Investigation into the Features of Public Key Infrastructures

Michael Jarrett

April 18, 2004

Contents

1 Introduction

Since its inception, public key cryptography (PKC) has revolutionized the way people think about secure communication. At its simplest level, PKC allows the encryption of a (normally small) message with one key which can be decrypted by another related key. It suddenly granted people from all over the world the ability to authenticate themselves to each other, establish private communications, and digitally sign their messages, as well as a myriad of other uses.

However, the use of public key cryptography introduces a problem in that the public keys need to be distributed and managed. The techniques and structures used to do this is known collectively as a *public key infrastructure*.

Fundamentally, a public key infrastructure (PKI) is the system which manages public keys for a particular form of communication. It tends to be an umbrella term, covering a variety of tasks and challenges.

- The actual form of the public key; what does it store, and how does it store it.
- Obtaining the public key for an entity one wishes to communicate with.
- How to bind a public key to an identity, and fundamentally, what is the concept of an identity.
- Verifying that a public key actually represents the entity it purports to.
- Connecting an identity in a PKI with rights and privileges.
- Revoking a public key should the security of the key become compromised.
- Storage of the private key itself, including additional security measures to protect it, and distribution of the private key to third parties for archival or legal purposes.
- The legal infrastructure to deal with the liability (or protections from liability) of certificate authorities, and the legal tools to enforce the weight of a secure digital signature.

While many have visions of a single, global PKI, providing the keys for any form of communication, in practice PKIs tend to be created on a much smaller scale, and created with specific forms of communication in mind. In fact, often properties of the system one is attempting to secure can be used find solutions to the challenges of PKI which are fundamentally more difficult for a generic system.

1.1 Form of Report

Section 2 goes into greater detail about the challenges that a PKI needs to address. After discussing these issues, section 3 will look at how some PKIs have evolved in reality, and some of the systems proposed by researchers. Finally, section 4 will attempt to draw some conclusions from the current reality of implementations of PKI.

2 Issues

2.1 Certificate Format

A *certificate* is the concrete form of a public key, bound securely to a set of information describing attributes of that public key. The certificate, at a very minimum, must contain some encoding of the public key itself, the type of cryptography that was used to generate the key, and at least some identifying information describing what entity the public key is supposed to represent. Most certificate formats include a great deal more than this, including a signature from one or more parties verifying the key, attributes describing the lifetime of the key and where to check for it's revocation, and sometimes even details of what actions the key can authorize.

A great deal of a PKI's structure is actually determined by the form of the certificate. In fact, for some popular PKIs, such as the certificate format in the ITU X.509[1] standard, a variety of protocol-specific PKIs are actually referred to simply as "X.509 PKI". The certificate format is often critical in determining the attributes of a PKI, and also ultimately in its acceptance.

The most popular certificate format currently is X.509, version 3 (the most recent version is 4, but it has yet to see wide-scale adoption). This is a standard from the International Telecommunications Union Telecommunications Standardization Sector (ITU-T). The certificates include an issuer-unique serial number, a 'distinguished name'[1], the public key, and the signature of the issuer. In X.509, a certificate is generated for the entity by the issuer, normally referred to as a certificate authority (CA), who has the responsibility of verifying the identity (and perhaps other) information inside the certificate. Each certificate has exactly one issuer: a certificate which is not externally vouched for is signed with its own associated private key, in a technique known as a 'self-signature'.

The RSA Public Key Cryptography Standards[2] (PKCS) includes a couple of standards relating to certificates. PKCS 6 describes an extended certificate format; at its core, an X.509 certificate, but also including an extended set of attributes (defined in another standard, PKCS 9). The whole PKCS 6 certificate is signed by the issuer, allowing the extended attributes to be verified with a single public key operation.

Pretty Good Privacy[3] (PGP) is one of the few systems that does not use X.509 certificates. The differences between X.509 and PGP certificates have strong repercussions on

[1] Another part of the X.500-series of standards, a distinguished name is a set of fields including location, organization, and name information, and forms a globally-unique identity string.

how the resulting PKIs which use them turn out. PGP certificates have the added capability of multiple identities; for example, a work email address and a home email address can be bound to the same certificate. The other key feature is that each identity bound to the public key can be individually signed by multiple people. A certificate is issued by the holder of the public key, using a self-signature. As others verify the authenticity of the key binding to particular identities, they add their own signature to it.

