

Walking the Web of Trust

Germano Caronni

Sun Microsystems Laboratories, Palo Alto
Email: <gec@acm.org>

Abstract

Most currently deployed Public Key Infrastructures (PKIs) are hierarchically oriented and rely on a centralized design. Hierarchical PKIs may be appropriate solutions for many usage-scenarios, but there exists the viable alternative of the 'Web of Trust'. In a web of trust, each user of the system can choose for himself whom he elects to trust, and whom not.

After contrasting the properties of web-of-trust based PKIs to those of hierarchical PKIs, an introduction to webs of trust and to quantitative trust calculations is given. The paper concludes with the presentation of an efficient, sub-exponential algorithm that allows heuristic computations of trust paths in a web of trust.

Keywords: Web of Trust, PKI, Heuristic Trust Calculation.

1 Introduction

The current boom in e-commerce relies to a certain extent on our ability to communicate securely. The slow but steady progress of IPSEC, the ongoing deployment of VPN firewall technologies and the ever-widening use of SSL for secure transactions over the world-wide web are just a few examples for this. But as the communication infrastructure offers more and more security-related functions, the limitations of the overlaying key management frameworks become more and more visible.

The ultimate purpose of key management is to assure its users that they are securely communicating with whom they want to communicate, and nobody else. A user of the framework must be able to get satisfying answers to the following questions:

- Is the person or service I am talking to really the one she claims to be?
- Is there somebody in the middle, relaying everything?
- How sure can I be about this?

Those and more questions translate into the realm of key management and trust calculations. In contrast to this, traffic authentication (message authentication codes and asymmetric signatures), covers only the most basic necessities on the communication side, and secure systems offer

required properties on the systems side. These two aspects will not be further discussed in this paper.

The most fundamental aspect of trust that is involved covers the participants (i.e. the peer authentication). All other trust relations, mechanisms and system components are created to support this fundamental one. In particular, Public Key Infrastructures (PKIs) are supposed to provide a satisfactory answer to above three questions.

Most currently deployed PKIs (with the notable exception of PGP, see section 3) use centralized and hierarchical models for their trust computations, accepting certain drawbacks coming with them. An interesting and viable alternate solution is, for some applications, the web of trust. This well-known model can be used to allow users to derive trust parameters without involving certification authorities (CAs) or the like.

The remainder of this paper discusses some issues with current PKI solutions, gives an overview over existing work, and then introduces the web of trust as an alternate solution to the peer authentication problem. The use of quantitative trust calculations is illustrated, and a viable heuristics to do fast calculations in a web of trust is given.

2 Hierarchical PKI vs. Web of Trust

Depending on the environment of the targeted user community, different needs exist, which lead to different schemes for key management.

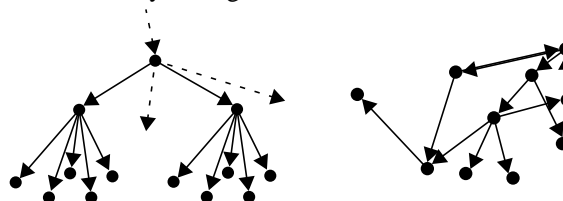


Figure 1: Hierarchical PKI vs. Web of Trust

Figure 1 illustrates the principal difference of two solutions, namely the rigid structure of the traditional hierarchically organized PKI, in contrast to the flexible, distributed nature of the web of trust, which makes it at the same time a more generic solution, and potentially more difficult to manage.

In the following subsections, three usage scenarios will be explored, leading to different requirements for a PKI.

2.1 The Hierarchical Organization

Assume that the infrastructure of a large bank or industry is to be made secure. All participants have the need to authenticate their communication partner, and to transfer data privately over a public infrastructure. At the same time, they are embedded in an environment where rules of trust and dependencies are well specified. Thus, a central certification authority (CA) or a small hierarchy of those can be employed to provide certificates to all communicating peers. The peers can easily verify their authenticity and thus communicate securely.

Naturally the situation becomes more difficult when several usually competing entities, each owning its own certification hierarchy, want to enter a limited collaboration, and use each others certificates to secure communications.

2.2 The Divided Community

This is an actual and interesting scenario. Everybody who has established a link to a Web server using 'secure' http has experienced it. Customers of an on-line service usually want to be sure that they are talking to the real service provider, and not an impostor. On the other hand, the service provider does not have an immediate need for the authentication of the customers. They authenticate themselves e.g. by giving appropriate payment information. As a result, this scenario can be modeled as representing two distinct sets of entities. The first set provides services, and owns well known identities, the second set uses the available services, and wants to authenticate the service provider.

