PKI Seeks a Trusting Relationship [⋆]

Audun Jøsang[1], Ingar Glenn Pedersen[2], and Dean Povey[1]

[1] Distributed Systems Technology Centre
Brisbane, Qld 4001, Australia
[2] The Norwegian University of Science and Technology
N-7491 Trondheim, Norway

Abstract. All human co-operation is based on trust, meaning that we choose co-operation partners and make commitment decisions based on how much we trust the other party. Digital certificates and public-key infrastructures represent an attempt to mimic real-world human assessment of identity and trustworthiness in an automated and mechanical fashion, but present implementations are based on a very limited trust model making them inadequate as a general tool for trust assessment and decision making. This paper describes public-key infrastructures in general and discusses issues related to trust management of public-key infrastructures.

1 Introduction

Public-key cryptography solves security problems in open networks but creates key management complexity. Digital messages can for example be signed by a private key allowing anyone with access to the corresponding public key to verify that the message is authentic, but this principle depends on the authenticity of public keys and the problem boils down to finding a method for secure distribution of public keys.

Public-key infrastructures (PKI) simplify key management and distribution but create trust management problems. A PKI refers to an infrastructure for distributing public keys where the authenticity of public keys is certified by Certification Authorities (CA). A certificate basically consists of the CA's digital signature on the public key together with the owner identity, thereby linking the two together in an unambiguous way. In order to verify a certificate the CA's public key is needed, thereby creating an identical authentication problem. The CA's public key can be certified by another CA etc., but in the end you need to receive the public key of some CA out-of-band in a secure way, and various solutions can be imagined for that purpose.

However, there is a problem in this design. What happens if a CA issues a certificate but does not properly check the identity of the owner, or worse, what happens if a CA deliberately issues a certificate to someone with a false owner identity? Furthermore, what happens if a private key with a corresponding public-key certificate is leaked to the public domain by accident, or worse, by intent? Such events could lead to systems and users making totally wrong assumptions about identities in computer networks. Clearly CAs must be trusted to be honest and to do their job properly and users must be trusted

[⋆] Appears in the Proceedings of ACISP 2000, Brisbane, Australia, July 2000 to protect their private keys. Trust management includes methods for assessing policies regarding issuance and handling of public-key certificates and for determining whether these policies are adhered to by CAs and users, with the purpose of making decisions related to on-line activities.

2 Public Key Infrastructures

Internet users can not base their judgements about identity of remote parties on faces or familiar voices, meaning that electronically received information a priori can not be trusted. Instead public-key certificates combined with public-key cryptography can be used to authenticate identity and message origin.

2.1 Certification Chains

The first certification system in any standard was the X.509v1 Authentication Framework[1], designed by CCITT/ITU for the purpose of securing the X.500 Directory[2] and intended to work with a hierarchy of certification authorities.

X.509v1 has been further developed into X.509v2 and X.509v3 in order to overcome weaknesses in the earlier versions, and X.509v3 is the basis of the IETF PKIX working group which is aiming at developing a general purpose public key certification infrastructure for the Internet. The format of X.509 certificates is illustrates in Fig.1.

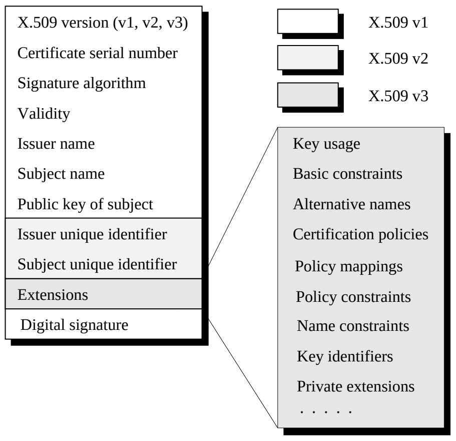

Fig. 1. X.509 certificate with extension part

The X.500 Directory was never adopted by the Internet community because it and the corresponding Directory Access Protocol (DAP) [3] were seen as too complex for simple Internet clients to use. Instead LDAP (Lightweight Directory Access Protocol)[4] which is a relatively simple protocol for updating and searching directories running over TCP/IP is used.

