DESIGN AND PERFORMANCE EVALUATION OF THE NESTED CERTIFICATION SCHEME AND ITS APPLICATIONS IN PUBLIC KEY INFRASTRUCTURES

by

Albert Levi

BS. in Computer Engineering, Boğaziçi University, 1991

MS. in Computer Engineering, Boğaziçi University, 1993

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of

Doctor

of

Philosophy

Boğaziçi University

1999

DESIGN AND PERFORMANCE EVALUATION OF THE NESTED CERTIFICATION SCHEME AND ITS APPLICATIONS IN PUBLIC KEY INFRASTRUCTURES

APPROVED BY:

Prof. M. Ufuk Çağlayan (Thesis Supervisor)	.....
Assoc. Prof. Cem Ersoy	.....
Prof. Çetin Kaya Koç	.....
Prof. Oğuz Tosun	.....
Prof. Nadir Yücel	.....

DATE OF APPROVAL .....

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to Dr. M. Ufuk Çağlayan who supervised me during the thesis period. I am also grateful to Dr. M. Ufuk Çağlayan, Dr. Cem Ersoy, Dr. Çetin Kaya Koç, Dr. Oğuz Tosun and Dr. Nadir Yücel for their participation in my thesis jury, careful reviews and comments. I also thank Dr. Cem Ersoy for the discussion on the performance evaluation subjects.

This work has been supported by Boğaziçi University Research Fund under Grant 97A0102, by the Scientific and Technical Research Council of Turkey (TÜBİTAK) under Grant EEEAG-237 and by Turkish State Planning Organization (DPT) under Grant 96K120490. I would like to give my thanks all of these institutions. In addition, I specially thank TÜBİTAK because of the Ph.D. scholarship that I was honored.

I appreciate the efforts of Dr. M. Ufuk Çağlayan and Dr. Cem Ersoy to establish an excellent research environment NETLAB (Computer Networks Research Lab). I would like to express my gratitude to all my past and present NETLAB mates, Dr. Sema Oktuğ, Tuna Tuğcu, Bilgin Kerim, Mete Yılmaz, A. Murat Eren and Emre Çelebi, and all other Computer Engineering Department staff, who support and help me all the time.

I thank Mr. Graham Russell for his proofreading the text. I am also grateful to the editors and the anonymous referees of various conferences, symposia and journals, who reviewed the preliminary versions of the text and presented important comments.

I would also like to thank my family for their support. Last but not the least, special thanks go to my beloved wife Derya for her endless understanding, patience and taking care of me during this hectic period. Moreover, I apologize to our future children whose births are delayed because of this thesis.

ABSTRACT

Digital certificates are employed in existing *classical certification* systems to certify the public keys of the users. In this thesis, a new certification scheme, which is called *nested certification*, is proposed. In simple terms, a nested certificate is defined as a certificate to certify another certificate. The nested certification scheme brings out a new certificate verification method, called *subject certificate verification.* Nested certificates can be used together with classical certificates in the Public Key Infrastructures (PKIs). Such a PKI, which is called *Nested certificate based PKI* (*NPKI*), is proposed in this thesis also. Moreover, it is shown in this thesis that subject certificate verification and the verification of certificate paths in NPKI have the same confidence as the classical cryptographic certificate and certificate path verification methods. Nested certificates give less assurance than the classical certificates and no trust assumptions are necessary to issue them. In this way, the certificate issuers and verifiers of NPKI become more flexible.

In this thesis, analytical and simulation based performance analyses are also carried out, in order to show the nested certification overhead and the efficiency improvement in certificate and certificate path verification. These analyses show that the subject certificate verification method and the usage of nested certificates in NPKI significantly improve the verification times as compared to cryptographic certificate and certificate path verification methods. The disadvantage of nested certification in NPKI is the overhead of a large number of nested certificate issuances, for the cases where nested certification is enforced. However, this overhead is acceptable in order to have quickly verifiable certificate paths.