X.509 (and anything that includes an X.509 certificate) can get very large. Often multiple certificates must be transferred to establish a chain to a known certificate. PGP is a bit more succinct, but a certificate can contain hundreds of signatures on several identities. Without some advance agreement on what to transfer, these certificates can also become very large, with the potential to grow even larger than X.509 certs.

These transfers, while reasonable on a desktop computer with a dedicated Internet connection, could become unreasonably large for a bandwidth-constrained wireless device where the user pays for airtime. This often becomes a critical issue in the wireless world, where any PKI with large certificates will be rejected.

2.2 Retrieving an Entity's Public Key

One of key responsibilities of a PKI is to actually get the public key (or more likely, a certificate containing one) to the party that wishes to communicate using it. While not technically complex in most cases, it is an essential aspect of any PKI.

Many systems, for example, Transport Layer Security (TLS), actually have each party transfer their public key certificate, and sometimes even an entire certificate chain, to the remote party before authentication. Normally, the trust model used in these systems allows for the verification of these certificates once they are received.

Transferring certificates during communication is quite effective in many systems since no central key storage server is required, but does not work in cases where there is no direct connection between the parties. The most important application today where this happens is email. In this case, the key must be obtained in some other manner.

Often keys can be stored on a centralized server. One good example of this is PGP, which defines a protocol for retrieving and uploading keys. Due to the trust model of PGP, no security is required at all; any valid certificate can be uploaded (merged) by anyone, and anyone can download a certificate. The downside of this is that there is little economic incentive to run such a server, as it is unlikely that people would be willing to pay for the right to download public keys.

Another approach, one often used in corporate environments, is to actually store the public key certificate with the identity itself, or in the directory where identities are stored. Often, a directory service can be used to store not only an email address but the public key corresponding to the owner. Alternatively, it would not be unreasonable to envision a mail server which could store the public keys for the users it hosts locally.

A variety of ad-hoc approaches are available. PGP keys, for example, can be exported into a file which can be hosted on a web page or sent as an email attachment. Microsoft's upcoming Longhorn operating system will also include public keys in their 'v-card' contact file format.

Identity-based encryption is an active research area, which would allow arbitrary strings to become a public key, with the private key being generated using the public key and a system parameter. This has the advantage of allowing any string, for example, a user's email address, to be used as a public key. The private key needs only be generated on demand, normally by a private key generator (PKG) with knowledge of some master secret for the system.

While identity-based encryption has many uses, it is considered by many as an intermediate step to encourage the use of cryptography and PKI. Such a system, used for email[4], would allow a sender to send encrypted email to a user without having to obtain and verify a certificate. In fact, the private key would not have to exist at all until the recipient wished to decrypt the message. Furthermore, other security features, for example, the ability to expire a key, can be added simply by concatenating the public key with additional strings such as the current date.

2.3 Trust Models

One area of PKIs that vary wildly is that of trust models. A good trust model is critical in making a PKI, and the communications that it facilitates, secure. A trust model describes who each participant in a system trusts. In terms of a PKI, this normally is used in terms of certificate signatures. Most models rely on establishing some chain of trust from some certificate you already know to be valid to the certificate you wish to verify. In some cases, the source of the certificate may also be used to establish this trust; in this case the certificate source is some form of trusted authority.

Certificates are normally either issued or signed, where the former describes a system where the certificate is generated by one central authority (a certificate authority, or CA), and the latter is normally generated by the holder of the private key, and then digital

signatures are applied to it. Sometimes, an issuer will include additional information in the signed certificate, most importantly being whether or not the signer trusts the holder of the private key for that certificate to also perform cerificate signatures.

Two common forms of certificate chains formed out of issued certificates are the hierarchical model, and the flat model. The hierarchical model models trust as a tree, with parent nodes being the issuers of each child node, and with users as the leaves of the tree. Users trust (or with the help of CAs, can establish trust in) every parent CA between themselves and the root node.

Trust in a hierarchical model can be established based on the trust of the root node. From there, trust in any node can be established. For example, in figure 1, if the holder of Cert2 wishes to communicate with the holder of Cert3, the Cert2 holder will establish trust in CA1's certificate, use this to verify CA4's certificate, then use this to verify Cert3.

The hierarchy of trust from the root node his is both a benefit and also the greatest weakness of the hierarchical model. First, if the tree gains any significant depth, verifying most certificates will take many public key operations, which will be computationally intensive. Furthermore, as the tree gets deeper, one cannot have as much faith in the trust of each leaf. A mistake by any one node would mean any node below it may not be trustworthy, and the deeper the tree, the more likely someone in the chain could make a mistake or be corrupted.

