy*, which is just descriptively mentioned in [5] and PEM RFCs [7], but is vital for operational global infrastructure, especially to enable automatic verification procedures.

IV. References

[1] Ford W., "Public Key Cryptography for the Financial Services Industry: Extensions to Public Key Certificates", ANSI Working Draft X9.xx-1994, November 1994.

[2] Gargano J., Weiss K., "Whois and Network Information Look-up Service-WHOIS++", RFC in preparation.

[3] Huitema C. et al., "Simple Object Look-up Protocol ", Internet Draft, October 1994.

[4] International Standard Organization, "Information technology - OSI-The Directory", ISO/IEC 9594 1 through 8, 1992.

[5] International Standard Organization, "Information Processing Systems, OSI Reference Model - Security Architecture", ISO 7498-2, July 1988.

[6] ITU-T, "The Directory: Authentication Framework", Recommendation X.509(E), Geneva, 1993.

[7] Kent S., "Privacy Enhancements for Internet Electronic Mail, Certificate Based Key Management", IETF RFC 1422, February 1993.

[8] Roe M., "Certification Authority Requirements", PASSWORD document R2.5, November 1992.

[9] Roe M., W. Schneider et al., “Service Requirements”, PASSWORD document R1.1, August 1992.

[10] Trček D. Jerman B.B., Pavešić N., “Security policy space definition and structuring”, Computer Standards & Interfaces, North Holland Elsevier Science B.V., Amsterdam 1995 (to be published).

[11] Zimmerman P., "Pretty Good Privacy Users' s Guide", Version 2.0, September 1992.

7

A Certificate-Based Authorization Model,

Rich Ankney - Fisher International

A Certificate-Based Authorization Model

Rich Ankney
Fischer International
September 25, 1995

To date, major corporations and banks have declined to invest in public key technologies. While much of this reluctance is due to the lack of an established certification infrastructure, there are also a lack of well-defined risk models and auditing standards, and uncertainties regarding legality and liability issues.

Existing models for wire transfer (UCC 4A) and credit cards (Reg E) place all the liability on one party. Such models are not appropriate for general electronic commerce.

1 Requirements

No one disputes that paper is a bothersome anachronism in the electronic world, or that verifying pen-and-ink signatures is costly and error-prone. But at least with paper, one retains the basic "contextual controls" of document preparation and physical delivery. On a digitally signed electronic document, on the other hand, there is nothing but the encoded signature. All time, place and manner controls are absent, so nothing distinguishes a valid user signature from one produced by anyone else who somehow obtained access to their private key. It would not take too many multi-million (or multi-billion) dollar losses to erase all the savings produced by this new technology.

Serious investments to commercialize digital signatures will occur only after leading national auditing and legal experts are convinced that these systems contain adequate security controls to warrant reliance in mainstream intra- and inter-corporate business transactions, typically in the $10,000 to $10 million range.

The authorization model being developed by ANSI X9F meets the following requirements. While these restrictions seem complex, they merely reflect ordinary business procedures made explicit for purposes of machine verification.

1) Authorization is "offline"; all authorization information is conveyed in certificates. These authorization certificates are defined in detail below. The authorization process returns an indication of whether the subject document is acceptable given the restrictions and authorizations contained in the signers' certificates.

2) No human intervention is required in the authorization process. In complex and/or high volume environments, this is an absolute requirement to give these security controls credibility in the eyes of audit and legal experts.

3) Authorization may be based on
   - document contents,

- identity or role of the signer(s),
- intent (purpose) of the signer(s),
- transaction context, or
- any combination of the above.

4) A user may fill one or more roles in an organization, but only exercises a single role for a given document.

5) It must be possible to incorporate other documents by reference into a given document, and make authorization decisions based on the incorporated document(s).

6) It must be possible for a user to delegate portions of his/her authority to another user, on a short-term or long-term basis.

7) It must be possible to ensure that a single user cannot unilaterally authorize a document.

8) It must be possible to timestamp documents using trusted third parties. Such timestamps provide proof that the document was created at or before the time specified.

9) It must be possible to identify both the user and device which sign a document.

2 Authorization Model

2.1 Document Structure

This model requires the ability to attach multiple signatures to a document, as well as the ability to include per-signer information in the signature computation. [4] defines a useful format for this purpose.

This model does not constrain how a document or transaction is represented within a single system. However, a single unambiguous representation is required when signing or verifying a document, since both signer and verifier must perform their computation over the same representation. The *canonical* representation for a document consists of a set of attributes, extracted from the document, which are used in the authorization decision, and the actual document content.

2.2 Multiple Signatures

A document may be signed by one or more users. Each user's signature and other information is contained in a separate signature structure. Each signature structure contains an indication of the certificate needed to validate the signature, and a bit string containing the actual signature. Additionally, other information relevant to the particular signer would be included in an individual signature computation. This per-signer information, would be included in the signature computation, as *signature attributes*. A signature structure may also include per-signer information which is not signed, but merely appended to the signature structure (*unsigned* *attributes*). An important unsigned attribute is the *countersignature*. A countersignature is a signature on the signature structure in which it is found, rather than on the document itself. A countersignature thus provides proof of the order in which signatures were applied. Since the countersignature is itself a signature structure, it may itself contain countersignatures; this allows construction of arbitrarily long chains of countersignatures.

2.3 Signature Attributes

Useful signature attributes might include timestamps, location information, and signature purpose [5].

We can distinguish authorizing signatures, which must meet the restrictions specified in the signer's certificate, from other cosignatures by including an indication of the signature purpose in the data being signed, by including the signature purpose as a signature attribute. This *signature-purpose* attribute might have the values:

- *authorization:* authorization signature appropriate to the document,
- *cosignature:* authorization cosignature appropriate to the document; cosigner's certificate has sufficient authority to authorize the document, and
- *witness:* witness cosignature; cosigner's does not have sufficient authority to authorize the document.

Additionally, one might use a signature structure as a signed receipt on a transaction, in which case additional signature purposes for receipt confirmation might be defined. Application-specific purposes (e.g., medical record release) might also be defined.

A user may indicate the role s/he is acting in by including the role in the signature computation, as a (per-signer) signature attribute. The asserted role may be matched against a role certificate (or the user's attribute certificate) during verification. This approach eliminates the need for multiple users to share a public key certificate (and corresponding private key) assigned to the role.

2.4 Authorization Certificates

Authorization certificates are a particular type of attribute certificate [2]. An attribute certificate contains a reference to a "base" public key certificate, an issuer name, serial number, and validity period, and a set of attributes (in this case, attributes used for authorization). Such authorization attributes might include:

- *Transaction Limits:* These restrict the value of transactions (or other documents) which an entity may authorize. The user would be restricted to originate transactions up to a certain monetary limit, or between two boundaries.

- *Cosignature Requirements:* Additional signatures may be required for a given signature to be considered valid. Quorum and weighting mechanisms can be used to construct fairly elaborate checks and balances which explicitly govern the level of trust in each user [3]. The order of required signatures may also be specified. (Note that cosigners are specified via the digests of their public keys, to reduce the propagation of personal name information.)

The use of cosignatures allows an organization to effectively define checks and balances, and to explicitly specify the level of trust in a user. It also greatly reduces the risks from inadvertent compromise of a private key. It allows distribution of the authorization function over multiple locations and hardware platforms, with the resultant minimization of risk from access control failures on one of those platforms.

- *Document Types:* The user can be restricted to signing only such things as ordinary correspondence, purchase orders or other EDI transaction types, business contracts, specified financial instruments, etc., as defined by industry-wide policies. It will be desirable, for efficiency, to exclude large classes of transactions and documents.

- *Authorized Signatories:* An organization can indicate that only specific individuals can "sign for" the organization, similar to a standard "corporate resolution" to this effect. This might complement the document-type concept, as an additional control on signing of "corporate" document-types.

- *Geographical and Temporal Controls:* These restrict the locations and time periods from which transactions are considered valid. Use of a local trusted "timestamp notary," is assumed. Such a notary would append a trusted timestamp to the originator's signature on a document, and sign the result.
    - Time-of-day restrictions would normally coincide with the work-week of the user's locale.
    - Location information would be associated with the notary, so as to restrict access to a specific network segment, typically the user's assigned work area. The "granularity" of location controls would depend on the network architecture.

- *Age of Signature:* The document is not valid unless the signature is verified within some specified time period. For high-value transactions this period would be quite short, while for more normal transactions (especially those sent via store-and-forward systems such as X.400), a longer interval would be appropriate. The time of verification would be provided using a receipt signed by a trusted timestamp service, containing, at a minimum, the recipient's name and the signature from the original transaction.

- *Preapproved Counterparties:* Restrict an entity to dealing with some small set of partners, a common requirement in dial-up home banking systems. Another way of stating this is that "free-form transfers" are forbidden. Sponsors know that in case of an error, they stand a much better chance of getting their money back from a large, solvent, creditworthy party (such as Merrill Lynch), and a much worse chance with a small, unknown, unauthorized one (such as the user's criminal confederate). Separate certificates should be issued for each counterparty, to prevent a competitor from obtaining the user's customer list (other than himself) in a single certificate.

- *Delegation Controls:* It must be possible to limit the maximum authorizations an AA can specify when issuing an attribute certificate.

- *Confirm-To Requirement:* The signature is not valid unless the verifier sends a copy of the verified transaction to a third party (typically the user's sponsor or work supervisor) at a specified mail or network address, and (a) receives an accept or reject message, or (b) a specified time elapses. This is similar to a cosignature, but occurs *after* rather than before the transaction is sent. Such after-the-fact confirmation could be preferable in lower risk situations, where few transactions will be rejected and obtaining the cosignature in advance may be unduly burdensome.

A set of basic policies must be defined for use throughout the financial services industry (and other industries) to provide a well-defined, predictable level of service for the verification process. These policies would be agreed to on a multilateral basis by every participating firm, and could stipulate that certain of the restrictions and authorizations discussed in this section would always be deemed to be in effect, unless expressly provided otherwise.

2.5 Incorporation by Reference

A mechanism is needed to reference previously defined documents, such as contractual terms and conditions, in a document. Such documents are treated as part of the referencing document for authorization purposes. Such an incorporated document may imply constraints on the document types and other attribute values which are acceptable in the subject document. This is analogous to existing boilerplate in many types of documents. For example, government procurement contracts typically incorporate clauses from the Federal Acquisition Regulation (FAR), by reference. The contract will typically cite the clauses by number, title, and date, and include language expressly making the incorporated clauses part of the contract.

An important example of incorporation by reference is an industry-wide authorization policy. Industries or industry associations will typically develop such "industry policy" statements that establish minimum requirements for signature verification. All participants would physically sign these multilateral agreements, to ensure that all counterparties would be bound by the encoded restrictions. (Normally, authorization certificates should be required in all cases, and digital signatures would be deemed otherwise null and void in their absence.) These agreements would be cited using the incorporation by reference mechanism.

3 Authorization Certificate Hierarchy

This section proposes a hierarchy of authorization certificates which might be useful in the commercial environment. Issuers are referred to as *Attribute Authorities* (AAs), analogous to the CAs which issue public key certificates. There are two distinct levels of certificates; those issued within an organization, and those issued by a third party trusted by all participating organizations. The trusted third party would certify organizations, and the chain of trust would then extend from the third party down through certificates issued within the organization. This provides accountability down to an individual, rather than the "organizational" accountability of current systems (e.g., wire transfer).