The X.509 standard is often vague and open-ended, which means that an additional X.509 certificate profile is required in order to implement certificate handling on a real system. Since the original X.509 certificate format was specified to fit with the original X.500 directory which is no longer used, the format contains a number of more or less redundant fields. In the main body only “Validity”, “Public key” and “Digital signature” are really needed. “Issuer name” and “Subject name” are still used, but they are interpreted as general names rather than entries in a directory. Apart from that everything else is found in the extension part.

In order for two users to verify the authenticity of each others public keys it is sufficient that there exists a certification path between them. A certification path is an ordered sequence of certificates which together with the public key of the initial certificate in the path can be processed to obtain the public key contained in the final certificate in the path. The rules for certification path validation are quite complex depending on what extensions are in the certificate. See e.g. [5] for examples.

2.2 Certification Hierarchies

Certification between nodes is directed from the certifier to the owner of the certified key. Certification is unidirectional when an agent X certifies the public key of another agent Y, and bidirectional when the agents X and Y certify each others public keys. In the PKI jargon *“certifying a user”* means *“certifying the user’s public key”*. Chained certification can form different topologies.

Strict Hierarchy. Most commercial PKIs are strict hierarchies, as illustrated in Fig. 2, and most only consist of one or two levels. The certification paths go strictly from the top root CA, eventually via intermediate CAs, and down to users, where the users are assumed to be certified by the leaf nodes.

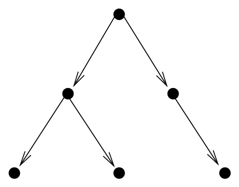

Fig. 2. A strict certification hierarchy

In a strict hierarchy all users can be easily identified and found because of the hierarchic structure. A user must know the public key of the top root in order to resolve certificate chains and establish a certification chain to any other user in the hierarchy.

General Hierarchy. A general hierarchy includes two-way certification between CAs, as illustrated in Fig. 3.

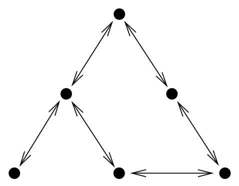

Fig. 3. A general certification hierarchy

When certification takes place in both upwards and downwards direction, each user will only need to obtain an authentic copy of the nearest CA's public key, while still being able to establish a certification path to every other user in the network. The X.509 standard [1] suggests a general hierarchy of this type, but no commercial PKI uses this topology.

Anarchic PKI. The opposite to a hierarchic structure is an anarchic structure where each CA (and user) is free to choose which other CAs (and users) it wants to certify, as illustrated in Fig. 4.

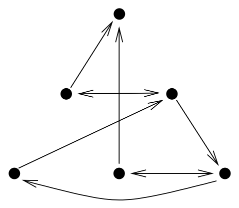

Fig. 4. Anarchic certification structure

The anarchic structure corresponds to the Web of trust on which PGP [6] is based. It consists of unidirectional and/or bidirectional certification between arbitrary agents. There is in principle no difference between users and CAs. The disadvantage of an anarchic certification network compared to a hierarchic structure is that there exists no simple algorithm for identifying certification paths between all users of an anarchic network, whereas such algorithms exist for hierarchic networks. A user must obtain as many public keys as possible in order to establish certification chains to other users.

Isolated Hierarchies. Many PKIs can exist in parallel without being linked to each other as illustrated in Fig.5.

A user must obtain the root public key of every PKI in order to verify certificates of users in all hierarchies. Users belonging to hierarchies with unknown root can not be identified.