Flat models are the exact opposite of a hierarchical model, where there is no (enforced) hierarchy of trust. Each CA only signs keys directly under them, signing keys within their own domain. For users under different CAs, the CA itself is responsible for establishing trust with the other domains, which is where flat models tend to get more difficult to work with. Often this can be done using 'cross-certified' certificate authorities, where each CA can issue a certificate for other CAs for use by its own users. For example, in figure 2, if the Cert2 holder wishes to communicate with the Cert3 holder, the Cert2 holder would first obtain a certificate for CA2 from CA3, and then obtain a certificate for CA4 from CA2, and finally use CA4's certificate to verify Cert3.

These schemes can often be combined to form hybrid models, and several of these have been proposed, with varying degrees of success.

Another popular model, pioneered by PGP, is a web of trust. In the web of trust, there is no absolute view of trust, rather trust is determined relative to each user. A user determines through some alternate means some certificates that they trust, and sign them locally on their own devices to record this. They can also decide that they trust some of these entities to sign the certificates of others. If they have strong confidence in the user, they can even

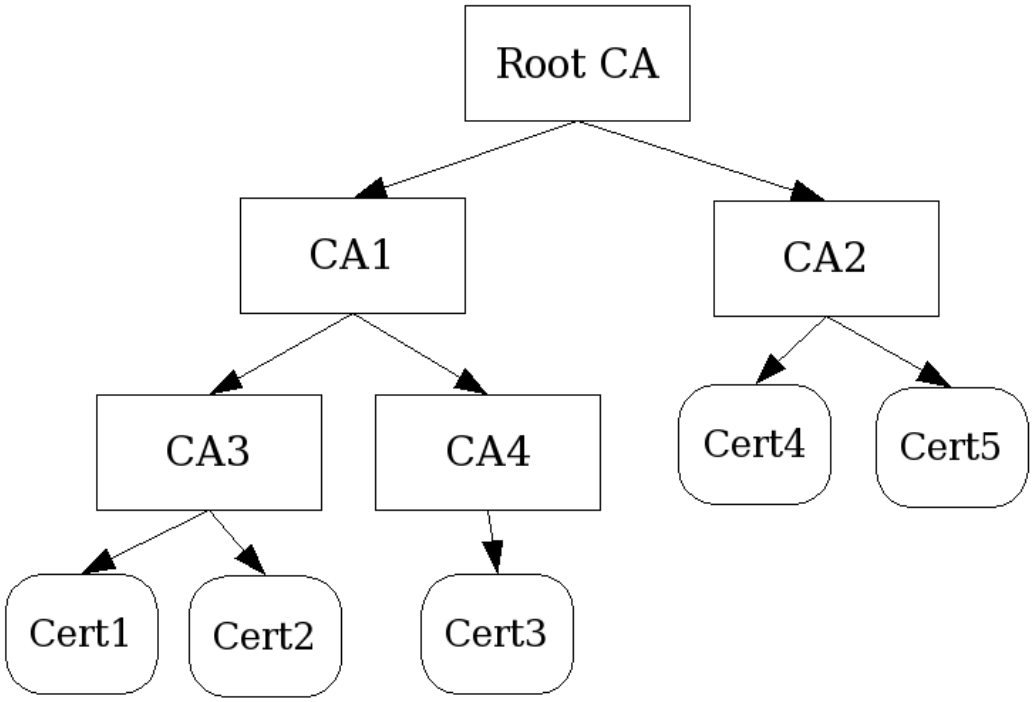

Figure 1: Example of a Hierarchical Trust Model

grant this transitively, so anyone that they trust to sign certificates may also grant others the trust to sign certificates.

The basic theory of the web of trust is that everybody knows everybody else through a small chain of people[2], and if everyone signs their friend's certificates, that most certificates encountered can be verified.

Its (rather vocal) proponents insist the web of trust is in every way superior [5] than other trust models, and insist that it is in fact a superset of hierarchical (and most other) trust models. However, its flexibility also hinders it, making it difficult to scale up, and difficult to manage, especially for newer users. Furthermore, transitively trusting over a long chain of end users is risky from a security standpoint.

[2]This is often referred to as "six degrees of separation", a theory that every Internet-connected person in the world is connected by no more than six people.

