A Multiple Signature Based Certificate Verification Scheme

Albert Levi and M. Ufuk Çağlayan
Boğaziçi University, Department of Computer Engineering
Bebek, Istanbul 80815
levi@boun.edu.tr caglayan@boun.edu.tr

Abstract
In this paper, we proposed a formal representation of certificate validation in Pretty Good Privacy (PGP) and X.509 systems. This representation uses new logical assertions to support public-key based certification systems and different trust levels. Although the meanings of some of those assertions are different in PGP and X.509 cases, the certificate validation can be expressed using the same assertions. We also proposed a novel multiple digital signature scheme, namely *nested signatures*, which is directly applicable to PGP and X.509 certificates to yield *nested certificates*. A nested signature is a signature over another signature and is used to verify the subject signature without using the public key of the issuer of the subject signature. This characteristic of the nested signatures and nested certificates allow the entities in the network to realise more flexible trust and certification scenarios. We also extended the formal specification of the certificate validation to support nested certificates for both PGP and X.509 cases.

1. Introduction

Public key based cryptographic techniques have become common practice in network security and authentication applications in the last decade. One of the most important reasons behind this popularity is the ease of key distribution in public key cryptography. The public-private key pairs are created by the owner of the key and the public keys can be known by everyone. That characteristic of public key systems makes the key distribution less difficult as compared to private key based systems. However, there are still some public key distribution problems in public key based systems.

There are two types of problems regarding the public key distribution: (i) the media for the distribution, (ii) authentication of the public key owner. Employing some global directories that contain the public keys and related information has solved the first problem. International Telecommunications Union (ITU) authentication framework recommendation (X.509) [1] suggests that X.500 directories can be used for public key publishing. Pretty Good Privacy (PGP) [2] uses public key servers, which contain the public keys along with the personal information. There are several PGP public key servers worldwide each holds the same data and runs synchronously.

The second problem related with the public key distribution is more important than the first one. Since the public-private key pairs are created by the owners themselves, there should be a mechanism to introduce them as valid entities to other partners of the application. The rationale for this requirement is to avoid name spoofing. A public key owner may introduce herself with a different name and the other partners of the application deem her as she is the claimed person, however she is not the person that she says so.

The general practice to avoid name spoofing is to use some trusted people to introduce newcomers to the system. Introducers digitally sign the public key - real identity binding of a person. The application partners verify the signature of the introducer over that binding and make sure about the real identity and the valid public key of the newcomer. Such a system can work if and only if the introducers are trusted entities for the verifier, because otherwise the verifier cannot comment on the signature of the introducers on the newcomer's identity - public key binding. This signed information which contain the public key of a person with

some managerial and identity details is called the *certificate*, and the introducers are called *Certification Authorities* (*CA*) in X.509 [1] terminology. PGP [2] uses the term *public key signature* and *introducer* for *certificate* and *CA* respectively. We will follow the X.509 terminology in this paper, since that terminology is widely accepted.

The contribution of this paper is twofold. First, we proposed a formal way to represent certificate validation in PGP and X.509 systems. This formal representation is a logical proposition in which the assertions are defined using some novel trust and certification related operators. Our aim to use a formal way to represent certificate validation is somehow to outline the certificate validation steps. We strongly believe that such a formal specification will further help the researchers to analyse the trust in the certification models.

Second, we proposed a multiple signature and a multiple certification scheme (*nested signatures* and *nested certificates*, respectively). A nested signature is a signature over the signature part of a signed message. By this way, the nested signer signs the first signature but does not sign the original message. However, if she wants, she may also sign the original message using the traditional digital signature mechanisms. This distinction between message signatures and signature (nested) signatures provides flexibility to the nested signer to choose her valid entities. If she believes that the message is authentic, then she signs the message by issuing a traditional digital signature. On the other hand, if she believes that the first signature over the message is authentic, then she signs only that signature by issuing a nested signature. Such flexibility also applies to the verifier of the signatures. If the verifier does not know the public key of the first signer but knows the one of the nested signer, then he can easily verify the message by verifying the signature due to nested signer. Actually, some trust considerations apply here that will be explained in later sections. We have also extended the formal certificate validation representation for nested certificates.