ÖZET

Sayısal sertifikalar, klasik sertifikasyon sistemlerinde açık anahtarları onaylamak için kullanılmaktadırlar. Bu tezde, *içiçe sertifikasyon* isimli yeni bir sertifikasyon modeli önerilmiştir. Basit bir deyişle, içiçe sertifika, başka bir sertifikayı onaylayan sertifika olarak tanımlanabilir. İçice sertifikasyon modeli, *konu sertifika doğrulama* isimli yeni bir sertifika doğrulama yöntemini de ortaya çıkarmıştır. İçiçe sertifikalar, *Açık Anahtar Altyapıları*'nda (PKI) klasik sertifikalar ile birlikte kullanılabilirler. *İçiçe sertifika tabanlı PKI* (*NPKI*) denen bu şekildeki bir PKI da bu tezde önerilmiştir. Bunlardan başka olarak bu tezde, konu sertifika doğrulama ve NPKI'da sertifika yolu doğrulamanın, klasik şifreleme tabanlı sertifika ve sertifika yolu doğrulama yöntemleri ile aynı güvenceye sahip olduğu gösterilmiştir. İçiçe sertifikalar, klasik sertifikalardan daha az garanti verirler ve onları üretmek için hiçbir güven varsayımında bulunmaya gerek yoktur. Bu şekilde, NPKI'daki sertifika üreticileri ve doğrulayıcıları daha esnek olurlar.

Bu tezde, içiçe sertifikasyon ek yükü ve verim artışını gösterebilmek için analitik ve benzetimsel başarım analizleri de yapılmıştır. Bu analizler, konu sertifika doğrulama yönteminin ve NPKI'da içiçe sertifika kullanımının, şifreleme tabanlı sertifika ve sertifika yolu doğrulama yöntemlerine göre doğrulama zamanını önemli derecede iyileştirdiğini göstermiştir. İçiçe sertifikasyonun mecbur tutulduğu durumlarda, fazla sayıda içiçe sertifika üretiminin getirdiği ek yük, NPKI'daki içiçe sertifikasyonun dezavantajıdır. Ancak, bu ek yük, hızlı bir şekilde doğrulanabilen sertifika yolları elde edebilmek için kabul edilebilir bir ek yüktür.

.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A_i	An authority (CA/NCA) on a certificate path
c	a cryptosystem
CA	A certificate authority
cc	A classical certificate
cc_i	A classical certificate on a path
c_i	A certificate on a path
c_{PKI}	Total number of classical certificates in a public key infrastructure
c_{v_i}	Number of classical certificates issued by v_i
D	Decryption process
E	Encryption process
f	Subject certificate verification speed-up factor
f_{path}	Path verification speed-up factor
H	Hash function
h z	a hash algorithm
k	number of levels in a balanced tree shaped public key infrastructure
KR_x X_s	Private key of user X
KU_x X_p	Public key of user X
m	number of children of each non-leaf node of a balanced trees shaped public key infrastructure
nc	A nested certificate
NCA	A nested certificate authority
n_{cc}^{path}	Total number of classical certificates *path*
nc_i	A nested certificate on a path
$NCPO_{PKI}$	Nested certificate propagation overhead to convert the whole PKI into NPKI
$NCPO_{v_i}$	Nested certificate propagation overhead for v_i
n_{nc}^{path}	Total number of nested certificates of the *path*

n_{PKI}	Total number of nested certificates in a public key infrastructure
n_{total}^{path}	Total number of certificates of the *path*
n_{v_i}	Total number of nested certificates from v_i towards the end users
npl_{NPKI}	Number of nested certificates on the average length nested certificate in a NPKI
pl_{NPKI}	Average nested certificate path length in a NPKI
r	Ratio of the subject certificate verification speed-up factor over the NPKI certificate path speed-up factor
$S(m)$	Bit length of message *m*
sc	A subject certificate
T	Target entity
T_{C-path}	Total *C-path* verification time
t_h	Fixed setup time for the hash algorithm *h*
T_{ncpath}	Total *ncpath* verification time
T_{N-path}	Total *N-path* verification time
T_{pkc}	Total time of cryptographic certificate verification
T_s	Total time of subject certificate verification
V	Verifier
V_i	Set of nodes in the level *i* of a tree
v_i	An element of V_i
X_p[I]	Inverse operation of X_s[I]. Expected to return I′, if I= X_s[I′].
X_s[I]	Signature of the user *X* over the information I. The signature is issued by X_s.
λ_h	Throughput of the hash algorithm *h*

1. INTRODUCTION

Public key cryptosystems [1] are used extensively in network security and authentication applications. 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. The private keys are used to decrypt messages and digitally sign information, therefore, they must be kept secret. On the other hand, the public keys are used to encrypt messages and to verify digital signatures. Since these operations can be carried out by anyone, the public keys can be known by everyone. This characteristic of public key cryptosystems makes the key distribution less difficult compared to conventional cryptosystems. However, there are still some public key distribution problems in public key cryptosystems.