There currently exists a set of commercial certification authorities, whose public keys have been made widely available, and where service providers can buy certificates (e.g. for their web servers). Potential customers can retrieve such certificates containing public keys from the service providers and verify the correctness using the public key of a commercial CA which they have received by out-of-band methods.

In this scenario, several issues can be observed. Firstly, service providers tend to become dependent on the CA services, having to buy new certificates regularly. Secondly, users need to get root certificates through a trusted channel — for now this is simply the distributed software (i.e. netscape) itself. Tampering with the software, malevolent configuration, or malevolent proxies can easily lead the user to believe things that are not what they seem to be. Additionally, it is quite hard for the average user to verify the correct configuration of his browser.

One may also be concerned about the strong segregation into service providers and service users that has evolved in recent years. It is hard for a user to gradually become a trusted and authenticated service provider,

instead he has to invest substantial effort and/or money to make the switch from one class to the other.

2.3 The Unstructured User Community

Last, but not least, individuals that do not pertain to one organization, and are not embedded in a certification hierarchy may still want to encrypt and authenticate their communications. They can achieve this by using traditional key agreement methods such as Diffie-Hellman, or use a public key infrastructure that represents a web of trust (e.g. PGP). This gives the communicating peers privacy, and authenticity, if the correctness of the employed public keys is verified out-of-band.

Again, to do this correctly can be hard for the average user, and it is a problem to create globally valid models to establish the same amount of trust into differently verified public keys. Additionally, each user needs to hold more data than in the hierarchical PKI scenarios.

2.4 Summary

Key management schemes that want to reach maximum deployment may have to provide mechanisms that suit all three of those scenarios and the policies associated with them. Public keys, as managed by a public key infrastructure (PKI), are bound to identities by CA's (which may in themselves be normal users of the system), must be easily retrievable, replaceable and revocable. Only this is the function of a PKI; whether it be hierarchically organized or based on a web-of-trust architecture.

The advantages of a web-of-trust based PKI are the possible (but not mandatory) genericity of trust relations, the ability to fit whatever organizational structure is present, and to easily evolve into other layouts when the users needs change. In the special case of webs of trust that map to hierarchical systems, trust calculations become easy, and the system shows the same properties as purely hierarchical PKIs, with more flexibility as a bonus.

3 Related Work

The primary issue of existing PKI solutions is to establish trust in the identity of the communication peers. Current trust relationships either work in a hierarchical manner ([X.509] for example), with one or more certification roots from which all participants derive their respective trust, or by using the model of web of trust, as it was introduced, for example, by PGP [PGP95], or, sometimes, using a mixture of the rigid hierarchies and freely interconnecting certifications. A formal definition of trust relationship networks and the quantification of trust can be found in [Maurer96a]. Although the approach given there is the most comprehensive formal and understandable solution to date to the expression of trust, the calculation of trust values is very expensive ($O(n^2)$). Additionally, a means of binding trust values to real-world authentication

processes (such as passport verification) is not yet agreed upon.

The expression of trust into prospective peers is but the first step in establishing a secure connection to this peer. As a next step, the peer needs to be authenticated, using one of many possible protocols that achieve this, see also [Caronni96]. For an overview of authentication in distributed systems, see [Liebl93]. For an in-depth discussion of other issues related to flexible and secure communication frameworks see [Caronni99], and for group communication key management issues, see [Waldvogel99]. [Maurer96b] gives an extended formal definition on the theory of authentication, and [Maurer94] introduces a intuitive and comprehensible method to formally express the state of associations. A selection of architectures that deal with the issue of peer authentication will now be explored.

3.1 PEM, PGP and Verisign

Privacy Enhanced Mail (PEM) [RFC1421] and Pretty Good Privacy (PGP) [PGP95] are two packages that provide mail security. Verisign provides certificates to WWW servers and clients. They all use asymmetric cryptography (RSA or Diffie-Hellman based) to provide peer authentication. Although electronic mail and web access seem unrelated, participants use the same mechanism (signed certificates) to insure peers of their respective identities. PEM and Verisign choose to limit themselves to the use of traditional certification hierarchies, PGP allows the establishment of a web of trust and unlimited cross certification.