Section 2 gives an overview of the organisations of the certificates in a global network. Section 3 describes the certificate validation and trust management of PGP and X.509 systems and proposes a formal representation of certificate validation. Section 4 is about the nested signatures. Section 5 explains the nested certificates and their effects in certificate validation in a formal way. Section 6 discusses generalisation of nested certificates and some implementation issues. Section 7 and Section 8 are the conclusions and the further work respectively.

2. The Network of Certificates

A single CA may suffice for small and centralised networks, but for large and distributed networks, like the Internet, several CAs should take place in some order. There are different approaches for the global organisation of the CAs. The topology of the CAs and the end users (users whose public keys are certified by CAs) is called the *Public Key Infrastructure* (*PKI*).

Before examining current trends in PKI design, it is worthwhile to describe the *trust* concept in PKI. In order to determine the legitimacy of a public key of a newcomer, the verifier should trust the CA who has issued a certificate for the newcomer. The verifier should trust the CA concerning that the CA is an honest entity and does not issue certificates to people who are unknown to CA. Although some researchers [7], [8] are not so comfortable by such an assumption, this is the only way for the verifier to make sure about the public key of the newcomer in a PKI based system. There are different interpretations of trust in PGP [2] and X.509 [1] based PKI schemes. Those interpretations will be discussed later.

Here, it is vital to point out that the relationship between a CA and his certified entity can not be considered as trust. A CA does not guarantee all of the behaviours of the entity for which

he issued a certificate. He only guarantees the identity-public key binding of the certified entity. In order to give such a guarantee, a CA must make sure about the validity of the information that he signs before certification. Thus, the relationship between a CA and his certified entity is *validation* rather than trust.

Although the X.509 recommendation does not specify and enforce any topology for a standard PKI, the general characteristics of any X.509 based PKI are almost the same. The reason for this is the X.509 standard certificate structure. There are some X.509 based PKIs proposed like Privacy Enhanced Mail (PEM) [3], Public Key Infrastructure for X.509 certificates (PKIX) [4], ICE-TEL [5], [6]. The general characteristics of a X.509 based PKI are: (i) strict distinction between a CA and the end user, (ii) tree hierarchy of CAs, (iii) forming optional CA networks via cross certificates (not applicable for PEM). A typical X.509 PKI is given in Figure 1.

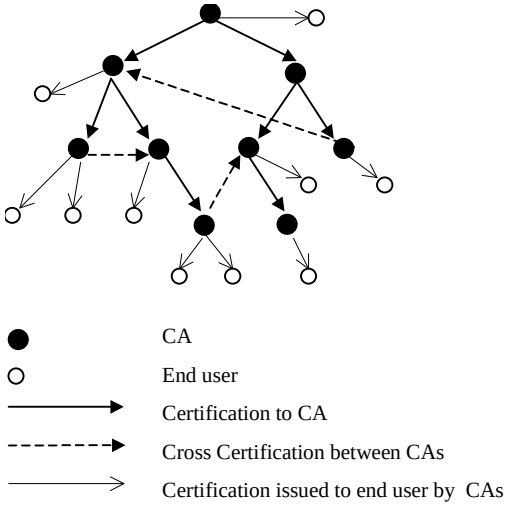

Figure 1 A typical X.509 based PKI

In X.509 based systems every CA is a potentially trusted entity, but in order to verify a certificate of a target, the verifier has to know the legitimate public key of the CA of the target. To do so, the verifier gets the certificate of the CA of the target and to verify that certificate she has to know the public key of the CA of the CA of the target. This recursion goes on until the verifier encounters a CA of whom she knows the public key. Such a mechanism yields a *certification path*, which starts with a trusted CA for which the verifier knows the public key and ends with the target entity. In this path each component certifies the next and the verifier trusts each CA in the path. In X.509, there are some mechanisms to avoid "blind trust" to CAs. The details of the X.509 trust mechanism will be explained in the next section.