There are two types of problems regarding the public key distribution: (i) the media for the distribution, (ii) correctness of the public keys. Employing some global directories that contain the public keys and related information has solved the first problem. International Telecommunications Union (ITU) certification standard X.509 [2] uses X.500 [3] directories for public key publishing. Pretty Good Privacy (PGP) [4,5] uses public key servers, which contain the public keys along with the identities.

The second problem related to 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 users to other users. The rationale for this requirement is to avoid name spoofing. A public key owner may introduce him/herself with a different name and the other users of the application may consider that the user is in fact the person he/she is claiming to be, however he/she may not be that person.

The general practice to avoid name spoofing is to use some trusted entities to introduce the users to the system. These trusted introducers digitally sign the binding between the public key and the real identity of a user. The application partners verify the

signature of the introducer over that binding using the introducer's public key and make sure about the validity of the public key of the user. Such a system can work if and only if the introducer is a trusted entity for the verifier and the verifier knows the introducer's public key, because otherwise the verifier cannot verify and comment on the signature of the introducer on the identity - public key binding of the user. This digitally signed information, which contains the public key of a person with the identity information and some managerial details, is called the *certificate* and the introducer is called the *Certification Authority* (*CA*).

It is obvious that single CA is not sufficient for a network with large number of users. Therefore, there must be several CAs throughout the global network. Moreover, there must be a certificate network to connect the CAs and other users. Such a network is called a *Public Key Infrastructure* (*PKI*). In that way, the users, who have certificates from different CAs, will be able to verify each other's certificates. In order to find out the correct public key of a target user, the verifier may follow a *certificate path* with several certificates. Each certificate on this path is verified to find out the public key of the next CA and each public key is used to verify the next certificate. The verifier has to know the public key of the first CA and has to trust all the CAs of the path.

The classical certification systems, like X.509 and PGP, use public key cryptography in order to digitally sign and verify the certificates. Public key cryptography operations are time inefficient. Moreover, to verify the certificate of a target user, all of the certificates of a certificate path must be verified one by one. The verifier only wants to find out the correct public key of the target entity, but it has to verify the certificates of all the intermediary CAs of the path and find out their public keys to reach the target entity. This is, actually, an unnecessary process and degrades the time efficiency of the verification.

In a X.509 based PKI, there are two types of users, which are the CAs and the end users. The CAs issue certificates to other CAs and the end users, but the end users cannot issue certificates. Similarly, there are two types of certificates. These are the certificates towards CAs and the certificates towards end users. The X.509 standard allows CAs to

restrict the usage of a certified public key to a specific action. In other words, a limited authorization is possible via X.509 certificates. This would be sufficient for the end user certificates, however, the CAs may want to enforce further restrictions within CA certificates. For example, a CA, say CA_1, wants to issue a certificate to another CA, but CA_1 also requires to enforce that the public key within this certificate should be useful only to verify specific certificates. This type of certificate based restriction cannot be realized using classical X.509 certificates.

One way of improving the certificate path verification time is to verify the public key of the user to be certified via a certificate path and issue a *direct classical certificate* for it. In this way, the verifiers who want to find out the public key of that user can quickly verify the direct classical certificate instead of following a certificate path. There are three shortcomings for the application of such an approach. One of them is the fact that all of the CAs on the certificate path must be trusted in order to verify the path and issue a direct certificate. Even if one of the CAs is not trusted, the path cannot be verified and the direct certificate cannot be issued. The second shortcoming is the strict hierarchical structures of some PKIs. Direct certificate issuance spoils the hierarchical structures of those PKIs. Therefore, such PKIs do not allow direct certificate issuance. The third shortcoming is the increase in the number of certificates.

1.1. Contribution of the Thesis

In this thesis, *nested certification* scheme is proposed as the partial solution to the above shortcomings of the classical certification systems. The certificates issued in this scheme are named as *nested certificates*. A nested certificate is used to certify another, say *subject*, certificate. Therefore, it can be considered as “a certificate for another certificate”. The subject certificate can be a classical or another nested certificate. The basic idea behind nested certification is to delegate the subject certificate signature verification responsibility to the nested certificate issuer. In this way, the signature over the subject certificate can be verified via a nested certificate. Moreover, the verifier need not know or

find out the public key of the subject certificate issuer. This verification method is a consequence of nested certification and named as *subject certificate verification.* In this thesis, it is shown that the subject certificate verification method has the same confidence as the cryptographic certificate verification method. The subject certificate verification method does not use public key cryptography operations, if there is a legitimate nested certificate for the subject certificate. Therefore, it is more efficient than classical public key cryptography based certificate verification.