3.2 Secure DNS

The Domain Name System (DNS) [RFC1033], [Albitz92] is a critical component of the Internet. It is implemented as a distributed database on a global scale, provides the translation of domain names to Internet addresses and also performs diverse other functions, *e.g.* the mapping of mail addresses to mail handling agents. Recently, an extension to the DNS that allows for data authentication through digital signatures and key distribution has been proposed [RFC2065]. The proposal expects that in most cases a single private key (the “zone key”) will generate all the signature records for a given zone. The zone would act as its own CA, its authority coming from its super-zone. The keys additionally stored in the DNS are accompanied by information such as for what to use the key, and a signature of the certifying authority. The secure DNS allows cross-certification between zones, eliminating the need for a root key. Such cross-certification is impractical on a large scale, but it does allow for an organization with several domains to set up a secure DNS in the absence of keys for the higher-level domains. The secure DNS extensions are new and time will tell whether they succeed in providing a general public key infrastructure that does more than add security to the DNS database.

3.3 Kerberos

Kerberos [RFC1510], [Neuman94] is a system of trusted entities, offering authentication to single users on unsecured workstations. The system relies on initial authentication, where the trusted entity provides user’s workstations with a credential ticket. This ticket may be used to request services from servers. Linking together different Kerberos-realms is possible by cross certification. Open problems are the limits imposed on the infrastructure (either a trusted computing base is employed, or only one user may be allowed to have access to a workstation at a time) and the delegation of rights. The usability of the system is somewhat reduced because all remotely operating applications must be made Kerberos-aware; this drawback is taken into account in favor of a more secure environment.

3.4 X.500

Within the OSI infrastructure, the directory service X.500 [X.500] is an important element. A general authentication framework for this service is specified in X.509 [X.509], together with a certifying hierarchy. Authorization issues are also considered by the X.500 recommendations, principally in X.501, Annex F [X.501]. Most components are only roughly sketched out, and details for implementation purposes of many mechanisms are at the care of the implementors. The certificate data format that has been defined by X.509 is coming into widespread use as a vessel to create interchangeable certificates.

3.5 Authentication and Trust

As illustrated by above subsections (and several more solutions exist), authentication and trust frameworks abound, and no basically new aspects are to be added to the field of knowledge in this area. In contrast to the basic aspects, practical Web of Trust computation offers space for improvement. One area of innovation specifically lies in the method how trust into a peer is calculated, and this is the focus of the coming sections.

4 The Web of Trust

Concepts and ideas presented here are, in much more complete form, elaborated in [Maurer96a]. The goal of the presentation in this subsection is to shortly familiarize the reader with the necessary concepts for web of trust calculations, and to introduce some trivial simplifications to Maurer’s model, as later implemented by the heuristic approach in section 5.

Assume the existence of a public key infrastructure, where a global database holds public keys of participants, together with claimed identities and signatures thereon, performed by other participants. This very well matches the current state of the PGP public key server infrastruc-

ture. The result is a web of trust where key holder can make publicly known whose keys they trust to be authentic. As a member of this infrastructure, one can decide whom to trust as an introducer of new keys, to a lesser or stronger degree. If the resulting web of keys and trust relationships allows one to establish a link between oneself and the target identity, then communication can take place. The general concept of a web of trust can be made to fit any certification hierarchy model, or it can be used without additional constraints, leading to a grass-roots approach.

The next logical step is to assign to each link in the web of trust a probability indicating its correctness. Thus finding the total trust from a starting point towards a desired key becomes a series of random experiments, and, at least in the model, is governed by probabilistic laws.

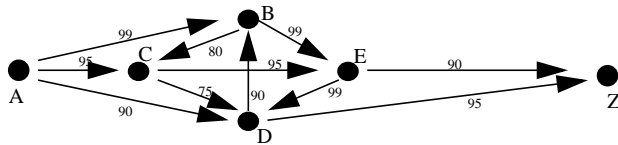


Figure 2: Sample Web of Trust

Figure 2 gives an illustration of such a web of trust. Point A is represents the user Adam as starting point, and Z (or Zacharias, if you want a name) is the target point. This translates as user A holding the keys of B, C, D, E, and Z, having signed keys B, C, and D himself, and wanting to communicate with Z. The values in the figure represent trust invested in links, represented as percentile values, or in other words, a value bound to a signature, assigned by the signer, and indicating how high an probability he assigns to the signee really being whom he claims to be.