Interworking Between Certification Domains

Rich Ankney
Fischer International
July 6, 1995

This paper discusses systems which construct and verify certification paths that span multiple certification domains and trust models. It assumes the reader is familiar with basic concepts of digital signatures and certificates, as presented in X.509 [1]. Such systems are required in order to fully realize the benefits of electronic commerce in both wholesale and consumer markets.

The paper first describes the requirements for such a certificate verification system, and the assumptions on the directory system used to store and retrieve certificates. Several existing trust models are then described, and an algorithm is presented which can construct a valid certification path between two entities, regardless of the model in use. Finally, the version 3 certificate extensions [2] which are used to constrain certification paths are examined, and the algorithm is enhanced to incorporate their use. [4,5] describe other work in this area.

1 Requirements

There are two major requirements for the type of system described above.

It must be possible to verify a certification path automatically. This provides the greatest flexibility and throughput, as well as enhanced security. In particular:

- The process must be automatable, perhaps in a self-contained module;
- No local trusted database is required (beyond a set of trusted public keys). If such a database is provided, it is only used for efficiency reasons;
- No real-time interaction with user is required (although some implementations may interact with the user in the event an unrecognized policy is encountered);

Additionally, interworking across a variety of environments is required:

- Multiple domains (certification policies, levels of assurance) must be supported;
- Multiple trust models must be supported;
- An entity (end user or CA) may have multiple certificates, issued by different CAs; and
- Any CA may, in general, cross certify any other CA.

2 Assumptions

This paper assumes certificates are stored in a distributed database which provides features similar to those of an X.500 directory. These features include:

- The database is structured hierarchically, with entries being accessed via a hierarchical name;
- Database servers can propagate requests which they cannot satisfy themselves (this includes returning a referral to a different server in response to a request);
- Each server holds a subtree of the total namespace. Replication and caching also allowed but not dealt with here (but see [3] for a discussion of caching issues);
- Different types of certificates are stored in different attributes (fields) of an entry. These include:
    - Forward certificates are issued by a CA to its subscribers. They are stored in the directory entry of the subscriber;
    - Reverse certificates are issued by a subscriber (which may be a CA) to its CA, as an aid in constructing certification paths. They are stored in the directory entry of the issuer;
    - Cross certificates are issued by a CA to another CA which the issuing CA trusts, to shorten the length of certification paths. This trust is gained by examining the subject CA's certification policy. They are stored in the directory entries of both the issuer and subject CAs, to ensure that a complete certification path can be built from either direction.

Using the existing X.509 attribute types, one can distinguish between certificate types as follows: Forward (subscriber) certificates are held in the **userCertificate** and **cACertificate** attributes; reverse certificates are held in the **cACertificate** attribute. These can be distinguished by whether the owner of the entry is the subject or issuer of the certificate (forward and reverse certificates respectively). Cross certificates are held in the appropriate (forward or reverse) component of a **crossCertificatePair** entry. If two CAs cross-certify each other, each of them holds both cross certificates, so both components of the crossCertificatePair would be present in each CA's directory entry.

3 Existing Trust Models

Although X.509 does not impose any particular structure on CAs, it may be reasonable to define a hierarchical structure in which each CA (in general) certifies only entities which are subordinate to it. Hence, we can construct a hierarchy of CAs, where the higher level CAs (perhaps banks) sign the certificates of the CAs beneath them (e.g., companies), etc. The lowest level of CAs sign user certificates. A variety of trust models have been defined, and some implementations make assumptions about the type of trust model in use. All of these trust models share the common property that a user need only trust one other public key in order to obtain and validate any other certificate.

3.1 Top Down Hierarchy

The Defense Message System uses a hierarchical trust model for certificate management. The model allows up to one million "root" CAs, each with one million subordinate CAs, each with up to one million subscribers. The distinguished name of the subject of a certificate must be subordinate to that of the issuer.

Internet Privacy Enhanced Mail (PEM) also uses a CA hierarchy, in which there is a single root, which certifies policy CAs (PCAs). Each PCA publishes its policy, which can be used by a user who is verifying a certificate within that PCA's domain. There may be any number of CAs in the hierarchy underneath a PCA; the distinguished name of the subject of all such certificates must be subordinate to that of the issuer. Any CA may only subscribe to one PCA.

The naming constraints in both of these models are rather restrictive (e.g., a user must have a different name for each CA he subscribes to). Additionally, the PCA concept will not allow for automatic certificate path validation if a large number of PCAs exist, since the user may have to make a judgment call on the acceptability of a new PCA policy at any time.

3.2 Bottom Up Hierarchy

The banking community prefers a model where CAs certify both their subordinates and their superior CA. Rather than receiving the public key of the root CA, a subscriber receives the public key of his/her own CA, and a certification path is constructed up the CA hierarchy to a common CA, then down to the certificate being validated. Assuming most communications is with entities close to each other in the hierarchy, this generally leads to a shorter certification path. Also, compromise of the root CA is not as catastrophic, since most certification paths don't go all the way up to the root.

3.3 Web of Trust

In Pretty Good Privacy (PGP), no certificates are required. Key management is done by face-to-face interchange of public keys (in which case each party signs the other's public key), or "introduction" by some entity that two parties both trust to sign their keys (effectively, a one-level CA hierarchy). Some PGP keyservers (analogous to CAs, in the sense that they sign the keys of many PGP users) have been developed, but the typically provide a very low level of assurance. The scalability of this approach is very doubtful, especially if the model must include other agencies, commercial entities such as EDI trading partners, and eventually individual taxpayers.

PGP does accommodate the concept of multiple signatures on a public key; a user might then trust a key signed by two individuals in which he has a minimal level of trust, or one individual in which he has a high level of trust. This model lacks complete semantics as to (a) what would be provided as input to the verification process to indicate these cosignature requirements, and (b) how this would extend to a complete certification path. Use of multiple signatures is not addressed further in this paper.

3.4 Composite Model

A structure is needed which supports multiple CA hierarchies of any depth. Forward, reverse and cross certificates are needed.

Cross certificates are used to issue certificates between hierarchies. Some likely scenarios might be:

- The PEM high-assurance PCA cross certifies the root of the banking hierarchy, indicating certificates from that hierarchy are trusted by PEM users under that PCA's policy.
- A CA in the banking hierarchy cross certifies the PEM high-assurance PCA, indicating all such PEM certificates are usable for subscribers to that CA (e.g., a particular application or particular class of users).
- A PGP user or keyserver cross certifies the PEM root CA, allowing PGP users to use PEM certificates. This assumes the PGP implementation can parse the PEM certificate format.
- The PEM low-assurance CA cross certifies a PEM keyserver, allowing use of PGP certificates by PEM low-assurance users (assuming the PEM implementation can parse PGP certificates).

The net result is a "forest" of CA hierarchies, with cross certificates between the hierarchies at various points. It is desirable that cross certificates involve CAs at a fairly high level in both hierarchies, to minimize the number of cross certificates and provide the maximum amount of interoperability.

4 Path Building Algorithm

The algorithm is quite straightforward, and lends itself well to parallel operation. Thus, an X.500 implementation which supports asynchronous operation could submit multiple requests in parallel. It relies on proper identification of cross vs. forward vs. reverse certificates, using the existing X.500 attribute types.

An entity may have any number of forward certificates from any number of CAs (although the number is expected to be fairly small). An entity will have one or more public keys of CAs which it trusts (again, a relatively small number).

As input, the algorithm receives a trusted public key (e.g., the user's CA or a root CA), along with the name of the trusted entity, and a target certificate. The algorithm builds a path up from the target (the forward chain), using forward certificates, and up from the trusted public key using reverse and cross certificates (the reverse chain), until a common point of trust is reached.

To build a path up from the target, one looks up the certificate(s) of the issuer, recursively, until the "root" of the hierarchy is reached or until a path is complete. The root has no forward certificates, although there may be reverse certificates (in another CA's entry) and cross certificates (in the root's entry) in which it is the subject. In general, only a single issuer certificate will be found, i.e., a public key of a CA is only present in one forward certificate, and possibly one or more cross certificates. The process results in a list or tree of potential certificates in the forward chain.

To build a path up from the trusted public key, one looks for reverse and cross certificates (recursively). This results in a list or tree of potential certificates in the reverse chain.

A path is complete when one of the certificates in the reverse list has, as its subject, the issuer of one of the certificates in the forward list. (If CAs have multiple keys and certificates, the reverse or cross certificate must of course certify the public key used in the forward chain.)
e

In the top down model, there is generally no reverse chain, and the path simply extends from root to target (possibly via cross certificates). In the bottom up model, the reverse chain extends from the user's CA to a common ancestor CA, where it intersects the forward chain (again, possibly with cross certificates in the chain).

5 Certification Path Constraints

The certificate extensions draft [2] describes constraints which may be placed in a CA's certificate, along with procedures for verifying a path (once it has been constructed). It also defines other useful extensions, e.g. a key ID which can be used to indicate which of a CA's certificates signed a particular subject certificate.

Constraint extensions include:

- Policy identifiers indicate which policy the certificate may be used with. The verifier may indicate that policy IDs must be explicitly present in all certificates in the path. This accommodates the situation where a CA issues a number of cross certificates to various CAs, with differing policies, but a verifier wants to restrict its acceptable policies to some subset of these. A CA certificate in the path may indicate that explicit policy IDs are required in all CA certificates in the remainder of the path. Note that knowledge of a policy is required by a CA, when issuing a certificate containing a policy identifier, and the verifier must indicate a set of acceptable policies, but the verification logic is not required to perform any processing beyond byte-by-byte comparison of policy identifiers. The existing model does not convey policy information (aside from that inferred from PEM PCAs), so a path would either be deemed valid on its face, or require user interaction (and familiarity with the relevant policies);

- Policy qualifiers are application data that may be used in the verification process (e.g., a reliance limit). These are (conceptually) passed to the application, and clearly their widespread use would hamper both interoperability and automatability;

- Policy mappings indicate which policies a CA deems equivalent for verification purposes. For example, a low assurance CA might consider its policy to be equivalent to a high assurance policy, but clearly not vice versa. Mapping may be prohibited by the verifier, or by some CA certificate within the path;

- A namespace constraint restricts the subjects of a CA's certificates to lie within a particular subtree of the X.500 namespace. This may apply only to certificates issued by this CA, or to all subsequent certificates in the path;

- Name subordination restricts the subject name in a certificate to be subordinate to either the issuer name, or to the issuer's superior's name. This applies to all further certificates in the path;

- The total length of the (remaining) certification path may be constrained. This limits trust in domains which are "distant" from the current domain in terms of cross certificates.

Clearly, these constraints can be checked when building the path, particularly for reverse and cross certificates, to prune unproductive paths from the forward and reverse lists. For example, certificates containing the wrong policy are clearly not worth pursuing, if explicit policy IDs are required. Similarly, if name subordination is in effect, entire portions of the directory namespace can be excluded from the search.

References

[1] ITU, "X.509: Directory Authentication Framework," 1993.

[2] ANSI X9F1, "Extensions to Public Key Certificates and CRLs," 1995 (draft).

[3] Cheung, T., "Management of PEM Public Key Certificates Using X.500 Directory Service: Some Problems and Solutions," in *Proceedings of the 1994 IEEE Symposium on Network and Distributed System Security*, February, 1994, pp. 35-42.

[4] Trcek, D., et al, "CA-Browsing System -- A Supporting Application for Global Security Services," in Proceedings of the 1994 IEEE Symposium on Network and Distributed System Security, February, 1994, pp. 123-128.

[5] Mendes, S. & C. Huitema, "A New Approach to the X.509 Framework: Allowing a Global Authentication Infrastructure without a Global Trust Models," in *Proceedings of the 1995 IEEE Symposium on Network and Distributed System Security*, February, 1995, pp. 172-189.