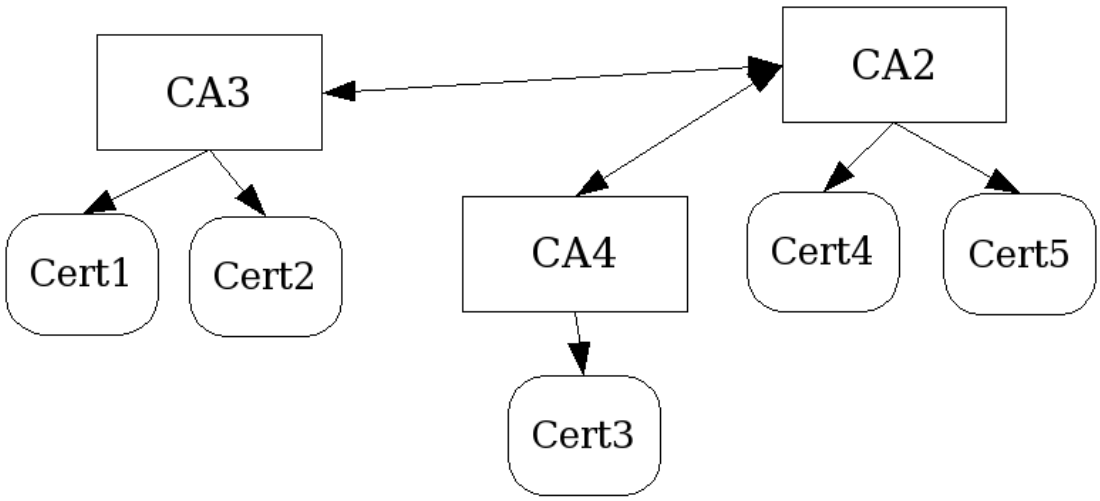

Figure 2: Example of a Flat Trust Model

2.4 Certificate Revocation

Certificate revocation is largely regarded as the most difficult problem to solve in PKI, but is an important requirement to make any PKI secure. Simply put, there must be a way to invalidate an otherwise valid public key. Every key pair has a security lifetime, after which the private key may have been compromised computationally. The private key may have been stolen, so trust of the matching public key must be immediately ended. In many ways worse, the private key may simply be lost; while not a security issue as is, one must notify others to change to a new public key - how does one trust such a request unless it's signed?

There are several difficult issues in revocation:

- Communicating the revocation of a certificate in a timely manner to all nodes.
- Storing, and distributing revocations in a scalable manner,
- Retroactive revocation: if you know a key was stolen a certain amount of time in the past, one must make it known that signatures made while that key was compromised may not be legitimate.
- Revoking a public key after the private key is lost.

The simplest scheme, and one used almost universally in PKI schemes, is a validity period for the certificate. This can trivially be stored in the certificate, and part of the information

signed by the CA. This means that the user must replace the certificate periodically, but also ensures that any security compromise is limited to the lifetime of the certificate.

The difficulty with certificate revocation is that users don't like having to get new certificates. So often, users will request certificates with infinite expiry dates, or will just use expired certificates. Invariably, the same sorts of users will be on the other side of the communication as well, ignoring the warnings about expired certificates.

A common approach to certificate revocation is that of a certificate revocation list (CRL). A list of invalid certificates is periodically published and circulated. A user verifying a certificate's validity is expected to check a recent CRL to ensure the certificate is not on that list. An compromised certificate can only be used within the period between the compromise and the next issue of a CRL; however this period must be carefully balanced. Too short, and the distribution medium is overworked, while too long a period will allow compromised keys to be used for too long a period of time.

Extensions to X.509[6] allow for the creation of CRLs, either stored in a directory system referenced by their distinguished name, or at a known URL. A CRL is created in a known format, containing a list of certificate serial numbers, the date the CRL was issued, and the expected date of the next CRL issue. The list is signed with the private key of the "CRL Issuer", which may be a different party from both the CRL distributor (who actually hosts the CRL) or the CA (who issued the certificates). In practice, the CRL issuer is often the CA. X.509 specifies an optional extension for a certificate to specify one or more distribution points for CRLs.

X.509's approach to revocation has a large advantage in that it can be implemented easily, especially where a LDAP directory exists. The downside though is the overhead: a large CA updating a CRL frequently would result in immense bandwidth requirements. Furthermore, the cost for a client to check a CRL on every operation, even if it was cached locally between CRL updates, could be prohibitive for many systems. Another problem with X.509 is that it relies on the issuer of the certificate to facilitate revocation - a certificate cannot be revoked directly by the end user.