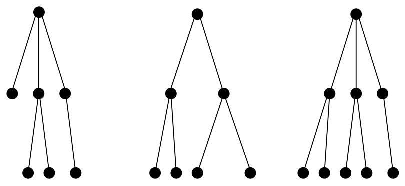

Fig. 5. Isolated certification hierarchies

PKIs used for the Internet Web consists of isolated strict hierarchies, and is in fact a topology of this kind. The root public keys are stored hard-coded in the most popular Web browsers. There are about 50 root keys delivered with Netscape Communicator release 4.6 or Microsoft Internet Explorer release 5.0.

Cross Certified Hierarchies. In order to avoid requiring that users acquire several public keys the hierarchies themselves can be cross certified as illustrated in Fig.6. In case the hierarchies are strict it is sufficient that a user obtains an authentic copy of the own root's public key while still being able to establish a certification chain to any user in any hierarchy.

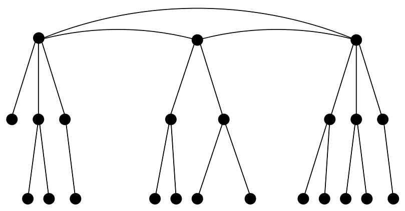

Fig. 6. Cross certified hierarchies

Cross certification between PKIs would simplify the distribution of root public keys, and would make PKIs truly open. The main problem opposing this development is that CAs in general have incompatible policies whereas cross certification requires some sort of policy alignment. A government can for example enforce a common policy and cross certification between all PKIs used by the public administration, but spontaneous cross certification between commercial PKIs has so far not been widespread.

3 PKI Management

The cryptographic aspects of PKIs are relatively well understood. The deployment of PKIs on the other hand requires management, and so far we have seen the emergence of two types of PKI management.

3.1 Web PKI

For Web PKIs the root public keys are hard-coded in Web browsers as *self signed* X.509v3 certificates, i.e. the public key has been certified by the corresponding private key. The only purpose of self certification is to simplify the certificate handling; the browser only needs to deal with certificates. Self certification provides no additional trust in the public key, and as such the term "self certification" can be misleading.

Since root certificates are hard-coded in the browsers they can not easily be upgraded. Root key management must in fact follow the pace of browser releases and distribution. Not only must changes be implemented in the next release of the most popular browsers, the users also have to upgrade the browser on their computers to the newest release. If for example public key revocation shall be useful it must be possible to enforce it relatively rapidly. Because this is not possible for root certificates it is in practice not possible to revoke them.

The most widely used application is presently to establish encrypted connections using the SSL protocol[7]. Another popular application is email encryption based on the S/MIME [8, 9] standard which consists of digitally encrypting the body (and not the head) of email messages. A third application is for digitally signing SW components. The security problem users are facing regarding active components such as Java applets and Microsoft's ActiveX components is whether such imported programs can safely be executed. One way this can be solved in Web browsers is to have the components digitally signed by the manufacturer's public key which previously has been certified by a CA. This only indicates the SW manufacturer's identity and does not say whether it is safe to let the SW component be executed.

3.2 Managed PKI

In contrast to Web PKIs, a managed PKI does not distribute root public keys piggy-backed with Web browsers, but is based on separate out-of-band procedures managed by the organisation that operates the PKI. This organisation usually operates CA servers from which user certificates can be down loaded. Managed PKIs are operated by an organisation to meet specific needs within the organisation or as a business activity. The organisation will have full control over the trust structure in the PKI hierarchy, but without being Web-born managed PKIs do not easily get global coverage. Managed PKIs can provide high trust and thus be suitable for high value transactions.

Organisations operating managed PKIs can decide to, or be enforced by law in a particular country, to establish cross certification to other managed PKIs and in that way create a PKI consisting of several interlinked certification hierarchies.

Secure distribution of the root public key is essential for managed PKIs, and a typical solution is to equip each user with a smart card containing the users private key in addition to the root public key.