8

Towards Secure Enterprise Management

Steven B. Davis, Computer Sciences Corp.

Towards Secure Enterprise Management

Steven B. Davis
Computer Sciences Corporation

1. Introduction

Much of the discussion of information and information security technology focuses on providing “new” capabilities to the customer. If we change our perspective towards what business value these new capabilities can provide, we may find both holes in terms of capabilities and a different perspective on security requirements. The most basic point is that ***paper enterprise management is secure***. The very real challenge any electronic solution faces is to be at least as secure as a paper system and that it *may* add performance and security advantages. Where technology has made a significant change in business practices, we have seen an explosion of fraud because the mechanisms for defeating the systems are easy, quick, and very profitable (cellular phones as well as credit and calling cards being the most obvious cases). There are two main factors that have driven this problem: inadequate use of security technology and defeating human, as opposed to technical, decision-making. This paper will discuss these problems, as they relate to the Public Key Infrastructure, the metrics for addressing these security problems and introduce a model to deal with one of the neglected topics: enterprise security management. This is a particular problem because organizations are realizing that they present a complicated security “face” to the outside world as well as internally. To meet this challenge, an organization needs a mechanism to articulate and enforce this security policy and to communicate this policy to external and internal elements. Whatever solutions are proposed, they must provide value-added service to the customer or else public key go the way of the “paperless office”.

2. Metrics

As noted above, inadequate security technology and “human engineering” are the two ways to exploit automated information systems. The complexity of managing public key cryptography, in a system environment, aggravates this situation. For public key systems, the strength of the implementation and the authorization of signatures are the critical metrics.

2.1 Strength of Implementation

There are two factors that affect the implementation of public key systems: cryptographic strength and platform assurance.

2.1.1 Cryptographic Strength

The characteristics of a cryptographic algorithm are its mathematical strength and key size (or more simply effective key size). Cryptographic strength is conceptually the easiest facet of the public key problem, however, for many reasons, it has not been solved nor is a standard solution likely in the near future. This lack of standardization requires organization's to address algorithms, algorithm strength, and interoperability solutions (such as internal or external cryptographic gateways). Though encryption and key exchange issues are important, a standard solution for signatures will is critical for both government and commercial enterprises. Without a standard signature structure, it is unlikely true electronic business transactions will be possible.

2.1.2 Assurance

Though there is significant argument about encryption, assurance is a larger challenge to support signatures. For signatures to be credible for electronic commerce, there needs to be very high assurance that the person signing the material is actually signing it for more than integrity, but to enter into some contractual agreement. This implies that the platform and environment that the cryptography is carried out is *trustworthy*. It may also mean that different signatures should be used for content as opposed to personal signatures and perhaps a different, more trustworthy implementation is required for a personal signature.

2.2 Authenticity and Authorization

If the Public Key Infrastructure is at all successful, nearly every person and organization will have one or more set of public key credentials. The critical issue will be whether these credentials give a person the privileges that they wish to use. Given the range of roles and responsibilities within an organization where paper signature authorization is necessary today, it is unlikely that a public key certificate alone will be sufficient to capture the rich, dynamic organization policy and therefore some additional policy service and management capability will be necessary.

3. Model Overview

The proliferation of a variety of security mechanisms from firewalls to key management and cryptography and the drive away from standard solutions is driving the community towards standard "meta-solution" frameworks such as the Internet Security Association Key Management Protocol (ISAKMP) and the Simple Network Management Protocol version 2 (SNMPv2). ISAKMP provides a tool for managing specific cryptographic associations and SNMPv2 allows the management of a variety of devices, but neither tool explicitly is aware of the organization's policy nor the roles of individuals within the organization. Organizational structures and policies will likely change independently of the network architecture and individuals may move independently of the network. The staff responsible for maintaining network and cryptographic services operating today face the additional burden of determining and implementing a corporation's de facto security policy and procedures. It is quite conceivable for a manager to be allowed to review corporate financial data using his workstation in the office, but not be allowed to do the same at home - even if he has a cryptographic token.

This paper introduces a security management model built around the enterprise and describes a notional implementation architecture for this model. The Enterprise Security Management Model (ESMM) framework uses the enterprise as its basic element that can

have a variety of internal and external security relationships. This approach will accommodate both corporations, governmental groups, and individuals since the structure naturally supports a rich set of security relationships. Internal and external security services including firewalls, certification bodies, timestamps, and notaries all support the security policy of the enterprise and thus need to be included in the security management framework. For example, financial transactions over the Internet may need to be authenticated by a third party notary (which could be available from several vendors or agencies). The participants in the transaction would need to negotiate which of these service providers they would both find acceptable or accept a notary certified by some accreditation body. They would also need to agree on both the authentication protocol and encryption protocol that would be acceptable (even if a number are supported by both parties). The organizations, nationalities, size of the transaction, and other factors may affect what kind of exchange can be completed.
Today, these decisions are handled via off-line procedures or on an ad hoc basis meaning that performance and security suffer. A key portion of ESMM is the communication of relevant aspects of the enterprise's security policy so that human security management intervention is the exception and not the rule. The implementation approach is analogous to that used for SNMP - a combination of software agents, security policy databases, and a backbone management protocol. These components work together to allow the modification of the enterprise security policy, the management of security entities, and the notification of security events. The implementation approach allows security policy to be handled as a policy - not as a router connection table or a cryptographic Application Programming Interface (API). It will work with these elements, however, to ensure that security services match operational capabilities and vice versa.

3.1 External Entities

The basic element of the ESMM is the enterprise. An enterprise is an entity that creates and manages its own security policy and has a peer-to-peer relationship with other enterprises and other external entities. The security policy of an enterprise can be understood as a map of allowed services and information exchanges between external enterprises and internal elements. All information and services in an enterprise are either explicitly or implicitly labeled at the granularity required by the enterprise. Governments, corporations, other organizations, security servers (explained below), and individuals can all be external entities. The security policy agreements between these entities can range from the simple - a company notifying a business partner that an Electronic Data Interchange signature or privilege verification has failed - to the sophisticated - a teaming agreement between several corporations where a variety of information and services of different sensitivities are shared.

Security servers provide security services to other entities. Time stamps, notaries, cryptographic gateways, key/compromise managers, and accreditation bodies are all security services that are useful in an inter-enterprise environment. These entities provide security management support services: key/compromise managers notify enterprises of compromised equipment or individuals; accreditation managers certify that other services, such as notaries, are approved to provide their services; and cryptographic gateways provide trusted encryption and authentication translation services for interoperability.

3.2 Internal Entities

Enterprises, in turn, may consist of subenterprises, security servers, individuals, and information and services. Subenterprises would typically be an organizational component of an enterprise. In addition to the security servers that exist at the inter-enterprise level, firewalls, critical databases, and Enterprise Management Workstations could all be found within an enterprise. Information and services include financial transactions, labeled data (such as personnel information or classified documents), and decisions or votes.

3.3 Inter- and Intra- Enterprise Relationships

In addition to the allowed usage of services and information by individuals and groups, enterprise security management needs to address the interconnection of enterprise security policies. Components need to be able to communicate their ability to support the security policy so that routing, encryption, and other measures can be used to ensure policy compliance or the detection of portions of the system that cannot support the overall enterprise security policy.

3.4 Global Management Framework

National or international accreditors are needed to create, oversee, and delete enterprises and other entities to adjudicate transactions and establish a legal basis for electronic business. These entities do not need to control security policies of the enterprises within their domain, but principally verify their existence to other enterprises.

4. Implementation Discussion

As noted throughout this paper, the actual security policies and procedures of an enterprise implemented by their AISs are dynamic, and often complicated. In order to realize these policies and procedures, some method is needed to communicate and manage them and tie together the communication, application, and security management systems so they work together, but do not interfere with each other. This could be done through extensions to existing protocols and applications, such as ISAKMP or SNMPv2. Many of the entities in the ESMM would be represented by an Enterprise Security Management Agent (ESMA) that acts as an interface between the enterprise security policy and the lower level devices and services that need to support that policy. Thus, a change in an organization's security policy could be entered in at an ESM workstation and result in changes to firewalls, user applications, and security servers: the notification of the loss of a user's cryptographic token could be sent to a key/compromise manager, servers, and other workstations in the organization, as well as to relevant external entities to ensure that actions on behalf of the corporation by that individual would not continue to be permitted.

The implementation of the ESSM consists of three elements. Enterprise Security Management Agents (ESMAs), Security Management Databases (SMDBs), and the Enterprise Security Management Protocol (ESMP). Security aware enterprise elements will have an SMDB and ESMA. The SMDB contains the relevant representation of the enterprise security policy as well as information about its own security configuration. The ESMA communicates with Enterprise Security Management workstations as well as network management applications, cryptographic APIs, and other security relevant portions of the element.
The ESMP will need to provide a series of basic service to support a dynamic security policy: add, delete, or change the status and configuration of ESMAs and SMDBs; add, delete, or change the status of organizational and other entities within the existing policy framework; change the Enterprise Security Policy; support security queries from ESM workstations; and allow alarms and other "pushes" of critical information from ESMAs. There are several parts of the ESMP that are particularly relevant to the PKI: the specification of allowed keying relationships, approved security servers, and compromise notification "pushes" as seen in the two examples below. These examples walk through the automation of ordinary business processes for a contract between two firms and the promotion of a company employee.

4.1 External Case - Contract between two businesses

This case walks through the process from the conclusion of negotiation between two firms over a contract.

1. Company One submits the contract to Company Two for signature. Company One may go through the signature process described in the following steps, however, the verification process would be identical.
2. Officer from Company Two cryptographically signs the contract and returns it to Company One. In addition to the signature, the Officer will need to include an authorization letter, signed by Company Two, for signing the contract or a generic letter authorizing agreements up to some specified amount.
3. Company One confirms that Company Two is the real Company Two with the IRS or SEC. The IRS or SEC acts as a registry for all companies (within the US) and provides the company with the keying material so the company can create its own public key certificates as well as manage its own security policy.
4. Company One verifies with Company Two that the Officer is Authorized to sign the contract. This could be done either via the authorization letter signed by the company or a query of its policy server requesting confirmation or a list of authorized company officers.
5. Company One verifies the Officer's signature. This is ensuring that the Officer, in fact, is empowered to enact the agreement.
6. Company Two carries out a similar process.

If a problem with the Officer (or his token) occurs, his company would have a record to notify other companies that might work with him. Also, a revalidation of some signatures may need to be done.

4.2 Internal Case - Promotion of a company employee

This case walks through the promotion of a company employee from the time the promotion decision is made through its implementation.

1. Manager submits and signs appropriate documentation.
2. Appropriate Senior Managers concur and sign. The level and amount of the promotion may require different signatures or sequences of signatures. It may also be possible for certain very Senior Managers to bypass intermediate steps and directly authorize promotions.
3. Accounting processes the request. The whole "Human Resources" process could be automated and transactions passed directly to the financial organization where verification of signatures and authorizations could be done. Accounting may also verify that funds are available and other policy factors are addressed.
4. The promotion is implemented.

5. Conclusion

This paper has posed a basic framework to allow the management of security objects at the level of organizations and individuals - where security requirements tie directly to functional services. If the concerns raised herein are addressed, credible use of electronic processing could provide quicker and more secure implementation of corporate policies. Some such structure is necessary to effectively support realistic security in the dynamic, heterogeneous security and operational environment facing organizations today.