Often a technique used from the user end for revocation is that of revocation certificates. These special certificates are generated either on-demand with the private key, or at the same time as the key-pair is generated. This certificate, signed with the private key corresponding to the user's certificate, is a message saying "my corresponding key is no longer valid!" The user keeps this revocation certificate stored securely until it is needed. Should the key be compromised, the user sends the revocation certificate to any source of the public key, where

the recipients record this action and mark the key as invalid. In cases where an issuer or signer needs to revoke a key, they can generate a similar revocation certificate specific to that signature.

The advantage of revocation certificates is that anyone in the system can revoke their part of the certificate. However, the difficulty in circulating this information can make it impractical in many systems. Furthermore, it requires the safe archival of the revocation certificate, which is something end users of certificates may not fully understand the implications of.

Implicit revocation refers to systems where a certificate is retrieved from a central authority whenever it is used. In these systems, revocation is implied by the successful retrieval of the certificate. To revoke a certificate, the central authority merely removes it from its storage. This has the disadvantage of not explicitly communicating the reason for revocation or date of compromise to parties who have recently used the certificate.

Windowed revocation[7] is a technique that combines CRLs and implicit revocation, by giving each certificate a 'revocation window'. This revocation window is normally longer than the release frequency of the CRLs, and shorter than the lifetime of the certificate. A certificate is acquired from a central authority, and can be used during its revocation window as long as the certificate is not on the most recent CRL. After the revocation window expires, the certificate must be refreshed from the central authority. This is similar to a CRL approach, except by using implicit revocation, CRLs are kept very small.

Instead of revoking a certificate, one could use a threshold scheme such that only with the approval of an authority can one use a private key at all. One scheme suggests the use of a semi-trusted mediator[8], who holds part of an RSA key. The user associated with the public key holds the other half. In this manner, both the mediator and the private key holder are required to contribute to sign a message or decrypt one. The advantage is instant revocation of a key is possible by removing the mediator's portion of it, however the disadvantage is that the mediator must participate in every private key operation. To users of the public certificate, the system is indistinguishable from regular RSA.

2.5 Key Escrow

While public keys can be distributed by any available medium, a private key must be closely guarded to ensure the security of a system. However, there are many reasons that one would want to make copies of a private key as well. Key escrow describes structures in which keys are shared with other parties.

There are several reasons people may choose to escrow keys. The most common asso-

ciation with the term key escrow itself is that of law enforcement. Many governments do not allow encryption to be used such that communications monitoring, as authorized by that government, cannot be performed. To protect the communication from parties other than the government, encryption is still used, but using a key available to the government. Normally, in a Government-run PKI, the government may generate the private key itself, in a service provided along with the signing of the public certificate.

Another reason to provide escrow is in a corporate setting, to allow superiors to recover encrypted files after an employee has left the company. This prevents the company from losing work performed by the employee when the employee is no longer with the company.

Finally, escrow can be used for archival purposes. Since another entity holds the private key, if the user should lose that key, they can recover the key from the entity that holds it. This can prevent loss of data.

Escrow must only be performed on private keys used for encryption. A digital signature key must never be escrowed, since the fact that multiple parties hold the private key can be used to break non-repudiation. Also, unlike an encryption key, past signatures can be verified with the public key and so do not require the private key to be archived.

One technique used in systems where the public key encrypts a session key is not to escrow the private key at all, but to escrow the session key. This is done by openly transmitting the session key, encrypted with a special public key. The government or other monitoring agency holds the matching private key, and can use it to recover the session key. Unfortunately, this relies on users to correctly use the system and not interfere with the transmission of the session key. While many governments have tried to legislate such a system into place, even offering tamper-resistant hardware to do so, it is difficult to enforce on a large scale.

One common place where such a technique is used is Windows' Encrypting File System (EFS). This system allows a user to generate a key-pair to be used to encrypt files and directories. The files are encrypted using DESX, using a random file encryption key, encrypted with the user's RSA public key. However, for each user with the 'recovery agent' role defined in the domain, the key is also encrypted with their public key. Any of the recovery agents can recover the files without the user's private key. Furthermore, the private keys are stored protected by the Windows operating system, but are accessible to the local system administrator, who may recover files, or even the keys themselves, at will.