A nested certificate is issued for an existing subject certificate. Therefore, nested certificate issuer gives assurance for a single action of the subject certificate issuer. In this way, certificate based restriction can be performed. Moreover, trust information is not conveyed within a nested certificate and the nested certificate issuer does not guarantee the correctness of the information within the subject certificate. The nested certificate issuer only guarantees the legitimacy of the signature over the subject certificate and conveys this information to the verifiers. Therefore, in order to issue a nested certificate, the nested certificate issuer need not trust anyone even the subject certificate issuer. In this way, the nested certificates become an alternative to direct classical certificates and they can be issued where the direct certification is not possible because of lack of trust.

Nested certificates are not designed to replace all the functions of the classical certificates. On the contrary, nested certificates are designed to improve the performance and flexibility of the classical certificate usage. Therefore, both classical and nested certificates can be used together in PKIs and certificate paths. In this thesis, a *Nested certificate based PKI* (*NPKI*) is proposed also. Two basic NPKI construction models are proposed in the thesis. Both models allow nested certification as well as classical certification. The first model is called the *free certification* model. This model suggests an organic growth from zero. In this model, every CA is free to choose classical or nested certificates to issue. There is no enforcement. The second model is called the *transition from existing PKI* model. In this model, there is systematic nested certification enforcement. Every CA issues nested certificates to the certificates that are issued by its neighbors in the PKI.

The certificates are verified via certificate paths in NPKI. The certificate paths that are formed by using the free certification model are called as *NPKI certificate paths*. The NPKI certificate paths contain both classical and nested certificates and the last certificate must be a classical one. In a NPKI certificate path, the successor certificates of the nested certificates are verified by using the subject certificate verification method. Other certificates are verified by using the public key cryptography based classical method. The certificate paths that are extracted from a NPKI, which is formed by using the transition from an existing PKI model, are called as *nested certificate paths*. In a nested certificate path, all of the certificates except the last one are nested certificates. Only the first nested certificate of a nested certificate path is verified cryptographically, other certificates are verified using the subject certificate verification method. In this thesis, it is also shown that the verification of a classical certificate via NPKI/nested certificate path verification methods has the same confidence as the cryptographic verification of the same certificate.

Since the subject certificate verification method is more efficient than the cryptographic certificate verification method, the usage of nested certificates on certificate paths is expected to improve the NPKI/nested certificate path verification time as compared to the classical certificate paths of the same length. The only overhead of the nested certification may be the increase in the number of certificates to convert an existing PKI into NPKI.

In this thesis, analytical and simulation based performance evaluations are carried out to show the efficiency improvement in the verification processes and the nested certification overhead in NPKI. These efficiency analyses are performed for the subject certificate verification method and the NPKI/nested certificate path verification methods. These analyses show that the subject certificate verification method performs significantly better than the cryptographic certificate verification method. The speed-up factor is between 8 to 3000, depending on the certificate size, cryptosystems and hash functions used. However, the speed-up factors for the NPKI and nested certificate path verification methods are smaller than the speed-up factor of subject certificate verification. The reason is that one or more cryptographic certificate verifications must be performed in these paths.

The speed-up factor is largely dependent on the number of nested certificates on the path. Other factors that effect the relative improvement are the cryptosystems, hash functions used and the average certificate size of the path. Nevertheless, the improvement is still remarkable. For example, the speed-up factor for nested certificate path verification method ranges between 1.87 and 8.83 for nested certificate paths with 1 to 8 nested certificates.

Moreover, the trade-off between the nested certification overhead and the efficiency improvement in the *transition from existing PKI* model is also analyzed. This analysis has shown that the usage of nested certificates significantly improves the verification time. However, the nested certification overhead is also quite high for the certification authorities, but acceptable to have faster verification.

1.2. Structure of the Thesis

In Section 2, an overview of the network security mechanisms is given. Moreover, the digital certificate concept is detailed there. A literature survey on existing certificate systems, PKIs and nested signatures are also given in this section.

Section 3 deals with nested certificates and nested certificate paths. The structure, issuance and verification methods of nested certificates are explained in this section. The subject certificate verification method is also given there. The nested certificate paths are also introduced in this section and the nested certificate path verification method is described. The characteristics, advantages and disadvantages of nested certificates are also detailed in this section. Moreover, it is proven that the subject certificate verification method has the same confidence as the cryptographic certificate verification method. A similar proof is given for the nested certificate path verification method also.