9

Trust through Metering in the Public Key Infrastructure

Robert Cordery, Leon Pintsov and Monroe Weiant, Pitney Bowes, Inc.

Trust Through Metering in the Public Key Infrastructure

by

Robert Cordery, Leon Pintsov, Monroe Weiant

Pitney Bowes Incorporated

Shelton, CT

Abstract

We propose a Certificate Meter to improve trust in the public key infrastructure (PKI) security services provided by certificates issued by a Certificate Authority (CA). Electronic commerce information security requirements include confidentiality, authenticity, assurance, and integrity. The PKI should provide services to meet these requirements based on the concepts of public key cryptography, certificates, and CAs. Use of a Certificate Meter addresses two PKI issues: The Certificate Authority is efficiently paid for its services; and the parties using a certificate in a transaction are provided with assurance that the corresponding private key is secure. The Certificate Meter embodies concepts already implemented in new Postage Meters -- a secure cryptographic processor that has the ability to store and charge funds for use of its cryptographic keys. This relationship of the Certificate Meter and Postage Meter concepts leads to an especially appealing implementation in which both meters are combined in a single secure processor, and the Postal Service acts as the Certificate Authority.

Background

Public Key Infrastructure

The Public Key Infrastructure provides information security services to meet the security requirements of parties to an electronic commerce transaction. The parties exchange information during a transaction and wish the exchange to have similar security properties as are achieved in a face-to-face business transaction. In particular, all parties to a transaction must be protected from misconduct by the other parties, and from eavesdroppers and impersonators. The security requirements include:

1. Confidentiality: Some transaction data must remain secret from outside parties.
2. Integrity: Each party must be able to verify that messages in the transaction have not been modified or replaced.
3. Authentication: Each party must be able to verify that the other party is indeed who they claim to be, and are authorized to perform their part of the transaction.
4. Non-repudiation: No party can deny that they sent the messages in the transaction.

5. Assurance: Each party must be assured that the PKI, the Certificate Authority and the other parties to the transaction are performing their tasks in a way consistent with that party's security requirements.

Many security services are provided by public key certificates signed by a Certificate Authority. The certificate provides evidence that the public key in the certificate was presented to the Certificate Authority by the entity identified in the certificate. However, several issues need to be addressed.

The certificate says nothing about the security of the private key. Each party must trust the other parties to maintain the secrecy of their private keys. This is a critical requirement of the PKI for business transactions. The users of the PKI have no basis for this trust. The private key is usually accessible on the computer system that is providing the communication services for the transaction, and thus can be accessible to hackers. Even if the private key is protected, usually through a password, it represents a weak link in the process.

Users of the PKI, who suspect compromise of their private key, are supposed to send a message to the Certificate Authority to revoke that certificate. Unfortunately, only users with the most tightly controlled computer security are likely to be aware of a private key compromise. Once a key has been compromised, the process of issuing a new certificate should require the user to again prove their identity to the Certificate Authority. This inconvenience means that users may prefer to assume that their key is not compromised, unless presented with overwhelming evidence.

What is the status of a message authenticated by a certificate that is revoked because of compromise of the private key? If the message is not received before the private key is compromised, then there is no reason to trust the authentication. In particular the date in the message is not trustworthy. If the message has not been stored securely from a date preceding the compromise, then the message is suspect.

Confidentiality is lost for any message previously encrypted with the public key associated with a compromised private key, or if the message is encrypted with a key communicated using the compromised key. Loss of confidentiality applies to all messages sent before or after the key is compromised.

Certificate Authority

The Certificate Authority operates under a security policy that specifies the method of publication and authentication of the Certificate Authority public key. The policy also specifies certificate issuance, revocation and the overall infrastructure security policy.

The Certificate Authority provides most of the trusted services of the PKI including:

1. Signing public key certificates that bind the authorization and identity of a user, to his public key. The signature is performed using the Certificate Authority private key.
2. Providing users with ability to access and verify certificates using the Certificate Authority public key.

3. Provide assurance of user authentication to the level claimed by the certificate, as specified by the policy governing the Certificate Authority. The security policy specifies procedures required to verify the user's identity and authorization.
4. Provide for certificate revocation and timely access to certificate revocation records.

While the Certificate Authority is trusted to provide these services, it generally assumes limited or no liability for security failures resulting from loss or leaking of private keys corresponding to certificates.

United States Postal Service and Electronic Commerce Services

The United States Postal Service is actively preparing to enter into electronic commerce as a Certificate Authority. The goal is to ensure that electronic commerce is afforded the same trust and security that hardcopy enjoys.

The USPS strategy is to enable business quality Electronic Commerce through a common ubiquitous, universal access, nationwide infrastructure endorsed by industry and government.

The USPS infrastructure will provide certificate management services including certificate issuance, identity verification, certificate distribution, status of certificate (existence, and revocation). The USPS also proposes electronic correspondence services including cryptographic postmark and seal, archive, and authentication. The postmark establishes the existence of the message in an unmodified state at a specific point in time.

The USPS Certificate Authority plans to charge a service fee for issuance of a certificate, and small fees for use of the service. As stated above, a Certificate Meter provides an efficient way to pay for use of the Certificate Authority services. This is especially true if the Certificate Authority is the Postal Service. A new IBIP meter is secure co-processors with the key management and payment infrastructures and capabilities to perform the function of a Certificate Meter.

Vendors and Consumers

During a transaction various parties place different levels of trust in each other. In particular the Certificate Authority is more trusted than the certificate holder.

A consumer and a vendor taking part in an electronic commerce transaction place different trust requirements on the transaction. The vendor wishes to be assured that the credit issuer will assume liability for a transaction, as is done now in credit card transactions. The customer wishes to be assured that the vendor is authentic and that credit and transaction information passed to the vendor will remain confidential. That is, the customer wishes to be assured that the vendor's private key is secure, while the vendor wishes to be sure that the credit issuer will cover the transaction. The vendor may therefore accept the signature of the customer after verifying with the credit issuer. The consumer needs assurance that the vendor's private key is secure. That assurance is usually explicitly excluded from the liability assumed by the Certificate Authority.

A good way to provide the consumer with the needed assurance is for the vendor to use an approved secure co-processor. The certificate can indicate that the certificate keys were generated within, and the private key retained within the secure co-processor. A certificate meter is a secure co-processor, that generates, hides, and controls use of a certificate private key.

Metering Postage

Many businesses world-wide have been using a secure funds-metering device for the past 75 years. The postage meter is a tamper-evident, periodically inspected device that maintains postage funds and accounts for their use in printing postage. Funds are deposited in postage meters either manually at the post office, or using a one time cryptographic code received by phone.

Postage by Phone® is a secure cryptography and telecommunications-based electronic postage payment system operated by Pitney Bowes Inc. Since 1979. Postage by Phone® provides an electronic commerce system for postage payment for more than a half million businesses. The experience gained from operating this secure payment system and dealing with users from all business segments can be effectively utilized in for operating a PKI.

USPS Information Based Indicia Program

This year the USPS announced the Information Based Indicia Program (IBIP). The goal of this program is to use digital printing technology and cryptography to provide added value and security in the postage metering process. The Vision Statement of the IBIP program is:

- Establish a new form of postage indicia that enhances the security of postage revenue and supports expanded Value Added products and services that are useable by all mailers.
- Improve the level of detail and type of information by which we manage USPS business and processing activities and enable more reliable and accurate performance measurement.

Meter Security

Today, postage meters are secure cryptographic co-processors. The advanced technology Postage Meters proposed in the IBIP program, and the most recent meters submitted to the USPS include many properties of interest to the PKI, including:

- Digital Meters are assigned to authorized businesses,
- Digital Meters are periodically inspected for presence, evidence of tampering, and correct operation,
- Digital Meters are secure cryptographic modules, that will be tested against a USPS version of FIPS 140-1,
- The USPS will maintain the ability to authenticate cryptographic messages from a Digital Meter.

Postage meters designed to meet this proposal have the security, cryptographic and communication features necessary to perform the functions of a certificate meter. A postage and certificate meter is a device that combines the capabilities of a postage meter and a certificate meter, providing significant advantage to the Postal Service Certificate Authority and to the user. .

The Certificate Meter

Concept

A Certificate Meter is a secure cryptographic device with secret information that allows secure communication with the Certificate Authority and capability to use and manage the private key of a certificate. The Certificate Meter includes metering and accounting capability. It provides convenient low cost payment of charges per use of a certificate private key. Postage metering and certificate metering functionality do not have to be combined. However, putting these functions together provides economic efficiency.

Advantages to the Postal Service or other Certificate Authority include:

- increased use of existing meter tracking infrastructure to provide assurance that the certificate meter is used by the authorized party,
- fewer devices to manage and inspect, and the legal right to inspect already in place,
- provides a secure communication channel using a unique secret key stored in the meter and known to the Certificate Authority to manage keys and certificates
- produce authenticated messages for Postal Service regarding the status and usage of the meter, thus providing additional security and assurance for postal funds and certificate authority payments.
- use of the postage and Certificate Meter for postage payment provides ongoing assurance to the Certificate Authority that the device is operating correctly, and has not been abused.
- Provides a convenient mechanism for the Certificate Authority to collect payment for its services.

Advantages to the user are:

- a single secure co-processor to validate payment of certificate use charges and postage reduces the number of secure devices to manage,
- provides a turnkey device to generate key pairs, and keep the private key secret,
- a single account to pay for certificate usage and postage,
- Postage and Certificate Meter can efficiently pay Certificate Authority charges for new certificates and for use of the certificate,
- secure installation and use of the private key corresponding to the certificate's public key,
- secure revocation of certificates.

Assurance of Private Key Security

Assurance that another user's private key has not been compromised can only be obtained by the use of a secure co-processor. The secure co-processor generates the public and private keys, produces a "new public key" certificate signed by a private key of the co-processor, and exports only the "new public key" certificate. The Certificate Authority can verify the certificate, and thus be assured that the private key is held inside the Certificate Meter.

Assurance that no one tampered with the Certificate Meter can be improved by requiring periodic inspection. If the owner of the Certificate Meter refuses inspection, then the certificates associated with that Certificate Meter will be revoked by the Certificate Authority.

Certificate Revocation

A user may decide to ask for a certificate to be revoked for several reasons:

- the conditions for use of the certificate may no longer hold,
- the user may suspect some misuse of the certificate,
- the certificate may be renewed as part of a standard security process.

The user can ask the Certificate Meter to sign a message requesting or confirming revocation of a certificate. That message is sent to the Certificate Authority, who thus has assurance that the request for certificate revocation is from a legitimate source and thus revokes the certificate. The Certificate Meter can erase the private key, thus removing any possibility of further use of that key.

The Certificate Authority can revoke a certificate while in secure communication with the Certificate Meter. The CA then has assurance that the private key of the certificate is lost and can not be reused.

Payment

The private key corresponding to the certificate's public key is never known outside the Certificate Meter. The meter controls the use of the private key and can assess charges for use or management of this key. The payment can be automatically debited from the funds stored in the meter, just as is done now with a postage meter. This simple means of paying for use of PKI provides a significant advantage of simplicity and security to the user and to the Certificate Authority.

Conclusion

The Certificate Meter provides useful functions for users of the Public Key Infrastructure and for the Certificate Authority. Users have assurance that their private keys and the private keys of other parties are secure. Users and the Certificate Authority have a convenient and secure method to pay for use of the Certificate Authority services. In particular, the integration of the postage metering system with a Postal Service based Certificate Authority provides a unique and consistent combination of security and convenience.

References

American National Standards Institute X9.30-199x, “Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry,” November 1994.