2.6 Legal Infrastructure

One of the many goals that users of public key infrastructures hold is that 'digital signatures' will be able to hold the same legal weight as their paper equivalents. The ability to sign a document implies liability, both to the one who signs it, but also to the signer of the certificate, should the signature not be made by the person associated with the public key certificate. [9]

Several issues must be resolved in a digital signature law.

- The enforceability of a digital signature, as compared to a traditional signature, where signatures are mentioned in law.
- What level of cryptographic security is required to enforce a digital signature, and what technologies can be be used?
- Who is liable for a transaction: the signer, the CA who vouched for the signer, or the CA that certified the CA for the one verifying the signature? Is it a crime to create a fake certificate?
- How does one verify a signature in the future, after a certificate has expired, and any revocations have long since disappeared from CRLs?
- What is the legal status of signatures made recently before or after a certificate revocation?

Often, when law in a country is insufficient, contract law can be used to add some measure of enforceability to digital signatures. The Credentials[10] model is one such example of this approach. Designed for business-to-business interactions, the model uses the concept of credential holders, which are remarkably similar to certificate authorities. These credential holders sign, as a conventional contract, a memorandum of understanding with the others interacting in the system. This memorandum of understanding is a legally-binding document guaranteeing that each corporation be bound by their respective digital signatures. Each credential holder will sign the credentials (a certificate with a fairly large number of attributes) of others who have memorandums of understanding with them. This forms a flat trust model between the corporations, with each user within a corporation trusting only their own credential holder.

2.6.1 United States of America

Digital signature laws have been proposed and passed in various forms for nearly a decade. One of the first, in the state of Utah, allowed for digital signatures under a PKI with third-party verification to be binding on a person who made a signature. In 1999, the Uniform Electronic Transactions Act (UETA) attempted to make a model law for states to implement, not mandating a specific PKI, but merely requiring that a signature cannot be deemed unenforceable simply due to the fact that it is made electronically.

However, the biggest progress made was with the passing of the Electronic Signatures in Global and National Commerce Act of 2000, or the E-Sign Act for short. This was similar to the UETA except on a federal level. It states that digital signatures are enforceable, but also mandates that no consumer is required to accept such a signature. It also excludes most family law (such as wills and adoptions) and the cancellation or recalls or products or services.

2.6.2 Canada

In 1999, Canada adopted the Uniform Electronic Commerce Act (UECA). This law, remarkably similar to the UETA-based E-Sign law in the States, merely clarifies that digital signatures can be used in place of a regular signature. Here, more the intent to sign, rather than the details of the signature itself, becomes the key factor in the courts. A Quebec law goes into a bit more detail on establishing what is required to make a valid signature, but is still quite technology-neutral.

The Personal Information Protection and Electronic Documents Act (PIPEDA) has a section explicitly addressing digital signatures. It discusses a 'secure electronic signature', which would consist of signatures that follow certain regulations. These regulations have yet to be defined; in the short term, only Government of Canada certified signatures can be used, or those from systems certified by the Government of Canada. Secure signatures can be used wherever a signature is required by law, and can be entered into evidence in court.[11].

Many departments of the Canadian government are already using PKI, and there are plans for a Canadian Government central PKI for use by government and citizens. This technology will be based on technology provided by Entrust, a popular Canadian security firm. However, this has yet to be fully deployed.

2.6.3 European Union

The European Union passed the Electronic Signatures Directive in 1999. As a directive, each member country is required to draft laws within their own legislatures to attempt to implement the law. The directive describes the requirements for various level of digital signature, and describes what legal protection such digital signature provides in a court of law. It also describes the requirements and responsibilities of a certificate authority and a key holder for the law to apply.

Many EU countries have started to deploy PKIs. In Italy, government departments are already being connected with digital signatures. Spain has gone as far as to provide a certificate authority such that citizens and the government can communicate securely. Austria has actually started providing smart-cards for social security, which can also be used for private businesses. Others have been much slower to implement the law, or do not yet have digital signatures widely used.

For more information, see [12].

3 Examples of Public Key Infrastructures

This section will discuss several protocols and systems where PKI is actually being used. A PKI combines a variety of the attributes discussed above to make a working system, often combining it with details of the system that one is attempting to secure.

By analyzing the reality of how PKI is used, one can see the connection between theory and implementation reality. Unfortunately, there are very few truly widespread uses of PKI that are sufficiently well-documented to evaluate.