The PKI incorporated in PGP is the only example of its type. PGP does not conform to any standard, therefore its PKI is typical. In PGP PKI, every node can be CA, therefore there is no CA - end user distinction. Therefore, the PKI of PGP is called the "web of trust". The phrase "web of trust" is due to Phil Zimmermann [2], [9]. However, the word *trust* in that phrase is misleading, because, as described above, the relationship between the CAs and the certified entities is *validation*, not *trust*. Thus, it would be better if the PKI of PGP has been called "web of validation", but for the sake of consistency in the terminology between this paper and

its references, the term “web of trust” will be used. The importance of PKI of PGP is being a *web*, i.e. it is a directed graph in which each node is a PGP key owner (associated with a public key) and each arc is a certification relationship between two users. The source of the arc is the CA and the target is the certified user. A typical “web of trust” is given in Figure 2.

The trust model in PGP “web of trust” takes care of personal freedom more than the X.509 based PKIs. In PGP everyone is free to choose her own trusted CA, since there is no predefined hierarchical CA topology that a user should accept. The basic consideration here is the fact that the only ultimately trusted entity for a PGP user is herself and each certification path starts from herself. However, a PGP user may assign a level of trust to another PGP user to introduce other PGP users to her, but this trust is not transitive. That means, a PGP user trusts the validity of the certificates issued by the trusted CAs, but does not trust those trusted CAs to introduce other CAs. The reason behind this is not a design or implementation bug, but to give enough freedom to a PGP user to choose her own trusted parties. The lack of trust transitivity does not mean that the length of a certification path from a user is at most 2, The user (call the *verifier*) may find a longer certification path from herself to the user that she wants to validate (call the *target*). In this path each entity certifies the next, but, unlikely X.509 based systems, in order to verify the target, the verifier must assign enough trust to all of the intermediate CAs beforehand. The details of the PGP trust mechanism will be explained in the next section.

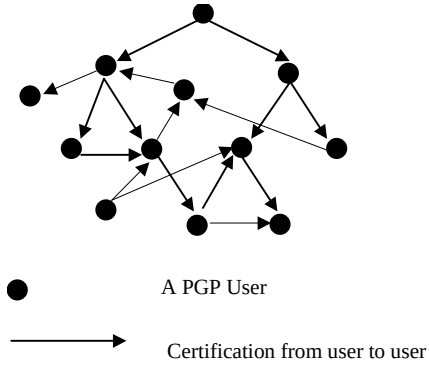

Figure 2 A Typical PGP “web of trust”

3. Verification of a Certificate

In order to proceed in the certification path, the verifier should be able to verify the validity of each certificate in the certification path. This section explains the logic behind the principles for the verification process for a single certificate in X.509 and PGP models.

The basic and common principle for certificate validation is the legitimacy of the signature over the certificate content. That signature can be interpreted as the warranty of the signer for the integrity and the correctness of the information content of the certificate. The verifier uses the public key of the signer to check for the signature. However, having a valid signature over a certificate does not suffice to qualify that certificate as valid, because the verifier may have some extra personal privacy requirements. Those requirements are mostly related with the *trust* concept. PGP and X.509 schemes have different approaches for trust manipulation. The trust models incorporated in PGP and X.509 will be detailed in the next two subsections.

In both PGP and X.509 based PKIs, the length of the certification path can be constrained. Since this topic is outside of the scope of this paper, we will not give the details on the ways of doing this. The reader may refer to [1], [2] and [9] for more information.

3.1. Verification of a Certificate in PGP and PGP Trust Model

In PGP, each user assigns a trust level to each CA. PGP supports 4 trust levels: *complete*, *marginal*, *untrusted* and *unknown*. Multiple CAs may sign a certificate content, but each signer signs the raw content of the certificate, not together with other signatures. Therefore, these multiple signatures are called *independent signatures*. The certificate content is stored together with all of its signatures. In order to validate a certificate, the verifier looks for either one legitimate certificate issued by a *complete*ly trusted CA, or two legitimate certificates issued by two *marginal*ly trusted CAs. The numbers one and two are default values and can be adjusted by the verifier. The current implementation does reject certificates issued by *untrusted* and *unknown* CAs.