4 Trust Management

It is assumed that trust is a belief based on knowledge, experience and perception. In the physical world trust in things and in other people is based on our experience with them, information we have received about them and how they appear to us. All this makes trust a very subjective phenomenon, meaning that I don't necessarily trust the same things or the same people as you and vice versa. The number of people we can potentially relate to within a physical world is also limited by distance and physical constraints. In the cyberworld on the other hand the number of people we can potentially relate to is only limited by the number of people that are on-line.

Cryptography can be interpreted as a mechanism for transferring trust from where it exists to where it is needed. For example if you initially trust the authenticity of a public key and you verify a messaged signed by the corresponding private key, then you will also trust the authenticity of the message. As such certificates and PKIs do not create trust, they merely propagate it, and users must initially trust something. Initial trust is traditionally established off-line, i.e. in the physical world, but it is perfectly possible to get experience and gain trust purely through on-line activities.

The challenge is to find a good method for making trust assessments about potential remote transaction partners in computer networks. A transaction partner can be someone you already know but it can also be someone who is totally unknown and with whom you have never interacted before and with whom you might never interact again after the transaction.

Trust assessment must be based on some initial trust combined with trust propagating mechanisms, and should provide a basis for decision making. Trust management is about all that, and can be defined as activity of making trust assessments by collecting, analysing and codifying relevant evidence with the purpose of making trust based decisions. The number of potential transaction partners on the Internet is presently around 100 million and the goal must be to make trust management schemes scalable to that size.

4.1 Certification Policies and Certification Practice Statements

The degree to which users can trust the binding between the public key and the owner identity stored in a certificate depends on several factors, including the practices followed by the CA in verifying the identity of the owner, the CA's operating policy, procedures and security controls, the owners obligations e.g. regarding secure storage of the private key, and legal obligations of the CA such as e.g. warranties and obligation limitations.

According to X.509 a certification policy is "*a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements*" [1]. To the degree that a certification policy exists or is applicable to a particular application it provides users with evidence for assessing the trustworthiness of certificate issued by CAs that adhere to the policy.

A more detailed description of the practices followed by a CA can be found in its Certification Practice Statement (CPS). According to the American Bar Association Digital Signature Guidelines "*a CPS is a statement of the practices which a certification authority employs in issuing certificates*" [10]. The CPS defines under what conditions certificates are issued and which liabilities the CA takes on itself and which are put on the user. Usually a CA offers different certification classes with varying degree of confidence in the key-to-owner binding with the purpose of being suitable for different applications. The general principle is: The higher the confidence, the more thorough the identity verification before issuing a certificate. For commercial CAs it can be added that the price you pay for a certificate increases with the confidence level.

The concepts of certificate policy and CPS were developed by different bodies for different reasons. A CPS is very specific and usually applies to a single CA whereas a certificate policy is more general and is intended to be applicable to larger domains.

Computers do not understand policies and CPSs although some attempts are being made for that purpose by including a certification policy extension in the X.509 certificate together with policy constraints and policy mappings. The certification policies extension in its minimal form provides a means of identifying the policy a certificate was issued under.

The idea behind the policy constraints and policy mappings is that the application can automatically check and enforce them at run time, for example by checking that the certificate policy corresponds to the application policy or that the certification policies of certificates in a chain are compatible. This of course requires standardised policies and constraints, and although seemingly difficult to achieve it would create several new possibilities.

4.2 Trust Management for the Web PKI

Unfortunately the notion of certificate policy does not exist for the Web. Primarily it would be rather difficult to define a policy that satisfies all user communities as well as all present and future applications, and secondly if such a policy could be defined it would probably be too general to be useful to anyone. Instead each CA that operates on the Web has its own CPS.

If it shall be meaningful for a user to trust a PKI based on the trust in the root CA, then the CPS strength of intermediate CAs must be equal or increasing downwards in the hierarchy. In the same way, if it shall be meaningful to trust other cross certified PKIs based on the own root CA, then the CPS strength in other PKIs must be equal or greater. However, CPSs are usually difficult to compare unless they are accredited against an industry standard policy and can not be translated into a one-dimensional measure. It is therefore difficult to establish cross certification between PKIs, and as a result the Web PKI consists of isolated hierarchies, but because the public keys of all root CAs are hard-coded and distributed with the Web browsers cross certification is not really needed.