There is a huge number of systems using minor variants on X.509-based PKI structures. These function similarly to the way that https uses PKI, with the certificates in X.509 format, leading to the use of distinguished names, a single issuer, and X.509 certificate revocation lists. One key application is that of binary code signing, made popular by Microsoft-Windows based controls, as well as the permission-based security model on Java archives. A large variety of custom software used inside corporations and governments also use this model.

3.1 Secure Web Browsing

Perhaps the most pervasive use of PKI today, secure HTTP (https) is the protocol that protects most of today's web-based e-commerce from eavesdropping, tampering, and impersonation. It is simply HTTP, the protocol used for most web traffic, encapsulated inside a secure socket layer tunnel (SSL).

Many attributes of the system are determined by the system's use of X.509 certificates. The trust model is hierarchical, with each application containing a 'root set' of trust roots. In browsers, this means over one hundred certificates, and it is so far rare for the certificate resources to be shared across applications. While possible, in many applications it is difficult to actively manage which of these certificates are trusted.

Since root certificates are stored in each application, certificates can be sent as part of the connection negotiation. There is no central certificate store, making the system impossible to use in disconnected operation if one wishes privacy or to authenticate the receiver.

Revocation is handled optionally with X.509 validity periods, and with certificate revocation lists. In practice, certificate validity has many issues in implementation. Many of the popular certificate authorities will not hesitate to issue a certificate without an expiry date, or without a CRL. Even when they are provided, very few browsers were found to correctly query the lists when such interactions were required[13]. However, it is quite uncommon for people to even question a revoked certificate, an expired certificate, or even a certificate

which is not issued by a known CA.

The system provides privacy and authenticates at least one of the two parties. While the protocols (and most implementations) support client authentication, it is very rare that such features are actually used in practice. For this reason, client authorization is normally done by other more traditional means like username/password pairs.

3.2 Pretty Good Privacy

Pretty Good Privacy (PGP) has a strongly political history, originally released in 1991 in response to a government bill that would mandate placing back-doors into encryption schemes for government monitoring. The author, Phil Zimmermann, was eventually arrested for exporting munitions, but the charges were later dropped.[14]

PGP, while able to generically sign and/or encrypt any message, is designed for the support of end-user email. The system avoids the need for any central authority by allowing every user to generate their own certificate. Then, instead of a certificate being signed by an issuer, it is signed by zero or more 'signers', who have generated their keys the same way.

This forms a web of trust model; each user maintains a local list of public certificates in what is called a keyring. Certain certificates are chosen by the user to be trusted to sign other certificates, allowing a level of trust indirection. Furthermore, any certificate the user personally verifies with the owner may be signed with their own private key, giving them complete confidence in the key, and making the certificate valid for any user that trusts this signer to sign keys.

Since the trust model does not rely on any central authority, key-servers need no authentication, allowing anyone to upload any key. If a duplicate key is uploaded, any certificate changes are merged; for example, this is used by signers of a key to merge their signature with the public version. There is no way to remove information from a key on a key-server. One problem with the keyservers is that there is little commercial motivation to run such a server. While several are being run by volunteers at the moment, there are limits on what will be provided by administrators for free.

Revocation is largely ad-hoc, relying on revocation certificates. These act similarly to signatures, and can be uploaded to a key-server or emailed out to contacts. Signatures can also be revoked in the same way.

While a powerful tool, and used widely in technical and open-source communities, PGP has not had strong mainstream adoption. The average email user has little understanding of the need for encryption at all, and certainly does not understand the issues required for

secure management of their keyring. PGP requires users to be knowledgeable about the security implications of their actions, something that most people are not yet ready for.

While primarily used for email, one common use of PGP in recent years has been detached signatures on web pages and source code archives. Especially in the open-source world, there is a need to mirror source code to untrusted servers while at the same time allowing the files to be authenticated. The decentralized trust model of PGP avoids the cost of a central certificate authority, and there is often enough interaction between developer communities that a web of trust is effective in creating certificate chains.

3.3 Secure Shell

An upgrade to the classic UNIX 'r-tools' (rsh, rlogin, rcp) for remotely accessing servers, SSH provides features for remote access of machines coupled with strong cryptography. While an application-level protocol, SSH can act as a pipe between any two ports, allowing it to be used in much the same way as SSL. In 1997, IETF standardized SSH version 2, and it has since seen strong adoption among UNIX and Linux users.