More formally, let A and B be two PGP users with associated private and public keys. The assertion $A \multimap B$ means that A *knows* the correct public key of B and therefore can check the signatures of B. The assertion $A \rightarrow B$ means that A has *issued a certificate* for B, in other words A has signed the public key - identity binding of B. The assertion $A \Rightarrow B$ means that A *completely trusts* B to introduce another entity so that A can verify the public key - identity binding of another PGP user for which B has issued a certificate. Similarly, the assertion $A \gg B$ means that A *marginally trusts* B as a CA. The assertion $A \rightarrowtail B$ means that A has *verified* the public key - identity binding of B using the above assertions and PGP certificate verification procedure. The function $\mathtt{COUNT}(X|\mathbf{P})$ returns the total number of distinct PGP users X, who satisfies the logical proposition **P**. The verification procedure of PGP can be represented formally as the following logical proposition, where V is the verifier user, T is target to be verified, A is any intermediate CA, ct and mt are the total numbers of completely trusted and marginally trusted CAs necessary to qualify a certificate as valid respectively (A and V may be the same user, if V has issued a certificate for T).

$$(\mathtt{COUNT}(A \mid V \multimap A \ \mathtt{AND}\ V \Rightarrow A \ \mathtt{AND}\ A \rightarrow T) = ct)\ \mathtt{OR}\ (\mathtt{COUNT}(A \mid V \multimap A \ \mathtt{AND}\ V \gg A \ \mathtt{AND}\ A \rightarrow T) = mt)\ \mathtt{IF\ AND\ ONLY\ IF}\ (V \rightarrowtail T)$$

Although the certificates are public, the trust information is kept in the local databases of the PGP users, since that information is considered as a private information. However, in X.509 the trust information is somehow written in the certificates.

3.2. Verification of a Certificate in X.509 and X.509 Trust Model

The basic principle of certificate verification in X.509 is the existence of a legitimate and trusted signature over the certificate content, as in PGP. However, the trust information to CAs is kept in a different way. Trust information became an optional extension in certificates starting with 3rd version of X.509 (X.509v3) [1]. Each X.509v3 certificate may contain a *policy identifier* that gives an indication to the verifier about the reliability of the certificate. The *policy identifier* is a value assigned by the CA of the issued certificate and implicitly explains how the CA has verified the certified entity and how reliable is the CA for certificate issuance. Several CAs can use the same *policy identifiers* in their certificates if their certification policies are similar. On the other hand, a particular CA can use several *policy identifiers* if that CA issues different types of certificates based on different policies. For more information about policies and *policy identifiers* the reader may refer to [1], [5], [6].

The verifier of a X.509v3 certificate initially determines a set of *policy identifiers* (called *allowed policy identifiers*) and enforces the verification process to verify only the certificates that hold a *policy identifier* which is in the *allowed policy identifiers* set. By this way, the verifier specifies the class of trusted certificates and CAs. Since the trust concept is transitive in X.509, the verifier is also bounded by the *allowed policy identifiers* for the trusted CAs in the certification path. However, the policy identifier constraints of the intermediate CAs

cannot loose the initial constraints of the verifier. The policy and *policy identifier* concepts are actually the way of manipulation of trust in X.509v3 certificates. The advantage of such a scheme over PGP trust model is the fact that the verifier does not need to know the identities of the trusted CAs. She has to know only the trustworthy *policy identifiers*. Moreover, by this way the verifier delegates the trust to other CAs under surveillance. The disadvantages of the policy scheme for trust manipulation are the necessity of a mapping between *policy identifiers* and their trustworthiness, lack of CA based selection for trust assignment and lack of trust levels for the *policy identifiers* (a *policy identifier* is either trusted or not).