When accepting a certificate the user should really check the corresponding CPS, and if applicable the certification policy, to see what the certificate is worth, but because this would require the user to read a document of at least 10 pages each time a secure Web site is visited it is hardly ever done. The trust model of the Web PKI is more the result of the need for a business model than the need for good trust management.

The advantage for the major browser manufacturers is that they can provide CAs with global coverage by distributing the CA root public keys hard-coded in the browsers, the advantage for the CAs is that they can provide Web server operators with global acceptance of their server certificates, and the apparent advantage for the users is that they get a Web interface that anyone is able to use and that hides the complexities of the underlying cryptographic mechanisms.

A problem with this model is that distribution of root keys is linked to the distribution of particular Web browsers. As already mentioned the present implementation of PKIs in Web browsers does not allow certificate revocation and thereby represents a time bomb. If a private key is leaked to the public domain the corresponding certificate can be misused by anyone in order to set up false secure Web sites, send false email messages or distribute malicious SW under false name, and tragically there exists no simple way of stopping such attacks even after they have been reported. Another problem is that the Web interface provides very poor trust management to the user. In order to trust a secure Web site, an encrypted message or a digitally signed SW component only the certificate owner identity and the CA identity are available to the user. The user can only make an informed decision if both are known, but is left totally in the dark in case they are not. The browser interface usually presents dialog boxes but experience shows that users find dialog boxes annoying and tend to blindly click "yes" or simply turn off the security settings in order not to be interrupted by dialog boxes anymore. The following section describes potential consequences of these problems.

4.3 The SSL Attack

In order for a user to trust that something is authentic he or she must be presented with some evidence that can be correctly interpreted. For the purpose of verifying the authenticity of Web servers SSL represents a strong authentication mechanism, but unless the evidence of this verification is intelligibly presented to the user the mechanism can be quite useless.

Assume a user A who wants to access Web services from secure server B which for example can be a bank providing financial transaction services for its clients. In the normal scenario, client A points his Web browser to bank B's Web site. The Web server returns B's certificate Cert_B to A's browser which verifies the certificate using the pre-stored public key of the root CA that generated Cert_B. After successful certificate validation A's browser continues the communication with B in secure SSL mode.

Fig.7 below shows user A's client machine on the left and bank B's server on the right side. We will show that the intruder T in the middle is able to make both A and B think they are communicating with each other although they in fact communicate with T.

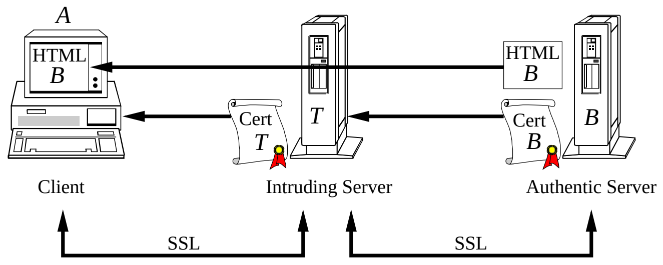

Fig. 7. Attack on SSL authentication

In the attack, the intruder Web server T acts as a relay between A and B passing the HTML pages from B to A and the requests from A to B. For the attack to work user A must be fooled into pointing her browser to T instead of to B. This can for example be done by placing a false URL on a portal until somebody accesses T from it in the belief that he or she accesses B. After a successful attack, the false URL can be removed in order not to leave any evidence of where the attack came from.

It is assumed that the intruder T has obtained a valid certificate Cert_T, either by buying it or because a private key with a corresponding certificate has been leaked to the public domain. When the client A has established a SSL connection to the intruding server T using T's certificate, the intruding server establishes a SSL connection to the bank B using the bank's certificate Cert_B and simply relays the data sent by A and B to the opposite sides via two different SSL connections, including possible user passwords, so that A and B think that they are communicating with each other. When A sends a request to transfer money from her own account for paying a bill, T is able to modify the destination account number and the amount.