This promptly gives raise to some concerns. For example, for such a model to be valid, the way trust is determined must be standardized, as to make clear what a probability value actually means. Probability values could be assigned to a close acquaintance being whom she claims to be, or to a stranger whose passport and drivers license have been inspected being whom he claims to be. Methods to derive such values must be globally known, and, if there are different ones, conversion functions have to be assigned.

Additionally, one must not only examine the value of trust bound to individual links, but also the trust one must invest in intermediate nodes to introduce third ones. While this aspect is not further covered here, it is very important to make a valid model, please see Maurer's paper for a complete treatment. This aspect has been neglected here as it does not significantly add to the complexity of the technical problems that are to be solved.

In the next section both the theoretically correct (but expensive) method, and a cheaper heuristic method of calculation on the simplified model will be introduced.

5 Heuristics for Trust Calculations

To further illustrate the behavior of trust in remote peers, some simple scenarios will be visited first:

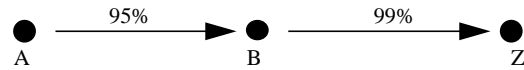


Figure 3: Serial Trust Path

Figure 3 shows a very reduced web, containing only one path with one intermediate entity. Trust in the final destination decreases when more entities lie between the start and the target node. In terms of probabilities, the resulting trust (about 94%) is the result of the product of the separate trusts. In contrast to serial trust, a scenario

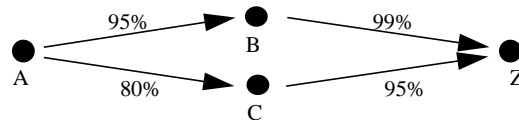


Figure 4: Parallel Trust Paths

with parallel paths, such as the one in Figure 4, is supposed to lead to increased trust in the destinations authenticity. Again, this translates nicely to probabilities, since the destinations key can be used if at least one of the paths leading to it is correct. If there are more, it is for the better. This reinforcement may be treated like independent probabilities in statistics. Thus A is sure about the identity of Z with $T_{AZ} = 1 - (1 - 0.99 \cdot 0.95)(1 - 0.80 \cdot 0.95)$, that is about 98.6%. Up to now, the examples would allow for the iterative calculation of trust (this is what the heuristic approach tries to achieve, see below), but some more peculiarities have to be accounted for first.

In the real world, a web of trust will contain loops and interdependent paths, as illustrated in Figure 5 and Figure 6. In fact, the example with interconnected paths is

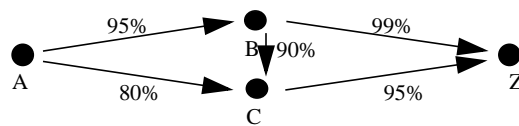


Figure 5: Interconnected Paths

the most simple scenario where all heuristic approaches fail to correctly model the theoretically correct approach. If one refrains from using a purely probabilistic model, different ways of reasoning would be possible: The optimistic approach would be to assume that B reinforces A's trust in C (like 'Parallel Trust Paths' above). The pessimist could argue that B could use C (without its knowledge) to reinforce As trust in Z. But that is okay, as C believes in Z independently of B (but then again, what if they do cooperate?).

There are different ways to look at this scenario: B reinforces A's belief in C and/or C reinforces B's belief in Z. Some links in the path are used more than once, dooming iterative processing.

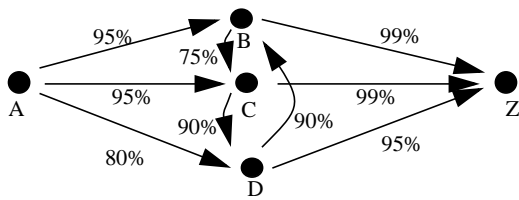


Figure 6: Looping Path with Intermediate

Figure 6 represents an even more general case of trust webs, where arbitrary complex loops are bound to occur.