In X.509, in order to validate a certificate, the verifier checks if the *policy identifier* of the certificate is in the *allowed policy identifiers* set or not. If so, the verifier validates the public key - identity binding declared in the certificate. If not, the verifier rejects the certificate. Actually, by allowing a *policy identifier* in a certificate to validate, the verifier trusts the issuer of that certificate.

To formalise the X.509v3 certificate verification principle, we will use the same notation as in the PGP case. Let *A* and *B* be two users in X.509v3 based PKI with associated private and public keys. The assertion $A \multimap B$ means that *A knows* the correct public key of *B* and therefore can check the signatures of *B*. The assertion $A \rightarrow B$ means that *A* has *issued a certificate* for *B*, in other words A has signed the public key - identity binding of *B*. The assertion $A \blacktriangleright B$ means that *A trusts B* as a CA (that is, the *policy identifiers* in the certificates that *B* issues are in the *allowed policy identifiers* set of *A*) so that *A* can verify the public key - identity binding of another user for which *B* has issued a certificate. Finally, the assertion $A \twoheadrightarrow B$ means that *A* has *verified* the public key - identity binding of *B* using the above assertions and X.509v3 certificate verification procedure. By using the above assertions, the X.509v3 certificate verification proposition can be expressed formally as:

$(V \multimap A \text{ AND } V \blacktriangleright A \text{ AND } A \rightarrow T) \text{ IF AND ONLY IF } (V \twoheadrightarrow T)$

where, *V* is the verifier user, *T* is target to be verified and *A* is any intermediate CA.

In X.509, an entity may have more than one certificate as in PGP. Each CA signs the certificate without other signatures on it. Moreover, those certificates are signed by different CAs and probably with different *policy identifiers*. Therefore, those signatures are called *independent signatures*. Multiple certificates are useful to improve the verifiability of the certificate owner. There may be several verifiers with distinct sets of *allowed policy identifiers*. If a user owns different certificates with different *policy identifiers*, then she might be verified by those verifiers with distinct sets of *allowed policy identifiers*.

As can be seen easily, the assertions and the operators used to represent the PGP and X.509 certificate validation procedure are the same (except $\geqslant$, it applies only to PGP). We preferred to give the assertions separately for PGP and X.509 cases in order to specify the differences in the semantics of trust in both cases. We will continue to use the same notation throughout the paper.

4. A New type of Multiple Signature: Nested Signature

One can easily recognise from the previous section that multiple signatures are allowed for a certificate content. Those signatures are *independent signatures*, since each CA signs the raw certificate content and has no relation with each other. The certificates that are created using *independent signatures* over a certificate content are called *independent certificates*.

In this paper, we propose a new multiple signature type which is called *nested signature*. A *nested signature* is the signature over only the signature part of a message. The issuer of the

nested signature is called *nested signer*. An important point for a nested signature is that the nested signed part does not include the original message, but consists of only the initial signature over the message. By this way, the nested signer does not give any guarantee about the integrity and authenticity of the message, but warrants the integrity and the correctness of the first signature. On the other hand, the verifier, who checks the correctness of the nested signature by the public key of the nested signer, verifies the authenticity of the first signature over the message. Consequently, she verifies the integrity and the authenticity of the original message without literally verifying the first signature using the public key of the first signer. The only assumption made by the verifier is that both signers are trusted. Therefore, the verifier makes sure about the authenticity of the message by verifying the nested signature. Such a scheme is useful when (i) the nested signer does not know the message owner and unable to sign the message content, but knows the first signer and can sign the signature due to first signer, and (ii) the verifier trusts both signers but does not know the public key of the first one to verify the message.

5. Nested Certificates

The certificates with *nested signatures* on them are called the *nested certificates*, and the issuer of the nested certificate is called *nested certifier*. In nested certificates the signature due to first certifier over the certificate content is signed by the nested certifier. The raw certificate content is not signed in nested certificates.

The certificate issuance and validation scenarios from both the nested certifier and verifier points of views are as follows.