When a secure SSL connection is established the server is supposed to be authenticated by the client, as indicated by the key or the padlock icon on the browser window. However, this only indicates that something is authenticated and not what in particular, which for all practical purposes means that nothing at all has been authenticated. The blame for this vulnerability can of course be put on poor interface design, but the problem is also related to user awareness, and technology can only help making awareness easier to practice.

The browser does allow viewing a certificate by clicking on the padlock icon, but users hardly ever do this, and even security aware users who view the certificate when accessing a secure Web site can have difficulty in judging whether the information on the HTML page has been sent by the owner of the certificate. The system interface should make it easy to view, understand and believe in certificates. Requiring the user to explicitly make several extra mouse clicks in order to view a certificate that is hardly intelligible is simply not good enough.

A better user interface can be based on including the server company logo in the certificate and let the interface always display it when a secure connection is established. This will allow a quick visual check anytime without extra mouse clicks in order to verify the server identity. Alternatively an audible message can be included in the certificate and played to the user. However, users might not pay any attention to these features after a while, so that the problem of user awareness will not go away. All we can do is to create a system interface that helps users to be more aware.

5 A Better Trust Management

Some of the problems persisting in present PKI implementations include:

- **Computers don't understand the semantics of a policy.**
  A certificate policy or a CPS is a piece of rather lengthy and complex prose text that can only be read by humans. It would be desirable to be able to specify certificate policies in such a form that they can be interpreted by computers in an automatic and mechanical fashion.

- **Cross certification requires equal Policies.**
  If it shall be meaningful to trust the root CA in a PKI as a representative of cross certified PKIs then the policy strength must equal in both PKIs. However, "policy strength" is a multi-dimensional measure that is difficult to match. Cross certification is therefore difficult to achieve unless the policies themselves are equal.
- **PKIs do not handle trust dilution.**
  Real-world trust is intuitively diluted in a chain of recommendations, but present PKI implementations only provides a binary trust model.
- **PKIs do not take into account parallel certification paths.**
  Real world trust is often based on multiple recommendations. It would be useful to have something similar in PKIs but present implementations only handle single certification chains.
- **PKIs give little support for decision making.**
  PKIs give little support for answering questions such as e.g.: "*For a given certification path and a given user what is the upper transaction value you are willing to risk?* " and "*For a given transaction how can you select the transaction partner that will minimise your risk?* ".

5.1 Including Subjective Trust Measures in Certificates

One possible solution is to include subjective measures of trust within a certificate and combine these parameters in order to deduce trust in remote users and systems. Such a trust measure can represent the policy within the certificate in addition to trust in the reliability of CAs and users themselves, can be understood by humans and can be automatically handled by computers. In addition, trust dilution can be handled by combining trust measures in series and trust from multiple certification paths can be combined in parallel. It would allow cross certification to be established between arbitrary pairs of CAs by simply specifying within the certificate how much one CA trusts the other. Finally trust measures can be combined with transaction utility functions in order to support decision making.

This type of trust model is for example described in [11]. Unresolved problems related to this model is how users can consistently determine subjective trust measures. Another problem is that the trust measures are communicated to other users and thereby can not be kept confidential whereas a CA might not want to disclose that it distrusts users or other CAs. Finally CAs might not want to make any explicit statement about trust in other CAs as it might imply liabilities.