Now, how is such a web of trust evaluated using only probabilistic laws? In fact quite easily. One must consider each event (of a link either being there, or being absent, with the given probability), and the sum of random experiments where all those events are combined obviously has the probability of 1. From this universe, all experiments where at least one path from the start to the target remains intact are noted, and their probability is added. The resulting value indicates the trust that the starting point can invest in the target really being whom he is supposed to be. In pseudo-code the algorithm looks like this:

```

Input: Web w, Node start, Node target
Set result to 0.0
Treat links in the web as states in a binary vector
Set the vector to 'all 1', meaning all links are up
While the vector is not all zeros do
    If a path from start to target can be found
        Calculate product of all 'up' and 'down'
        probabilities
        Add product to result.
    Decrement vector by one, thereby enumerating
    all possible states
Return result

```

As can easily be seen, the algorithm is of exponential complexity. Evaluation time explodes with the size of the web, and a well connected web with more than a few hundred participants certainly evades computability in the foreseeable future. To increase efficiency, one may collate all purely sequential elements in a first pass, and all purely parallel elements in a second pass, until neither of the passes can achieve a simplification, see also the heuristic approach below.

As an alternative, different (some non-probabilistic) interpretations of the web of trust have been sought:

- The worst path: Only consider the worst available path leading from start to target. Nonsensical, since much crucial information is discarded. This certainly is the pessimistic approach.
- The best path: Even when only the trust over the best available path is computed, discarding parallel paths leads to significant distortion of the result. Even optimists would suffer.
- Find all independent paths and calculate their mean value: This keeps some of the information provided by the web of trust, and simplifies it sufficiently to make the computational solution easily feasible. It is not clear how taking a mean of different paths would relate

to reality.

- Hull: Select the best path from start to target, calculate and remove it. Repeat until no more paths can be found. This gives you all possible independent paths, which are then combined via their distrust. Hull makes sense if someone wants to get an result which does neither favour nor punish cooperation of intermediate links. Efficient calculation is given.
- All: Calculate the probability of all possible paths. Combine them via distrust. Not very useful, as the same link may be used for different evaluations. The resulting values that are combined via distrust are not independent, which is bad. You break fundamental rules of probability calculus by doing this.
- Cooperative ('Coop'): This is like 'Int' below, with one addition. Starting from the neighbours of A you record the nodes passed on the paths. If two paths come together later on, and if they share earlier nodes you average them instead of using the reciprocal value. This actually weakens paths were the partners trust each other.
- Interleaving ('Int'): Use iterative processing, first finding the trust to direct neighbours from the start node, and the build on these results to finally reach the target node. As long as the web of trust suffices certain laws of non-interdependence, the result is equal to the theoretically correct one – thus making it an acceptable heuristic. The approach is fast and has been found to be of practical use. Its pseudo-code description follows:

```

For all direct neighbours to target node Z do
    If direct neighbour X is a start node
        Multiply trust into X with the trust into
        the link from X to Z
        Calculate new trust into Z as:
        Z := 1-(1-X*link)(1-Z)
    Else
        Create a copy of the web
        Drop target node Z in the copy
        Designate node X as new node Z in the copy
        Recurse in the copy
        Add returned value to trust into Z as:
        Z := 1-(1-return)(1-Z)
    Endif
Endfor
Return resulting trust in Z

```

It is important to note that the initial trust in the target node is initialized as 0 and the trust in the start node is set to 1. Creating a copy of the web that is used for the recursion assures that the computational results to derive the trust in one neighbor do not influence the trust in any other neighbor of Z. Even though the algorithm starts close to Z, it will first compute the trust in direct neighbors of the start node, due to the recursion, and from there move on to derive the trust into their neighbors, and so on until the target node is reached.

The examples given above, and additionally Figure 7 computed with both approaches yield the following trust values:

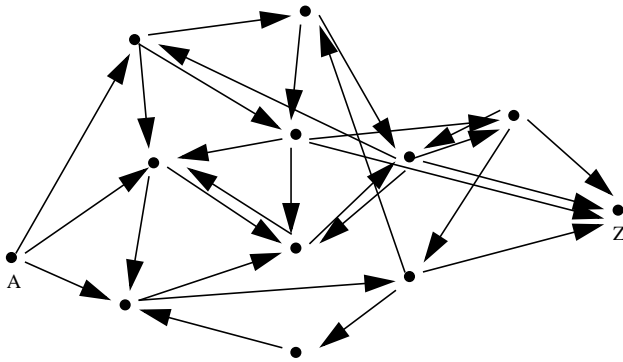


Figure 7: Complex Scenario (all links set to 90%)