- The nested certifier knows the valid public key of the first certifier, but does not trust him as a CA. Therefore, she verifies the signature but cannot validate the user whose information is in the certificate. Consequently, she cannot issue an independent certificate for that user. However, since the nested certifier has verified the authenticity and the integrity of the first signature, she can easily issue a nested certificate by signing the signature part of the certificate. By issuing a nested certificate, the nested certifier guarantees that the first signature is due to first certifier and relays that information to people who verify her nested signature.
- The verifier trusts both the first and the nested certifiers. He also knows the valid public key of the nested certifier, but does not know the public key of the first certifier. Since the verifier does not know the public key of the first certifier, he cannot validate the certificate by checking the signature due to first certifier. However, he can validate the nested signature, because he knows the public key of the nested certifier. By the validation of the nested signature, the verifier makes sure about the authenticity of the first signature over the certificate content, because he trusts the nested certifier. Consequently, he validates the certificate and makes sure about the legitimacy of the public key - identity binding of the owner of that certificate, since he trusts the first certifier.

The trust concept in nested certificates can be interpreted in two different ways. On one hand, one may argue that the trust to the nested certifier is looser than the trust to the first certifier. Because, the trust to the nested certifier does not suffice to validate the certificate, the verifier should also trust the first certifier to validate the certificate. However, the trust to the first certifier would suffice for certificate validation, if the public key of the first certifier was known.

On the other hand, it may be argued that the trust to the nested certifier is tighter than, or the same as, the trust to the first certifier. Because, the trust to a nested certifier allows to validate

the initial certificate signature without literally verifying it and consequently, to validate the certificate. This situation can be interpreted as trust delegation. Therefore, the nested certifier must be more trustworthy than the first certifier, or at least, their trustworthiness's must be the same. In this paper, we will distinguish between the trust to the first certifier and the trust to the nested certifier in order to give enough freedom to the users about choosing one of the above interpretations based on their personal opinions.

5.1. Formal Representation

In order to represent formally the certificate verification procedure with nested certificates, two more assertions will be defined in addition to those defined in Section 3. Those new assertions are related with nested certificate issuance and trust to the nested certifiers. A trusted nested certifier is named as *Nested Certification Authority* (*NCA*). For the sake of simplicity, we will not consider trust levels for the NCAs. In other words, a NCA is either trusted or not.

In addition to the assertions and operators defined in Section 3 (namely the operators $\multimap$, $\rightarrow$, $\Rightarrow$, $\geqslant$ (only for PGP) and $\twoheadrightarrow$), two more assertions are defined for NCA processing in PGP and X.509 certificate validation procedure. Let *A*, *B* and *C* be three users. The assertion $A \overset{n}{\rightarrow} \{B \rightarrow C\}$ means that *A* has issued a nested certificate for the certificate that *B* had issued for *C*. In other words, *A* certifies the signature of *B* over the certificate content of *C*. This assertion also means that *B* has issued a certificate for *C*. It is important to point out once more that, the scope of the nested signature of *A* is only the signature of *B*, not the whole certificate. The assertion $A \overset{n}{\Rightarrow} B$ means that *A* trusts *B* as a NCA, that is *A* can validate any signature that is nested signed by *B*.

The formal certificate verification propositions with nested certificates will be defined for both PGP and X.509 cases separately in the next two subsections.

5.1.1. PGP

In order to represent the certificate verification process of PGP with nested certificates, we will use the COUNT function, which is defined in Section 3.1, and define another function, NCOUNT(*X*,*Y*|**P**), which returns the total number of distinct PGP user pairs (*X*,*Y*) which satisfy the proposition **P**. Using the above assertions and the COUNT and the NCOUNT functions, the certificate validation can be expressed as the following proposition, where *V* is the verifier user, *T* is target to be verified, *A* and *B* are any intermediate CAs, *ct* and *mt* are the total numbers of completely trusted and marginally trusted CAs necessary to qualify a certificate as valid respectively.