5.2 Policy Alignment

A common misnomer with X.509 based PKIs is that CAs are making statements about trust when issuing a certificate. In most cases, a CA is merely following a defined procedure and evaluating objective evidence to determine a binding between a public key pair and the issue of whether to trust this user is one that the relying party must resolve, based on their subjective judgements about the statements and policy made by the CA. This is particularly important in the case of certification chains, which are usually not "chains of trust" as some PKI literature suggests, but are in fact simply a series of signed statements by CAs that attest to a binding between keys and some objective evidence as determined by following the process outlined in a particular policy and CPS. It is believed that the commercial nature of most CAs precludes them from making subjective statements, as this exposes them to too much risk. Contrast with the PGP system which is based on meeting the needs of small communities of users, and where users are much more likely to make statements about trust.

One of the biggest problems for users when evaluating policy chains is the problem of ensuring equivalent policies across all links of the chain. The approach taken by X.509 is to specify a path validation algorithm that requires that all certificates in a chain contain the same policy identifier, or a policy that has been "mapped" to an equivalent policy using the policyMappings extension. This approach reflects the general philosophy of X.509, which tends towards a centrally managed design. This is also consistent with related work such as the NIST PKI architecture[12] and work being done in the Australian Standards working group IT/12/4/1 to develop an architecture for PKI systems [13]. Both these approaches have an architectural model which consists of a hierarchy of components consisting of the following:

– **A Policy Approval Authority (PAA).** This is a root authority whose job is to approve the policies of other entities in the hierarchy. Note that the PAA itself does not assert any policy, it simply issues a self-signed certificate that must be trusted by some out of band means. The PAA will most likely be a statutory body created by a government and will operate according to a particular set of rules.
– **A Policy Creation Authority (PCA).** The role of the PCA is to act as a policy creation body of a community of users. An example would be a group of banks who get together to decide on a common policy for retail/wholesale banking. These banks would form a single entity – a banking PCA, that was responsible for setting policy in the banking sector. The PAA would then approve the policy(s) set by this PCA by issuing it with one or more certificates asserting the policy identifier.
– **Certification Authorities.** The CAs will issue certificates under one or more policies. These policies may have been set by a PCA, or they may be specific to the CA (either with an approved policy – i.e. the CA is acting as a PCA, or an unapproved policy – i.e. the CA is operating outside the architectural model).

This paradigm encourages users to trust the PCA to set an appropriate policy for the domain of application. The user simply selects the policy identifier defined by the PCA for the given application and trusts that the infrastructure and requirements set up by the PCA and PAA are such that the policy will be appropriate, and that CAs operating under that policy will behave appropriately. In a sense the user is deferring some of their trust to the infrastructure. This is definitely appropriate in many cases, we tend to rely on our legal, and socio-political systems to a large extent when conducting commerce in the real world, so it only seems natural to translate this into the electronic world.

However, there are also a number of problems with this model. Many governments (including Australia's) have punted on the idea of setting up a PAA. This seems largely due to the perception of this body as being a huge liability risk due to the large amount of damage possible from compromising the PAA's private keys. The absence of a PAA favours a disjoint model, with a number of PCAs acting as root authorities for their community of interest, and requiring the user to obtain the public keys/self-signed certificates of each of these CAs by out-of-band means.

In addition, it makes the issue of cross-certification quite difficult. In order to cross-certify, two CAs must agree to mark each other's policies as equivalent. However, it is more than likely that this will not be the case. X.509 contains a number of mechanisms for parent CAs to prevent sub-ordinate CAs from doing this. They can inhibit the ability for child CAs to do policy mapping, and they can restrict the certification path such that a subordinate CA can only issue certificates to end-entities and not to other CAs. It is more than likely that cross-certification would have to occur at the level of PCAs to enforce the policy. This can also be used to enforce commercial limitations on CAs. For example, a commercial CA with it's root keys embedded in the browser sells a cross-certification service so that another CA can issue certificates that do not require browser bootstrapping. However, this CA is unlikely to want to allow the subordinate CA to then resell this service to other CAs and so will seek to limit the subordinate CA to only issuing end-entity certificates.