A little-known feature of SSH is the ability to use public key cryptography for authentication. Many UNIX users were unhappy with the requirement (unlike the previous rsh tools) of having to enter a password to log in. By using PKC, passwords can be avoided, again allowing passwordless use of the tools. The PKI used is truly minimal, showing how system properties can be used to avoid many of the larger systems, as is common in X.509 systems.

Certificates are very simple in SSH, consisting of only a key type identifier (key-size, PKC system), the key itself in ASCII-encoded form, and an unconstrained string identifier. These keys are stored in a special directory within user accounts on the host. Trust in the key is implicit in its association with a user account; to associate the key with the user account, one must have access to the account. This means that fundamentally there must always be at least one alternate method of access to the account, normally through intervention of a system administrator or through traditional account passwords (which ssh also supports).

Revocation is implicit: a key can be instantly revoked by removing it from each server on which it resides. This can be a time-consuming process if many servers are using the key, but since keys are only propagated by the holder of the private key, the number of hosts that need to be contacted is limited.

4 Observations and Conclusions

The world seems to be moving quickly towards centralized and hierarchical PKI systems. X.509 in particular has entrenched itself deeply, and will likely be the certificate format that the world commits to, for better or for worse.

While the last several years have been dominated by the secure-HTTP experience, the idea that everyone will hold hundreds of trusted root certificates is slowly dying. Recent laws seem to show an increasing push for regulation of, or even administration, of CAs from a government level, with a hierarchical trust model. More and more, the governments of the world are likely going to become the roots of hierarchical systems, with many countries linked by cross-certification.

As corporations rely more and more on advanced directory services, PKI will become more prominent within organizations. Cross-organization communication, while coming much later, will eventually come as new laws give these companies the confidence and incentives to use such systems.

One thing is certain: despite a slow start, PKI is definitely realizable with today's technologies, and will see much larger deployment in the near future.

References

[1] *ITU-T Recommendation X.509, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*, International Telecommunication Union Telecommunication Standardization Sector, 1996.

[2] B. Kaliski Jr., "An Overview of the PKCS Standards," Technical Note, RSA Laboratories, 1993.

[3] "How PGP Works", *The International PGP Home Page*, http://www.pgpi.org/doc/pgpintro/ (current April 2003).

[4] "IBE Secure E-Mail,", *Applied Crypto Group - Stanford University,* http://crypto.stanford.edu/ibe/ (current April 2004), Apr. 8, 2002.

[5] P. Zimmermann, "Why OpenPGP's PKI is better than an X.509 PKI,", *OpenPGP Alliance*, http://www.openpgp.org/technical/whybetter.shtml, Feb. 27, 2001.

[6] R. Housley, W. Polk, W. Ford, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC3280, Network Working Group, The Internet Society, April 2002.

[7] "Windowed Revocation of Public-Key-Encryption Certificates," *NASA Tech briefs*, http://www.nasatech.com/Briefs/Aug02/KSC12149.html (current April 2004).

[8] D. Bonehm, X. Ding, G. Tsudik, C. M. Wong, "A Method for Fast Revocation of Public Key Certificates and Security Capabilities," *Proc. 10th USENIX Security Symposium*, pp. 297-308.

[9] J. Angel, "PKI and the Law," *Network Magazine,* http://www.networkmagazine.com/article/NMG20001004S0010 (current April 2004), Oct. 5, 2000.

[10] "Credentials - a new Trust Model for Business-to-Business Communications," http://www.dga.co.uk/customer/publicdo.nsf/0/AE18F74BE8C1BF43802568B3005922DB?OpenDocument (current April 2004).

[11] J. Gregory, "Canadian and American Legislation on Electronic Signatures with reflections on the European Union Directive,", http://droit-internet-2001.univ-paris1.fr/pdf/ve/Gregory_J.pdf (current April 2004), Nov. 13, 2001.

[12] M. Mazzeo, "Digital Signatures and European Laws," *SecurityFocus,* http://www.securityfocus.com/infocus/1756 (current April 2004), Jan. 26, 2004.

[13] P. Festa, "Experts say browsers fall short on certificate alerts," *CNET News.Com*, http://news.com.com/2100-1023-247851.html?legacy=cnet (current April 2004), Oct. 30, 2000.

[14] "Origins and Ideas of PGP," http://www.lugod.org/presentations/pgp/history.html (current April 2004).