$$
\begin{aligned}
&(\mathtt{COUNT}(A \mid V \multimap A \ \mathtt{AND}\ V \Rightarrow A \ \mathtt{AND}\ A \rightarrow T) + \\
&\ \mathtt{NCOUNT}(A,B \mid B \overset{n}{\rightarrow} \{A \rightarrow T\} \ \mathtt{AND}\ V \multimap B \ \mathtt{AND}\ V \Rightarrow A \ \mathtt{AND}\ V \overset{n}{\Rightarrow} B \ \mathtt{AND\ NOT}\ V \multimap A) = ct) \ \mathtt{OR} \\
&(\mathtt{COUNT}(A \mid V \multimap A \ \mathtt{AND}\ V \geqslant A \ \mathtt{AND}\ A \rightarrow T) + \\
&\ \mathtt{NCOUNT}(A,B \mid B \overset{n}{\rightarrow} \{A \rightarrow T\} \ \mathtt{AND}\ V \multimap B \ \mathtt{AND}\ V \geqslant A \ \mathtt{AND}\ V \overset{n}{\Rightarrow} B \ \mathtt{AND\ NOT}\ V \multimap A) = mt) \\
&\mathtt{IF\ AND\ ONLY\ IF}\ (V \twoheadrightarrow T)
\end{aligned}
$$

5.1.2. X.509

In X.509, the certificate validation with nested certificates can be expressed more easily than the PGP case, since there is no trust level concept in X.509. By using the above assertions, the X.509 certificate verification proposition can be expressed formally as:

$$
\begin{aligned}
&(V \multimap A \ \mathtt{AND}\ V \Rightarrow A \ \mathtt{AND}\ A \rightarrow T) \ \mathtt{OR}\ (B \overset{n}{\rightarrow} \{A \rightarrow T\} \ \mathtt{AND}\ V \multimap B \ \mathtt{AND}\ V \Rightarrow A \ \mathtt{AND}\ V \overset{n}{\Rightarrow} B) \\
&\mathtt{IF\ AND\ ONLY\ IF}\ (V \twoheadrightarrow T)
\end{aligned}
$$

where, *V* is the verifier user, *T* is target to be verified and *A* and *B* are any intermediate CAs.

6. Discussion

Nested signatures and nested certificates can be extended to more general cases. Moreover, although we have used the same nested certificate assertions and operators for both PGP and X.509 cases, they have different meanings. Besides, in order to implement a nested certificate scheme in PGP and X.509, further investigations are necessary. In this section, we will briefly discuss these topics.

6.1. Hierarchical Nested Certificates

The nested signature concept can be adapted to the nested certificates also. One may want to sign only the signature part of a nested certificate. By this way, the signature over a nested certificate becomes certified. This type of nested certificates constitute a hierarchy of nested certificates, in which there are direct certificates to the user information, nested certificates to direct certificates as well as nested certificates to other nested certificates. It is worthwhile to repeat once more that, the nested certificates contain a nested signature only over the signature part of the signed certificate. This rule is also valid for hierarchical case. An example hierarchy is depicted in Figure 3.

To validate a user in such a certificate hierarchy, the verifier should find a certificate path towards the user info. That certificate path starts with a user for whom the verifier knows the public key and ends with the user to be validated. In such a path there may be several nested certificates. The verifier has to trust the issuers of the nested certificates in the certification path as NCAs and the issuer of the direct certificate in the certification path as a CA.