American National Standards Institute X9.31-199x, “Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry”.

International Telecommunication Union, CCITT Volume VIII - Fascicle VIII.8”Data Communications Networks Directory X.509 The Directory - Authentication Framework”, November, 1988.

“Public Key Infrastructure Study, Final Report”. Prepared by MITRE under contract to NIST, April 1994.

National Institute of Standards and Technology FIPS Publication 140-1: “Security requirements for cryptographic modules,” April 1982, re-issued January 1994.

R. Rivest, A. Shamir, L. Adelman. "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM 21:2, 120 - 126, February 1978.

R. Rivest. "RFC 1321: The MD5 message digest algorithm." Internet Activities Board. Online document available at URL ftp://nic.ddn.mil/rfc/rfc1321.txt, April 1992.

L.H.Stein, E.A. Stefferud, N.S. Borenstein, and M.T. Rose. "The Green commerce model." First Virtual Holdings Incorporated Technical Report. Online document available at URL http://www.fv.com/tech/green-model.html. October 1994.

S. Ketchpel, “Transaction Protection for Information Buyers and Sellers,” DAGS95: Electronic Publishing and the Information Superhighway, May 30--June 2, 1995.

Steve B. Cousins, Steven P. Ketchpel, Andreas Paepcke, Hector Garcia-Molina, Scott W. Hassan, Martin Röscheisen. "InterPay: Managing Multiple Payment Mechanisms in Digital Libraries" Submitted to Digital Libraries '95. On-line document available at URL http://www-diglib.stanford.edu/diglib/pub/reports/cousins-dl95.ps.

Semyon Dukach. "SNPP: A Simple Network Payment Protocol." MIT Laboratory for Computer Science Technical Report. Online document available at URL ftp://ana.lcs.mit.edu/pub/snpp/snpp-paper.ps. 1992.

10

Trusted Third Party Based Key Management Allowing Warranted Interception

Nigel Jefferies, Chris Mitchell, and Michael Walker, Royal Holloway, University of London

Trusted third party based key management allowing warranted interception

Nigel Jefferies, Chris Mitchell, Michael Walker

Information Security Group
Royal Holloway, University of London

ABSTRACT

In this paper we propose a novel solution to the problem of managing cryptographic keys for end-to-end encryption, in a way that meets legal requirements for warranted interception. Also included are a discussion of what might constitute a reasonable set of requirements for international provision of such services, as well as some analysis of the cryptographic properties of the scheme and how it might operate in practice.

I. INTRODUCTION

There has been much recent discussion on the question of how to meet users' requirements for security services, such as confidentiality and authentication. This has been largely prompted by the US government's Clipper proposals [**1**], as well as the increasing use of electronic means for transferring commercially sensitive data. On the one hand, users want the ability to communicate securely with other users, wherever they may be, and on the other hand, governments have requirements to intercept traffic in order to combat crime and protect national security. Clearly, for any scheme to be acceptable on a wide basis, it must provide the service users want, as well as meeting the legal requirements in the territories it serves.

To create a platform that can be used to provide user services, it is anticipated that solutions will be based on the use of trusted third parties (TTPs) from which users can obtain the necessary cryptographic keys with which to encrypt their data or make use of other security services. Law enforcement agencies' requirements will be focussed on the need to obtain the relevant keys from a TTP within their jurisdiction, so that they can decrypt precisely those communications that they are authorised to intercept.

In this paper we propose a novel mechanism that will enable TTPs to perform the dual rôle of providing users with key management services and providing law enforcement agencies with warranted access to a particular user's communications. Unlike other proposals, the mechanism allows users to update their keys according to their own internal security policies. We then list typical requirements for such a scheme, and consider how well the proposed mechanism meets these requirements. We conclude by considering possible variants of the basic method and also how other proposed schemes for using TTPs in this way relate to the described method.

This paper was produced as part of the UK DTI/EPSRC-funded LINK PCP project `Third-Generation Systems Security Studies.' Participants in this project are Vodafone Ltd, GPT Ltd and Royal Holloway, University of London.

II. THE MECHANISM

The proposed mechanism is based upon the Diffie-Hellman algorithm for key exchange [2]. In order to simplify our description, we consider the mechanism only in relation to one-way communication (such as e-mail). The adaptation of the scheme for two-way communication is very straightforward.

More specifically we present the mechanism in the context of a pair of users A and B, where A wishes to send B a confidential message and needs to be provided with a session key to protect it. We suppose that A and B have associated TTPs TA and TB respectively, where TA and TB are distinct.

Prior to use of the mechanism, TA and TB need to agree a number of parameters, and exchange certain information.

- Every pair of TTPs whose users wish to communicate securely must agree between them values g and p. These values may be different for each pair of communicating TTPs, and must have the usual properties required for operation of the Diffie-Hellman key exchange mechanism, namely that g must be a primitive element modulo p, where p is a large integer (satisfying certain properties). These values will need to be passed to any client users of TA and TB who wish to communicate securely with a client of the other TTP.
- Every pair of TTPs whose users wish to communicate securely must agree the use of a digital signature algorithm. They must also each choose their own signature key/verification key pair, and exchange verification keys in a reliable way. Any user B wishing to receive a message from a user A by TTP TA must be equipped with a copy of TA's verification key (presumably by their own TTP TB) in a reliable way.
- Every pair of TTPs whose users wish to communicate securely must agree a secret key $K(TA,TB)$ and a Diffie-Hellman key generating function f. This function f shall take as input the shared secret key and the name of any user, and generate for that user a private integer b satisfying $1 < b < p-1$ (which will be a 'private receive key' assigned to that user–see immediately below). The secret key $K(TA,TB)$ might itself be generated by a higher-level Diffie-Hellman exchange between the TTPs.

Given that B is to be provided with the means to receive a secure message from A, prior to use of the mechanism A and B need to be provided with certain cryptographic parameters by their respective TTPs.

- Using the function f, the secret key $K(TA,TB)$ and the name of B, both TA and TB generate a private integer b satisfying $1 < b < p-1$. This key is known as B's *private receive key*. The corresponding *public receive key* is set equal to g^b mod p. The private receive key b for B needs to be securely transferred from TB to B (like the other transfers discussed here, this can be performed 'off-line'). Note that B will be able to derive its public receive key from b simply by computing $g^b \bmod p$. Note also that this key can be used by B to receive secure messages from any user associated with TA; however, a different key pair will need to be generated if secure messages need to be received from users associated with another TTP.

- A must also be equipped with a copy of B's public receive key. This key can be computed by TA using f, the name of B, and the key $K(TA,TB)$, and then is transferred in a reliable way from TA to A.
- Finally, by some means (perhaps randomly), TA generates another Diffie-Hellman key pair for use by A to send secure messages to any user associated with TB. Hence TA generates for A a *private send key*, denoted a (where $1 < a < p-1$). The corresponding public send key is equal to $g^a \bmod p$.

In addition TA signs a copy of A's public send key concatenated with the name of A using its private signature key. The public and private send keys, together with the signature are then passed to A by some secure means.

Hence, prior to commencement of the mechanism, A possesses the following information:

- the private send key a for user A;
- a certificate for the public send key ($g^a \bmod p$) for user A, signed by TA;
- the public receive key ($g^b \bmod p$) for user B, and;
- the parameters g and p.

This information can be employed to generate a shared key $g^{ab} \bmod p$ for the encryption between A and B. This key can be used as a session key, or, even better, as a key-encryption key (KEK). The KEK would then be used to encrypt a suitable session key. This would facilitate the sending of email to multiple recipients, for instance, as well as allowing the use of a new key for each message. User A then sends the following information to user B:

- The message encrypted using the session key $g^{ab} \bmod p$
- The public send key $g^a \bmod p$ for user A (signed by TA)
- The public key $g^b \bmod p$ for user B

Once received, the public receive key $g^b \bmod p$ allows user B to find its corresponding private receive key b (there will be a different receive key for each TTP with whose users B communicates). User B can then generate the (secret) session key $g^{ab} \bmod p$ by operating on the received public key $g^a \bmod p$ for user A using its private key b, and thus can decrypt the received message.

Should there be a warrant for legal interception of this communication, an intercepting authority can retrieve the private key of one of the users from the associated trusted third party within its jurisdiction and use this in conjunction with the public key of the other user (which is transmitted along with the encrypted message) to find the session key for the encryption. There is no requirement for the intercepting authority to retrieve the private keys of both users.

III. A TYPICAL SET OF REQUIREMENTS ON A TRUSTED THIRD PARTY SCHEME

Clearly, the definition and agreement of a set of requirements acceptable across a broad set of countries is largely a political process. However, we can give a set of typical or likely requirements on which to base an analysis of the suitability of the proposed mechanism.

1. *Use of the scheme should provide visible benefits for the user.* The design and operation of the scheme means that the TTPs are capable of offering their services to users on a commercial basis. By signing up to a licensed TTP, the user will be able to communicate securely with every user of every TTP with whom his TTP has an agreement. The user would potentially be able to choose from a number of TTPs in his home country, thus increasing his trust in the TTP.
2. *The scheme should allow national and international operation.* The proposed scheme achieves this by ensuring that the intercepting authority can obtain the required keys from a TTP within its jurisdiction.
3. *Details of the scheme should be public.* This is achieved for the proposed scheme by the publication of this paper!
4. *The scheme should be based on well known techniques*, and Diffie-Hellman certainly qualifies.
5. *All forms of electronic communication should be supported.* The proposed scheme can easily be adapted to include two-way communication such as voice telephony.
6. *The scheme should be compatible with laws and regulations on interception, as well as on the use, export and sale of cryptographic mechanisms.* This matter is the subject of further study, but no problems have yet been identified.
7. *Access must be provided to the subject's incoming and outgoing communication, where a warrant is held.* This is clearly achieved for the proposed scheme, as the subject's TTP will be able to provide the appropriate send and receive private keys.
8. *The scheme should support a variety of encryption algorithms, in hardware and software.* As the proposed scheme deals solely with key management, any suitable encryption algorithm can be used, as long as it is available to the recipient and legitimate interceptors. The best way to achieve this may be to use a standard list of algorithms, such as the ISO register.
9. *An entity with a warrant should not be able to fabricate false evidence.* This is particularly applicable in countries where intercepted communications are admissible as evidence in court. The proposed scheme as it stands does not meet this requirement, but the provision of digital signatures as an additional service by the TTP will allow it to be met.
10. *Where possible, users should be able to update keys according to their own internal policies.* The proposed scheme allows a user to generate new send key pairs as often as wished (provided that he deposits them with his TTP) or have them generated by his TTP. The receive keys, which are generated deterministically based on the TTP shared key and the user's identity, are intended to be fairly permanent and change only if the TTPs' shared key or the user's identity changes.

11. *Abuse by either side should be detectable by the other.* We believe that this is the case for the proposed scheme, although abuse by collusion between the two sides may still be possible. The main disincentive to such abuse may be the 'shrink-wrapped' provision of the software, which we would expect to be bundled in with, say, an email system or other telecommunications software.

12. *Users should not have to communicate with TTPs other than their own.* The only communication required in the proposed scheme is with the user's own TTP.

13. *On-line communication between TTPs should not be required.* The independent generation of the receive keys in the proposed scheme means that no such communication is required for the proposed scheme.

IV. OPTIONS AND OTHER ISSUES

A. Trusting TTPs

The receiving party must trust the sending party's TTP, in order to verify the sending party's public key, and also because the sender's TTP can generate the receiver's private key. However, this trust only concerns communications between the receiver and senders belonging to that TTP. There will be a need for a certification hierarchy to identify a common point of trust for different TTPs.