Table 1: Web of Trust Computation Results

Scenario	Complete	Heuristics
Figure 2, example scenario	0.99486	0.99494
Figure 3, serial paths	0.94050	0.94050
Figure 4, parallel	0.98572	0.98572
Figure 5, interconnected	0.98734	0.99538
Figure 6, loop	0.99944	0.99997
Figure 7, complex scenario, uniform probability, 90%	.0.99721	0.99985

As is well visible, results differ in some scenarios. The lower the absolute link probabilities are, and the more ‘interconnected’ paths exist, the stronger the deviations are, the heuristic approach usually being more optimistic than actually granted. It is possible to detect in advance when the heuristic approach leads to wrong results (every time when the direct neighbors of the starting node can not be computed without re-assigning them target node status), and it is also possible to mix both calculation methods to minimize errors. Ideally, the problem would be simplified by using the heuristic approach, and then be segmented such that the resulting segments can be computed with the correct algorithm.

6 Summary

A variety of issues connected to trust and authenticity has been examined. The focus was placed on fast of web of trust computations.

For web of trust computation, a heuristic has been presented that allows computation in subexponential time (depending from the makeup of the web of trust, this can be linear cost), in contrast to preexisting algorithms with exponential cost. As a severe drawback, the found solution is not exact in all cases, but tends to be overly optimistic when trust paths contain looping structures or are otherwise interconnected. Both algorithms were implemented, and tested on small web of trust configurations.

Availability

Source code for trust computations (both Maurer-style and heuristic) is available, just send email to <gec@acm.org> with a subject of ‘web of trust source’.

Acknowledgments

Many thanks go to Dr. Daniel Bauer (now at IBM), Prof. T. Plagemann and the UNIK in Oslo, Prof. B. Plattner, and Prof. U. Maurer, both at the ETH Zurich, for interesting discussions during the evolution of this work.

I am also very thankful for the critical review that Dr. Susan Landau gave this paper.

References

- PGP95: P. R. Zimmermann: The Official PGP Users Guide; MIT Press, Boston, Massachusetts, U.S.A., 1995.
- Maurer96a: U. Maurer, “Modelling a Public-Key Infrastructure”, ESORICS’96, 1996.
- Maurer94: U. M. Maurer, P. E. Schmid, “A Calculus for Secure Channel Establishment in Open Networks”, ESORICS ‘94, Nov. 1994.
- Maurer96b: U. Maurer, “A Unified and Generalized Treatment of Authentication Theory”, LNCS 1046, pp. 387-398, STACS 96, Springer Verlag, 1996.
- Liebl93: A. Liebl, “Authentication in Distributed Systems: A Bibliography”, Operating Systems Review 1993.
- RFC1421: IETF, “Privacy Enhancements for Internet Electronic Mail (Part 1)”, Internet Engineering Taskforce RFC 1421, 1993.
- RFC1033: IETF, “Domain Administrators Operations Guide”, Internet Engineering Taskforce RFC 1033, 1987.
- Albitz92: P. Albitz, C. Liu, “DNS and BIND”, O’Reilly & Associates, 1992.
- RFC2065: IETF, “Domain Name System Security Extensions”, Internet Engineering Taskforce RFC 2065, 1997.
- RFC1510: IETF, “The Kerberos Network Authentication Service (V5)”, Internet Engineering Taskforce RFC 1510, 1993.
- Neuman94: B. C. Neuman, T. Ts’o “Kerberos: An Authentication Service for Computer Networks”, IEEE Communications Magazine, v. 32, n. 9, pp. 33-38, Sept. 1994.
- Caronni96: G. Caronni, H. Lubich, A. Aziz, T. Markson, R. Skrenta, “SKIP – Securing the Internet”, Fifth Workshop on Enabling Technologies (WET ICE ‘96), 1996.
- Caronni99: G. Caronni, “Dynamic Security in Communication Systems”, doctoral thesis no. 13156, ETH Zurich, 1999.
- Waldvogel99: M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner: “The VersaKey Framework: Versatile Group Key Management”, IEEE Journal on Selected Areas in Communications, vol 17 no. 9, Septemeber 1999.
- X.500: CCITT, “X.500 Directory Services”, Blue Book, VIII.8, Geneva, 1988.
- X.501: CCITT, “X.501 Authorisation”, Blue Book, VIII.8, Annex F, Geneva, 1988.
- X.509: CCITT, “X.509 The Directory – Authentication Framework, Blue Book, VIII.8, Geneva, 1988.