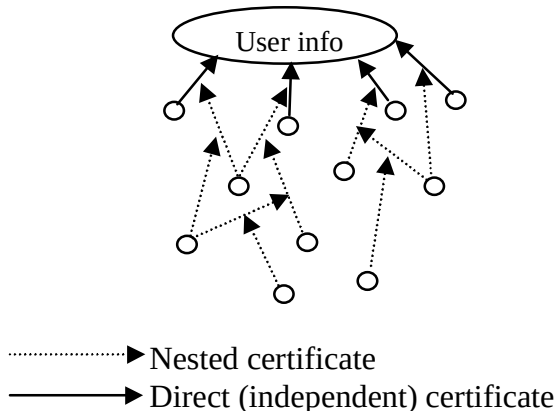

Figure 3 An example Hierarchy of nested and direct (independent) certificates

6.2. Implementation Issues

The nested certificates cannot be stored within the original certificate, because the signature of the first certifier of the certificate is sensitive to the raw certificate content. If some other data is put in the certificate after the first signature, then the integrity of the certificate spoils and that certificate cannot be verified. Therefore, the nested signatures must be kept in a different place, so that the verifier will be able to treat the nested certificates separately. In PGP, the independent certificates are appended to the corresponding raw certificate content. The nested certificates can be stored in a similar way in PGP. However, in X.509 the raw certificate content and the signature over it are stored in the same structure. There is no append mechanism to store multiple independent signatures over a certificate content and each X.509 certifier has to issue a new certificate which contains the raw certificate content and her signature. Since X.509 has no append mechanism, there must be a new structure and/or certificate type to store the nested certificates. Moreover, in each nested certificate, the information about the signer and the information about the subject signature must be kept.

The semantics of trust is different in PGP and X.509. In PGP, the trust to a CA is assigned by the verifier in different levels. However, In X.509, the trust information is embedded within certificates via policy identifiers (see Section 3 for details). The trust to NCAs may be assigned using the similar methods. In PGP, the verifier may assign a trust value to a NCA and there may be different levels of trust to NCAs. In X.509, a policy identifier, which gives an idea about the trustworthiness of the NCA, may be put in a nested certificate.

7. Conclusions

In this paper, we described the certificate verification schemes of PGP [2,9] and X.509 [1] and proposed a formal specification of certificate verification for them. We also proposed a new multiple signature scheme, namely *nested signatures*, which is applicable to PGP and X.509 certificates to yield *nested certificates*. A nested certificate is simply a certificate over only the signature part of another certificate. Nested certificates allow the verifier to validate another certificate without knowing the public key of the issuer of that certificate. We also extended the formal specification of the certificate validation to accommodate nested certificates for both PGP and X.509 cases.

8. Further Work

The nested certificate concept can be implemented in both PGP and X.509 PKIs. This implementation requires major changes in the designs of those systems. Some of the implementation issues have been explained briefly in Section 6, but the real implementation needs a deeper research.

Development of a trust model that accommodates the trusts to both CAs and NCAs is another further research orientation. Such a model can have trust levels for both CAs and NCAs. The certificate validation assertions and propositions should also be modified according to this trust model. Moreover, the formal specification of certificate validation scheme proposed in this paper can be generalised to support a generic trust model.

9. References

[1] ITU-T Recommendation X.509, ISO/IEC 9594-8, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, 1997 Edition.

[2] P. Zimmermann, PGP User's Guide Volume I: Essential Topics, available with free PGP software from `http://www.pgpi.com/download`

[3] S. T. Kent, Internet Privacy Enhanced Mail, Communications of the ACM, vol. 36, no. 8, pp. 48 – 60, August 1993

[4] R. Housley, W. Ford, W. Polk, D. Solo, Internet Public Key Infrastructure: X.509 Certificate and CRL Profile, Internet Draft <draft-ietf-pkix-ipki-part1-06.txt>, October 14, 1997

[5] D. W. Chadwick, A. J. Young, N. K. Cicovic, Merging and Extending the PGP and PEM Trust Models – The ICE-TEL Trust Model, IEEE Network, vol. 11, no. 3, pp. 16 – 24, May/June 1997

[6] ICE-TEL project homepage, `http://www.darmstadt.gmd.de/ice-tel/`

[7] Levi, M.U. Çağlayan, The Problem of Trusted Third Party in Authentication and Digital Signature Protocols, in the Proceedings of the Twelfth International Symposium on Computer and Information Sciences, ISCIS XII, pp. 317 – 324, October 1997, Antalya, Turkey

[8] Crispo, M. Lomas, A Certification Scheme for Electronic Commerce, in the Proceedings of Security Protocols International Workshop, LCNS vol. 1189, Springer-Verlag, pp. 19 – 32, April 1996, Cambridge, UK

[9] P. Zimmermann, PGP User's Guide Volume II: Special Topics, available with free PGP software from `http://www.pgpi.com/download`