B. The Choice of Values

There has been considerable discussion in the literature on the benefits of using a composite modulus for Diffie-Hellman, for instance. This, and other matters such as the length of the modulus p and the primitive element g, are beyond the scope of this paper.

C. Commercial Value

The proposed scheme relies entirely on its perceived value to users in order to be taken up. Service providers will want to recover the cost of setting up the service from their customers. Therefore the scheme must be able to provide value-added end-to-end services that users want. Further investigation is required to assess the level of demand for services such as:

- end-to-end encryption;
- end-to-end authentication;
- non-repudiation of sent message;
- message integrity.

Given that users will be paying for these services, they will expect a sufficient level of security. In the event of security failure with financial impact on the user, he will expect to be able to recover this, either via his insurers or from the organization running the TTP. This makes running a TTP a potentially expensive business, unless the financial risks run by the TTP can be adequately protected against. If TTPs are not commercially viable, then the scheme is a non-starter.

D. Combined two-way Session Key

The two-way version of the proposed scheme provides two keys for communication: one for each direction. These could be combined to form a single session key, or just one of the keys could be used. The advantages and disadvantages of this are a matter for further study.

E. Sharing Keys Between Trusted Third Parties

In an environment where commercial TTPs will be looking to offer additional services to their users, it is possible that some users will want the extra reassurance offered by having their keys shared between a number of independent TTPs. The proposed protocol is easily adaptable to provide this feature. For instance, the ideas of Micali [3] for adding secret sharing on top of existing schemes could be adopted.

V. OTHER PUBLISHED SCHEMES

A. The Goss Scheme

A scheme designed by Goss has been patented in the US [**4**]. In this scheme, a shared secret key is established by combining two Diffie-Hellman exponentiations using fixed and varying (per session) parameters. At first sight, this appears to correspond to the receive and send keys in the proposed scheme. However, the Goss scheme uses a universal modulus and primitive element. If x, x' are *A*'s fixed and variant keys, and y, y' are *B*'s, then the shared key is calculated as

$$\alpha^{xy'}\alpha^{x'y}$$

This could be viewed as a variant of the proposed two-way protocol whereby a universal modulus and primitive element are used and the two keys are combined by XOR-ing them.

B. Yacobi Scheme

This scheme [**5**] is almost identical to the Goss one, but uses a composite modulus and combines the session keys by modular multiplication rather than XOR-ing.

VI. REFERENCES

[1] National Institute of Standards and Technology, *FIPS Publication 185: Escrowed encryption standard.* February 1994.

[2] W. Diffie and M.E. Hellman, *New directions in cryptography.* IEEE Transactions in Information Theory **IT-22** (1976) 644-655.

[3] S. Micali, *Fair cryptosystems.* MIT Technical Report **MIT/LCS/TR-579.b**, November 1993.

[4] U.S. Patent 4956863, *Cryptographic method and apparatus for public key exchange with authentication.* Granted 11th September 1990.

[5] Y. Yacobi, *A key distribution paradox.* In *Advances in Cryptology - CRYPTO 90*, Springer-Verlag, Berlin (1991) pp. 268-273.

11

A Proposed Integrated International Security Infrastructure

Dennis K. Branstad, James M. Galvin, Trusted Information Systems

A Proposed
Integrated International Security Infrastructure

Dennis K. Branstad, Ph.D.
Trusted Information Systems, Inc.

James M. Galvin, Ph.D.
Trusted Information Systems, Inc.

Abstract

This paper hypothesizes the need for an integrated international security infrastructure that provides a number of security services needed to support international commerce. While the services could be independently developed and offered by independent service providers, there are economic reasons for integrating the services in a comprehensive architecture and offering them as a package. As with most technology-based applications, there are arguments for commonality and interoperability of the components of the infrastructure as well as arguments for local optimization of the services required by local users. There are numerous examples of technology-based applications (e.g., railroads, electrical power, telephone, television, cellular communications, and computer networks) where competing interests fostered the incompatibility of products. In some cases the users benefited from competing products until a market-driven "winner" was established. In many cases, the users were economically penalized if they picked an ultimate "loser." In some cases, the "best" technology was not the "winner," again as a loss for all users. This paper recommends the development of an integrated international security infrastructure that provides a minimal set of security services and supports a number of competing security technologies simultaneous with the goal of automatically selecting a "winner" based on the "best" technology and the widest user satisfaction.

Introduction

Computers date to the 50's, shared computers to the 60's, networks to the 70's, personal computers to the 80's, and international commerce applications to the 90's. Security has primarily been an afterthought in each area. Standards have been made and broken (bypassed, surpassed, or not passed) in nearly all areas. In many cases, the standards have been too early, too late, or too restrictive. In some cases, the standards encompass all competing interests and thereby serve none. This paper is not yet another call for standards but for a design (architecture, model, framework, ...) that can be used to provide a set of security support services that will be needed in the next decade and those beyond.

There are a number of technologies (automobiles, telephones, television receivers) that have little use without a supporting infrastructure (roads, switched communication networks, transmitters). There are great economic bases established by the manufacturer and sale of the products but even greater bases are established by servicing the products. In many cases, the costs of the infrastructure supporting the products are not charged directly to the users of the products (e.g., public roads, commercial television programming). In other cases, the costs of infrastructure services are paid directly and proportionately by the users (e.g., telephones).

The history of computer network services is varied. Within the U.S., local area services (connections, switches, cables) have been paid for by the using organization. However, the wide area services have been a combination of private, leased services and subsidized, quasi-public services, starting with the research community. Thus many users have become accustomed to "free" infrastructure services (i.e., services paid for by someone else).

Technology standards and economics are two primary forces that must be addressed in an international security infrastructure. Just as consumers are accustomed to purchasing telephones and paying monthly charges for the phone calls they make, so must international commerce consumers become accustomed to purchasing security products and paying charges for the services they consume. The goals of the workshop should include obtaining the broadest possible user base for a core set of ubiquitous security services and allocating the cost of providing the services on a fair and equitable basis.

In order to provide security services for international commerce, the security infrastructure should include support for:

- a set of security policies acceptable for international commerce applications ranging from low risk (e.g., less than $50) to medium risk ($50-$5000) up to high risk ($5,000-$500,000). Very high risk applications and risks resulting from cumulative failures may require special handling.

- various security technologies, including cryptographic algorithms, personal security tokens, and security mechanisms (e.g., 2-3 competitors in each major technical area based on user demand).

- on an optional basis, vital information repository applications and conflict resolution applications.

This support is necessary in order to be able to provide the following security services:

- digital signatures

- key management and distribution

- non-repudiation of transactions
- key escrow

Each item in the above two lists is described below.

Support

In order for a security infrastructure to be flexible enough to support a variety of applications in various contexts, it must not be constrained to enforce a single security policy and must not be required to make use of a single security technology. Each of these issues is discussed below. In addition, there will be a need in the future for vital information repositories and conflict resolution applications.

Security Policies

A security infrastructure will provide security services for a variety of international commerce applications ranging from low risk (e.g., less than $50) to medium risk ($50-$5000) up to high risk ($5,000-$500,000). Very high risk applications and risks resulting from cumulative failures may require special handling.

The rights and responsibilities of the infrastructure service providers and clients, and the expectations of each will vary according to the risk associated with the transaction. An infrastructure service provider that enforces a single security policy will limit the base of users from which it can draw its clients. Initially, a business will have a higher probability of success the larger its user base. Long-term, a business will have a higher probability of success if its growth is not limited to a market restricted to a relatively fixed size user base.

Security Technologies

While the security infrastructure service providers will probably be independent of, or only loosely related to, the technology providers, the latter are very important. Many areas of technology will be involved, including cryptographic algorithms, personal security tokens, cryptographic application program interfaces, communication protocols, and the applications themselves. Interoperability will be required between all the characteristics of sending and receiving applications as well as among the providers of support services for the applications.

For example, mobile users will require specialized applications and support services compatible with geographically distributed locations. Personal security tokens will be expected to function both correctly and transparently at whatever facility a user has chosen for information access. Mobile users of cellular telephones and users of credit cards have become accustomed to ubiquitous acceptance of personal equipment and offering of equipment. Thus, security services and their support services may not be able to be tied to location or equipment.

International standards have been adopted in some aspects of the needed technology. However, there are so many options and so many acceptable paths in international standards that a default strategy of establishing "profiles" of standards to guarantee compatibility and interoperability has been taken my many organizations. In essence, a group of organizations seeking commonality of functions and the ability to interoperate with large sets of diverse users establish a profile to use for procurement and operational purposes. Only equipment and services complying with the profile are obtained and used. A security infrastructure service provider must be prepared to support multiple profiles in order to best serve the broadest community of users.

Information Repositories and Conflict Resolution

Information is an asset to all organizations. The value of most information decreases with time, e.g., there is a "statue of limitations" assigned to many records that renders the information useless after it expires. A service that could reliably and irrefutably indicate when the information was created so that its expiration time could be reliably determined would facilitate a natural evolution to electronic media from the paper world of today. In addition, some information is valid for an indefinite period of time, e.g., records of public deeds. In this case, changes in technology may require that the information be re-deposited, for example, if the minimum recommended key length is increased. However, it may be important to record the history of deposits of the information. Obviously, the correct date and time is an essential component of an information repository and will be important consideration in conflict resolution.

An information repository could be used to store confidential information, also. The information could be encrypted before being deposited. All parties relevant to the information could share the necessary cryptographic key or it could be escrowed (see below). The information could be even be filed with an expiration time, at which time the information repository service provider would be able to destroy it. If necessary, perhaps during a conflict, the information could be retrieved and made available.

Services

Although there are many security services that could be supported by a security infrastructure, the following set have been found to be the most commonly requested and offer the most tangible benefit for the investment to begin the service offering.

Digital Signatures

There are many services related to digital signatures that could be offered. The most common is the creation of a digital certificate such that the authenticity of the subject named in the certificate and the integrity (protection against replacement) of the public key contained in the certificate is guaranteed. This service is somewhat equivalent of the notary public service in societies where many of the people could not write (i.e., sign their names). The notary public was granted authority by the state to identify a person through personal recognition and then "seal" the mark (e.g., X) of the person on a document, thereby giving it legal recognition. The electronic equivalent of this process has been given significant study both technically and legally, with a few states now giving statutory recognition to the process. A chain of authority needs to be established between a signer of a document anywhere in the world and a verifier (acceptor) of the document anywhere else in the world. While legally and technically foreboding, this can be done in a manner similar to the travelers check or credit card infrastructure. Financial liability is minimized by the service providers but residual liability is assumed by the providers (spreading the losses over the users of the service).

Key Management and Distribution

Key management and distribution on an international scale is more foreboding than providing a digital signature service for several reasons. However, in order to have confidentiality protection of sensitive communications, data encryption is necessary. This, in turn, requires the availability of "matched" keys (symmetric encryption/decryption algorithms use identical keys; asymmetric algorithms use a matched pair) at the point of encryption and the destination point of decryption. A secure key management system is needed within the security infrastructure to provide the necessary services.

Non-Repudiation of Transactions

Commerce is conducted among diverse organizations in distributed locations. Organizations that conduct continuing business with each other usually have contracts or agreements establishing transaction and liability limits. However, some organizations conduct many one-time transactions (usually individually and small in value) with individuals or organizations. Both types of business need protection from denial that the transaction transpired. In small value transactions, the attendant risk is small and protection may not be an integral part of the service. However, in large value transactions based on rapidly changing markets (e.g., stocks and commodities), the attendant risk is high and the protection is needed. Time stamps and Trusted Third Party seals are required as well as information repositories and conflict resolution services. These services could be offered by providers different than the more common security services.