Another criticism is that while a single policy identifier is enforced across multiple CAs, there are clearly different roles being played by each certificate in the chain. For example:

- The PAA will issue a self-signed certificate containing its public key. As noted this is simply a convenient mechanism for "bootstrapping" the certification path, and therefore asserting a policy in this certificate is basically meaningless.
- The PAA will then issue a certificate to the PCA asserting the policy identifier which the PCA has created. This certificate is a statement in effect that the PAA "approves" this policy for use within its framework.
- The PCA will then issue a certificate to one or more CAs asserting the policy identifier it has created. This certificate is a statement that the CA identified is authorised to operate within the community of interest that the policy defines.
- The CA issues a certificate to an end-entity. This certificate is a statement that the CA has followed the rules of the policy as they relate to registration of users, and has verified the binding between the user and their public key.

While these roles are very different, they are essentially implicit semantics, and the only thing that the user knows is that the certificates have all been issued under the same/equivalent policy. The user must simply trust that the framework that exists has ensured that the ordering of CAs in the chain is appropriate to their function.

While this may be appropriate in some instances, it makes it difficult to build ad hoc relationships, and it means that information is lost in the process. This is particularly problematic in the case of cross-certificates.

What is needed is a way for the information about policies to be expressed in a standard interpretable way such that a relying party can make all the decisions about whether policies are equivalent, and whether they are appropriate for a given application. It is after all, the relying party who has to bear any risk about who/what they trust.

6 Conclusion

A public-key infrastructure is technically seen as a structure of cryptographically linked data objects that can be transmitted through a data network and from which public keys can be extracted. However, extracting public keys and using them in security applications is only meaningful if the public-key infrastructure is linked to a corresponding trust structure, and for this purpose trust management is needed. Trust management can be defined as the activity of making trust assessments by collecting, analysing and codifying relevant evidence, with the purpose of making trust based decisions. Trust models in present PKI implementations are too limited for providing users with trust management facilities, and possible solutions include more rich trust models and standardised certificate policies.

References

1. ITU. *Recommendation X.509, The Directory: Authentication Framework* (also ISO/IEC 9594-8, 1995). International Telecommunications Union, Telecommunication Standardization Sector(ITU-T), 1993.
2. ITU. *Recommendation X.500, Data Communication Network Directory*. International Telecommunications Union, Telecommunication Standardization Sector(ITU-T), 1993.
3. ITU. *Recommendation X.509, The Directory: Abstract Service Definition* (also ISO/IEC 9594-3, 1995). International Telecommunications Union, Telecommunication Standardization Sector(ITU-T), 1993.
4. M. Wahl, T. Howes, and S. Kille. *RFC 2251 - Lightweight Directory Access Protocol Version 3 (LDAPv3).* IETF, December 1997.
5. R. Housley, W. Ford, W. Polk, and D. Solo. *RFC 2459 - Internet X.509 Public Key Infrastructure, Certificate and CRL Profile,.* IETF, January 1999.
6. P.R. Zimmermann. *The Official PGP User's Guide.* MIT Press, 1995.
7. Netscape. *The SSL 3.0 Protocol.* Netscape Communications Corp, 1995.
8. S. Dusse. *RFC 2311 - S/MIME Version 2 Message Specification.* IETF, March 1998.
9. S. Dusse. *RFC 2312 - S/MIME Version 2 Certificate Handling.* IETF, March 1998.
10. ABA. *Digital Signature Guidelines: Legal Infrastructure for Certification Authorities*. American Bar Association, 1995.
11. A. Jøsang. Trust-based decision making for electronic transactions. In L. Yngström and T. Svensson, editors, *Proceedings of the Fourth Nordic Workshop on Secure Computer Systems (NORDSEC'99)*. Stockholm University, Sweden, 1999.
12. W. Burr, D. Dodson, N. Nazario, and W.T. Polk. Minimum Interoperability Specification for PKI Components, Version 1. Technical report, NIST, September 1997.
13. Standards Australia. *PKAF Report: Strategies for the implementation of a public key authentication framework (PKAF).* WG IT/12/4/1, 1996.