Key Escrow

The concept of escrowing cryptographic keys has grown since the April, 1993, announcement of the proposed Escrowed Encryption Standard by the U.S. government. In that proposal, the "escrowed key" was given to one or more "escrow agents" which would relinquish the key under certain conditions to certain unnamed parties. The proposed government standard escrowed a cryptographic key unique to each hardware device implementing the standard with two government agencies (each holding a component of the key) who relinquished the key to government authorities when authorized by government rules. Commercial organizations found a little too much “government” in this proposal. However, some commercial organizations embarked on programs to develop commercial devices and services that would escrow the key (or keep a copy of the key available through a key recovery process) using commercial procedures for commercial purposes when authorized by the commercial users. Law enforcement could use its long standing legal rights to access facilities and information. A recent government initiative is calling for broader applicability of these programs and a harmonization of the needs of government and commercial industry. Such needs could be met by support of escrowing keys in the security infrastructure in addition to, or in conjunction with, the key management services. A great deal of legal and technical work must be done in this area for international encryption applications.

Recommendations

Specifically we recommend actions in several areas we believe are essential to establishing a security infrastructure. In addition, we include a discussion of the cost of establishing the infrastructure and the organizations most likely to succeed as service providers.

Action Areas

Participants in this workshop and similarly interested parties should work to establish technical standards and profiles for interoperability and mobility. An essential component of interoperability is the ability to replace a technology as a result of its evolution or having been succeeded.

Technical products should then be developed and offered for sale implementing these. Services should be developed by candidate service providers that match the requirements of these products, initially providing low cost services to the early users.

Governments should be a catalyst in both processes, assisting in the development of the standards, profiles, initial applications, and initial services. Registration of services should be performed, with full disclosure in areas such as risk assignment and long-term service assurance.

Governments should also take an active role in establishing strong certification chains of assurance for intra-government purposes, for government-private sector purposes, and for international purposes, both inter-government and commercial uses. Early establishment of these chains, perhaps including ones that support applications requiring very high assurance to ones supporting very low cost, ubiquitous needs, is mandatory for the fielding of the needed integrated international security infrastructure envisioned by these authors.

Cost Model

The most difficult part of establishing an international security infrastructure providing the needed security services is deciding who will pay for it. Within the U.S., costs are ultimately borne by one of two groups: the consumers or the government. Optimally, the government is tasked for providing for the common good and the taxpayers are tasked with providing the common funds. Pragmatically, special interest groups influence what projects are performed by the government and how they are paid for. Consumers are usually offered services in a competitive (in some sense) market but can be a captive consumer of public regulated services.

The second most difficult part is scheduling the development and dispersal of products and services. Products are needed before services are offered and few users want to procure products unless they are assured that the needed services will be available. Cars require gas stations which require cars. In general, products come first with the early users struggling to provide services for themselves or their closed user group.

An optimum cost model would be one where the product providers have formed associations with a service provider that can easily scale services to the number of users. The first products will cost more but initial services can be provided "free." Once product competition brings the cost of products down to where large numbers of consumers can enter the market, then profits often follow the "best" service provider. Cellular telephone service is an interesting model to study in this regard.

Early "portable" cellular phones cost about $1000, plus several hundred dollars to install in an automobile. They could be moved, but with difficulty. The price of service was not cheap but, even so, the revenues from the early users still not justify the installation and operation cost. However, today, a customer can get a truly portable cellular phone for a (subsidized) price of less than $50 (often free) with activation of a service (usually with a minimum 1-2 year duration and sometimes with a minimum monthly usage). The activation fee becomes the sales commission, the monthly minimum pays for the telephone, and the usage charge pays for the service. This may be a reasonable target model for the development of a security infrastructure. However, the model works for cellular telephones because of the regulations on radio frequency allocations and the assignment of service areas to a few (generally 2) service providers.

Service Providers

Providers of commercial services fall under many diverse rules and regulations, ranging from FCC regulating of communications carriers, federal licensing of commercial banks, state licensing of nurses, county licensing of plumbers, and local licensing of masseuses. To date, there are no ubiquitous rules and regulations of security service providers (special security services like inter-bank wire funds transfers are usually regulated as a part of organizational regulations).

Successful security service providers will probably satisfy the following criteria: well established in another service area; widely distributed throughout a geographic area; associated internationally with organizations providing services in other countries; large consumer of its own security services internally; no inherent technical conflict of interest. Organizations (organized in a priority decreasing with the number of times they have been considered for this role) typically meeting these criteria include: postal services, communications services, financial services, transportation services, medical services, military services, and organized religions. The latter are considered by some as good candidates for providing trusted third party services.

Background of Authors

Dr. Branstad was the first group leader in computer security at the National Institute of Standards and Technology (NIST, formerly the National Bureau of Standards). He started the group in 1973 when he came to NIST after working in the computer security research group at the National Security Agency. At NIST, he was instrumental in the development of the Data Encryption Standard and in adoption of the standard by the American National Standards Institute. He also aided several standards groups associated with the American Bankers Association. In 1983, Dr. Branstad was made an NIST Fellow and given the opportunity to perform independent research work in security. He initiated an activity which eventually lead to the Digital Signature Standard. During this process, he and Miles Smid identified the need for a security infrastructure to support computer applications needing cryptographic based integrity and confidentiality protection. They made several proposals for research and development to the management of NIST for the infrastructure. They worked with several organizations developing products and standards, addressing the needs of government users. With the assistance of Lynn McNulty, Associate Director of Computer Security, they initiated a Public Key Infrastructure study, conducted by the MITRE corporation and funded by a consortia of government agencies. Since then, Robert Rosenthal has taken a leadership role in furthering the availability of a public key infrastructure. Dr. Branstad worked with NIST and the Security Infrastructure Program Management Office following his retirement and joining Trusted Information Systems in 1994. He helped develop the project action plan designed to foster the availability of security support services for government applications.

James M. Galvin is a Senior Computer Scientist at Trusted Information Systems, Inc., Glenwood, MD. Dr. Galvin has almost 10 years experience in network security. He is the co-author and senior technical lead for the implementation of the MIME Object Security Services (MOSS) protocol. He was the senior technical lead for the Internet reference implementation of Privacy Enhanced Mail (PEM), MOSS' predecessor. The PEM security services are based on participation in the Internet certification hierarchy, a security infrastructure that has failed to be widely enabled and deployed. It is partly in response to this that MOSS was created, which depends only on public/private key pairs, although it supports certificates and their attendant infrastructure, and thus facilitates a bottom-up approach to the deployment of a security infrastructure. He is currently involved in Internet Infrastructure Protection issues, especially the Domain Name System and routing security, both in need of an international security infrastructure. He is active in the IETF where he is co-author of the SNMP Security Protocol, Chair of the DNS Security Working Group, and Executive Director of the Security Directorate of the Security Area.

12

The Need for Additional Research Into the Cost Considerations Inherent in the Creation and Operational Use of the Federal Public Key Infrastructure

Lynn McNulty, McNulty and Associates

THE NEED FOR ADDITIONAL RESEARCH INTO THE COST CONSIDERATIONS INHERENT IN THE CREATION AND OPERATIONAL USE OF THE FEDERAL PUBLIC KEY INFRASTRUCTURE

LYNN MCNULTY
MCNULTY AND ASSOCIATES

The 1994 Public Key Infrastructure Study prepared by the MITRE Corporation for the National Institute of Standards and Technology (NIST), contained one section that addressed the issue of cost. Since the issuance of that document little has been published about the cost that will be incurred in the development and operation of the Federal Public Key Infrastructure (FPKI). It is the intent of this paper to suggest that additional research will be required so that the entire range of government institutions that become involved in budgetary decision making will have sufficient information upon which to make sound resource allocation decisions. If solid, well supported cost information is not available, the implementation of important components of the FPKI may be delayed. Of equal importance is the fact that user agencies may not be able to anticipate adequately their initial and recurring costs. The ultimate objective of the FPKI is to deliver an efficient and cost effective service to the user agencies.

One of the surprising discussions that has not been held to date involves the costs inherent in the development and operation of the FPKI. In some respects this issue has been obscured by other technical, managerial and organizational issues such as: (1) Should the FPKI support both the DSS and RSA, or just the DSS? ; (2) What is the relationship of the Security Infrastructure Program Management Office and NIST's PKI activities? ; and (3) What is the proper allocation of the Innovation Funds between NIST and GSA. The author believes that since the issuance of the MITRE study little additional research has been done into the cost issues associated with the creation FPKI. Yet in this time of constrained federal budgetary resources, this very issue must be a constant consideration impacting all basic FPKI technical and managerial decisions.

The author has been disappointed by the lack of critical analysis and comments regarding Section 6, "Infrastructure Cost Analysis Results," of the MITRE study. While the material presented in the study is based upon the best information and analysis available at the time the report was prepared, there are some basic assumptions that have not been rigorously questioned or challenged. Yet the potential costs projected in Section 6 are rather significant, and in some respects overwhelming. For example the study asserts on page 6-6, that if one assumes that each federal employee is provided with a hardware implementation that contains the user's private keys and signing capability, the estimated start-up cost is in excess of one billion dollars. If these figures are accurate, will this funding be available over the next ten year? This is but one of the significant costs identified in the MITRE report. Another is the maintenance of the Certificate Revocation List (CRL), estimated to cost between \$610 million and \$780 million per year, assuming that all federal employees are enrolled in the FPKI. Clearly someone in the GAO or the Congress may in the future suggest that there are cheaper, less sophisticated alternatives to accomplish the government's fundamental business without resorting to digital signatures.

Therefore it is imperative that each stage in the development of the FPKI takes into account the issue of initial and annual operating costs. A solid business case for FPKI must be established at each step in the evolution of this important activity. Without this perspective the entire project may be vulnerable to the fiscal realities that confront government managers for the foreseeable future.

13

Issues Raised During Public Key Infrastructure (PKI) Study

Jisoo A. Geiter, MITRE Corp.

Issues Raised During Public Key Infrastructure (PKI) Study

Jisoo A. Geiter

geiter@mitre.org

703-883-6571

28 September 1995

MITRE

Overview

- Study background
- Purpose of Public Key Infrastructure (PKI)
- Purpose and scope of the Study
- Study methodology
- Recommended PKI architecture
- PKI internal issues
- PKI external interface issues

Study Background

- NIST, together with eight other Federal agencies, sponsored PKI Study
- Study period: Summer 1992 through Spring 1994
- Authors of the PKI Study Report
  - Shim Berkovits
  - Santosh Chokhani
  - Judith Furlong
  - Jisoo Geiter
  - Jonathan Guild

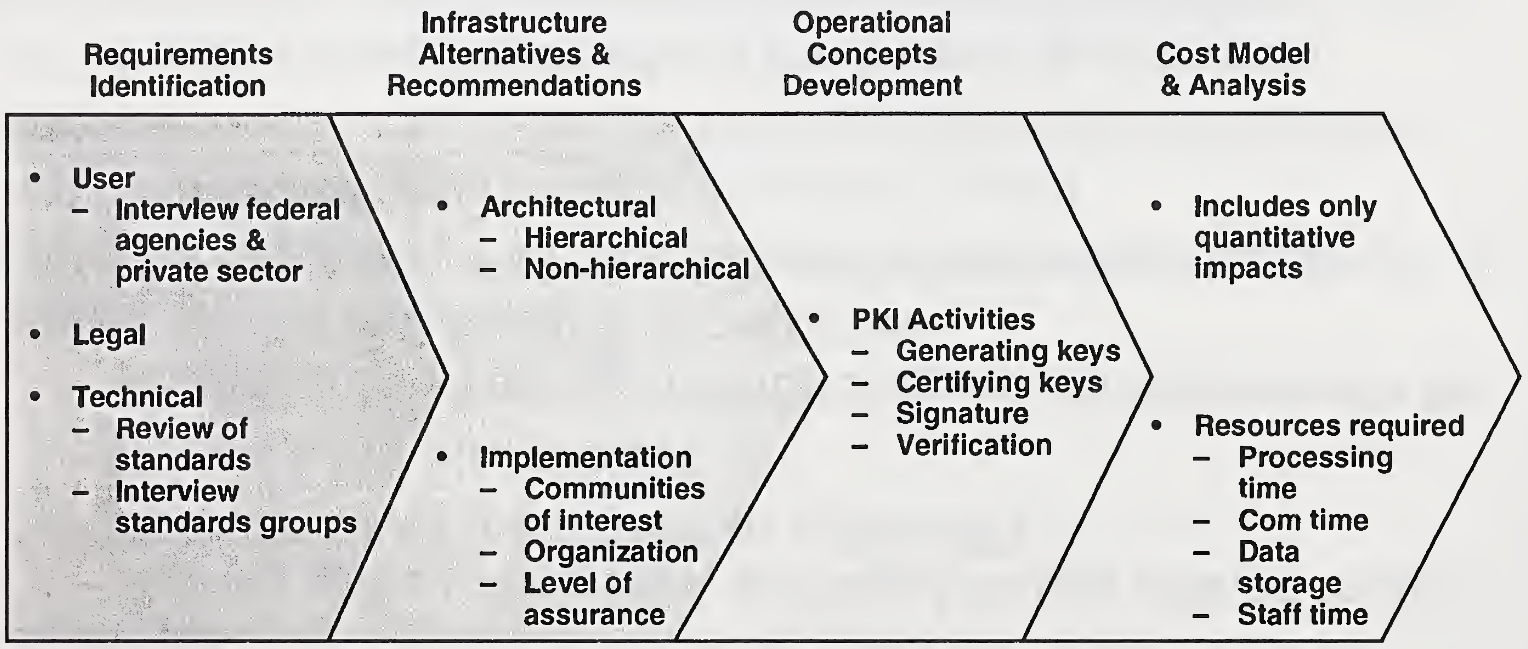
Purpose of Public Key Infrastructure (PKI)

- Generate public key certificates that bind the identity of users and their public keys in a secure manner
- Provide users, directly or indirectly, with easy access to the certificates of other users
- Provide users with easy access to circumstances (security policy) under which certificates were issued
- Provide users, directly or indirectly, timely announcements of certificate revocations

Purpose and Scope of the Study

- Purpose
  - Identify policy, technical, and legal issues related to the operation and management of the PKI
  - Develop PKI alternatives
  - Provide a PKI costing methodology to calculate costs for planning and budgeting purposes
  - Identify policy and legal issues in the use of digital signatures
- Scope
  - Focus on the PKI needs of the federal government
  - Consider other national and international entities
  - Plan for PKI interoperation with other algorithms and infrastructures

Study Methodology



Requirements Identification

- User
  - Interview federal agencies & private sector
- Legal
- Technical
  - Review of standards
  - Interview standards groups

Infrastructure Alternatives & Recommendations

- Architectural
  - Hierarchical
  - Non-hierarchical
- Implementation
  - Communities of interest
  - Organization
  - Level of assurance

Operational Concepts Development

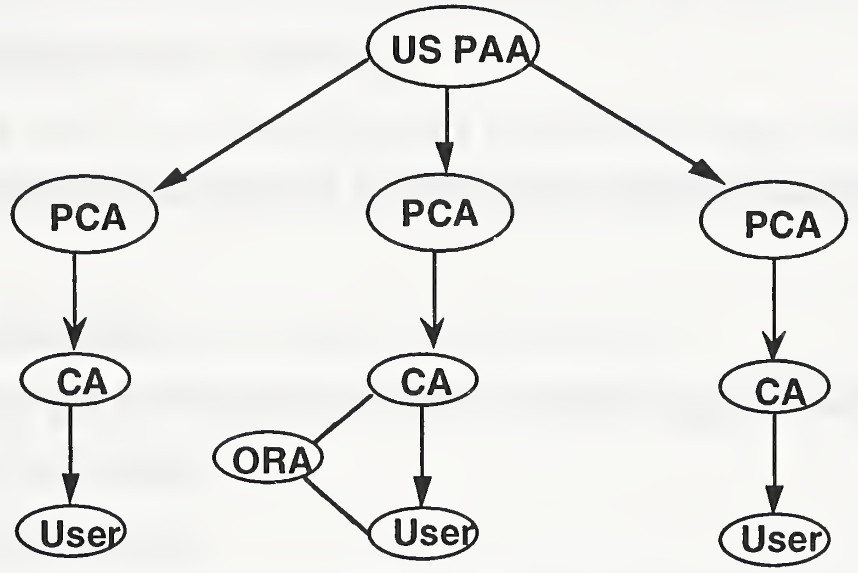
- PKI Activities
  - Generating keys
  - Certifying keys
  - Signature
  - Verification

Cost Model & Analysis

- Includes only quantitative impacts
- Resources required
  - Processing time
  - Com time
  - Data storage
  - Staff time

MITRE

Recommended PKI Architecture



- PAA: Policy Approving Authority. It creates the overall guidelines for the entire infrastructure.
- PCA: Policy Certification Authority. It establishes policy for its subtree.
- CA: Certification Authority. It certifies the public keys of users.
- ORA: Organizational Registration Authority. Its sole purpose is to vouch for the identity and affiliation of the user and register the user with its CA.

PKI Internal Issues

- PKI Architecture
  - Cross certification vs. strictly hierarchical
  - Depth of certification hierarchy
- Certification Revocation List (CRL)
  - How to distribute
    - Push vs. pull
  - Trust and its dependence on ready CRL availability
  - Liability issues
- Archiving
  - What federal archiving regulations apply to certificates and CRLs?
  - What needs to be saved?
    - The document, accompanying signature, certificate chain, and signature application version number?

PKI Internal Issues (Continued)

- Policy resolution
  - How do I know a certificate I received was created under an acceptable policy for a particular application?
- Legal
  - What I see is what I sign? Not more?
  - Does secured electronic transactions satisfy traditional legal indicia of reliability?
- Distinguished name
  - Should there be a name subordination rule ?
- Integrity of a key pair
  - How to ensure a presented public key has a correctly matching private key

PKI Internal Issues (Concluded)

- Directory service
  - Storing entire certificates or only references to them
  - Storing CRLs or removing the revoked certificates or both
- Confidentiality
  - Can DSA be used as a key exchange protocol?

PKI External Interface Issues

- Proof of Authorization
  - Within the PKI
    - Separate distinguished names, keys, certificates
  - Outside the PKI
    - Separate infrastructure for authorization certificates
- User Attributes
  - Within the PKI
    - Separate distinguished names, keys, certificates
  - Outside the PKI
    - Separate infrastructure for attribute certificates

MITRE

PKI External Interface Issues (Concluded)

- Interoperability
  - Occasional interoperating user may need support of a trusted verification service
  - Global interoperability
    - One global root vs. cross certification of the other infrastructure roots
- Time and date stamps
  - Time of submission
  - Time of receipt

14

Public Key Infrastructure Study - Cost Model

S. T. Kawamoto, MITRE Corp.

Public Key Infrastructure Study - Costing Model

S. T. Kawamoto

28 September 1995

MITRE

Public Key Infrastructure Study

- NIST sponsored study to identify and to evaluate architectural and implementation alternatives for the PKI
- Identify policy and legal issues in the use of digital signatures
- Identify policy, technical, and legal issues related to the operation and management of the PKI
- Develop PKI alternatives
- Provide a PKI costing methodology to calculate costs for planning and budgeting purposes

MITRE

Scope of Study

- Focus on the PKI needs of the federal government
- Consider other national and international entities
- Plan for PKI interoperation with other algorithms and infrastructures

Cost Model

- Developed to estimate a PKI-wide price tag for deployment and annual operations which can be used for planning and budgeting purposes
- Includes only costs which are attributable directly to the PKI
  - Also considers the costs associated with the additional demand the PKI places on the directory services and its supporting communications
  - Cost of doing business electronically excluded

Cost Model Implementation

- Implemented as a spreadsheet
- Model is flexible and can accommodate a variety of concepts of operation
- Resources necessary to complete each activity defined by the CONOPS are enumerated within model
- Model consists of six modules
  - One for each of four PKI entities: PCA, CA, ORA, and key generator (KG)
  - Additional modules for user and for directory
- Within each module, costs and frequencies for all needed resources are computed and a total cost is derived

MITRE

Costed CONOPS

- Generating, certifying and distributing keys
  - User goes to a centralized key generator and generates his own key pair
  - User presents himself and public key in person to CA or ORA
  - User's credentials delivered on PCMCIA card
- Signature
  - Hashing done on the computer
  - Message digest signed on the PCMCIA card
- Verification
  - Message digest computed
  - Sender's public key certificate obtained and verified
  - Signature verified on computer

Costed CONOPS (Concluded)

- Obtaining certificates
  - Responsibility of the receiver to calculate the certification path
  - Receiver obtains all certificates in the path between the sender and a common ancestor
    - Locally cached (certificates verified before caching)
    - Query the directory for certificates not cached
- Certificate revocation
  - CRLs may be pushed or pulled
  - User requests CRL for every certificate requested from the directory
  - All cached certificate checked periodically to see if they have been revoked

Costing Assumptions

- Certificate size: 500 bytes
- Size of CRL will be equal to 10% of current certificates
- CRL set is 51 bytes of header and 9 bytes per entry
- All end devices connected to a LAN and use of the LAN is free
- VAN costs are $0.02/KB (based on FTS 2000 pricing)
- 80% of the needed certificates are cached
- CRLs are distributed bi-weekly

MITRE

Cost Model Results

- Two sets of cost results presented in following slides
  - PKI cost
  - Total cost
    - Includes PKI cost along with costs associated with the users and with the directory
- Cost variances reflect cost scenarios used within the model
- Scenarios vary the following quantities
  - Number of users (1.5M to 3.0M)
  - Local versus remote communication
  - Number of messages per user per day (5 to 100)

MITRE

Representative Total Cost: User, Directory, and PKI

	Low	High
Parameters	1.5M emp	3M emp
	70% local	30% local
	5 mess./day	100 mess./day
Start Up Cost	$525M	$1083M
Operating Cost		
per year	$239.9M	$15433.8M
per user per year	$156.80	$5043.73
per message	$0.13	$0.26

MITRE

Analysis of Total Cost Estimates

- Start up costs
  - Single largest cost contributor is supplying a PCMCIA interface to every user
    - $1,031M when 100% of federal employees are supplied (95% of start up cost)
- Yearly Operating Costs
  - CRL distribution is the major cost driver
    - Appears in cost model as a communications cost which ranges from $136M to $14,654M (95% of representative high operating cost)
    - Yearly costs without CRL distribution range from $104M to $780M

Representative PKI Cost

	Low	High
Parameters	1.5M emp. 70% local 5 mess./day	3M emp. 30% local 100 mess./day
Start Up Cost	$9.5M	$9.5M
Operating Cost		
per year	$16.2M	$16.2M
per user per year	$10.59	$5.29
per message	$0.01	$0.00

MITRE