In Section 4, NPKI is detailed. The general characteristics, structure and certificate path processing in NPKI are given in this section. The transition from an existing PKI to NPKI is detailed also. Moreover, X.509 conformance issues for nested certificates and NPKI are described here.

Performance evaluation of nested certificates is given in Section 5. In this section, both analytical and simulation based performance analyses are given. The efficiency improvement in subject certificate verification, NPKI certificate path verification and the nested certificate path verification methods are analyzed in this section. Moreover, the nested certification overhead is also analyzed and the trade-off between this overhead and efficiency improvement is interpreted.

Section 6 gives the conclusions of the thesis and some possible future works that can be carried out to improve the current status of this study.

2. OVERVIEW OF NETWORK SECURITY AND CERTIFICATION SYSTEMS

Digital certification is closely related to network security requirements and cryptographic security mechanisms. In this section, an overview of these mechanisms is given and the existing digital certification approaches are detailed.

2.1. Introduction to Network Security Systems

The data sent over computer networks are sensitive to attacks by several forms of intruders. This is because of the sensitivity of the applications that run over computer networks, such as banking, accounting, auditing, electronic payment and electronic voting. These applications should run over the network in a robust manner such that no person can perform an unauthorized transaction. Therefore, it is essential to construct a security system for computer networks in order to protect the data during their transmission and discard the effects of intruders. In this subsection, some important potential *threats* and *security mechanisms* against these threats are discussed.

2.1.1. Threats

1. **Data interception:** The observation and/or recording of user data by an unauthorized user during a communication.

2. **Data manipulation:** The alteration, insertion, deletion or misordering of user data by an unauthorized user during a communication.

3. **Masquerade:** Pretence by a user to be a different user in order to gain access to private information or to acquire additional privileges.

4. **Repudiation:** The denial by a user of having participated in part or all of a communication. For example, the denial of sending of an e-mail message.

2.1.2. Services and Mechanisms

The security services and mechanisms that provide solutions to above threats are given below.

1. **Data Confidentiality:** This service provides protection of user data from unauthorized disclosure. Therefore, it is used as the solution to *data interception*. This service can be realized by using *encipherment*. Encipherment is the transformation of an original message, called *plaintext*, into a form such that only the intended recipient can restore the plaintext. Cryptographic operations are used for encipherment, as will be discussed in Section 2.3.1.

2. **Authentication:** This service is the process of peer-to-peer identity verification in a communication. In a networking environment, mostly the receiver wants to verify the identity of the sender. However, there are some cases where both parties want to verify each other's identity mutually. Authentication is the solution for the *masquerade* threat. The realization of this service is possible by using cryptographic operations, as will be discussed in Section 2.3.2.

3. **Non-repudiation:** This service prevents the sender from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the message was in fact sent by the alleged sender. This is the solution for *repudiation* threat. The difference between authentication and non-repudiation is that in non-repudiation, the

receiver must be able to prove the fact to a third party as well as the sender. However, in authentication, assurance between sender and receiver is enough. The non-repudiation service can be realized by using *digital signatures*. The digital signature mechanism will be overviewed in Section 2.3.3.

4. **Data Integrity:** This service provides solution for the *data manipulation* threat by putting some extra data dependent and unforgeable information within the transmitted data. In this way, any malicious modification can be located. The systems that provide authentication, non-repudiation or confidentiality also satisfy the integrity requirement as will be discussed in Section 2.3.

2.2. Cryptographic Operations

The network security solutions are based on cryptography. Cryptography [1] is the science of secret information processing. Moreover, it can be used for authentication and digital signatures. In this section, the cryptographic operations that will be referred often in the forthcoming sections are overviewed.

Cryptographic systems are based on two different groups of algorithms. One of them is called *conventional algorithms*, they are also known as *private key algorithms* and *symmetric algorithms*. The second group of algorithms is called *public key algorithms* or *asymmetric algorithms*. These algorithms are used for confidentiality, authentication and digital signatures as will be overviewed in Section 2.3. *Hash algorithms* are used to create a unique fingerprint of a message. They are mostly involved in integrity control.

2.2.1. Conventional Cryptosystems

In conventional algorithms, the encryption and the decryption keys are the same or each of them can be obtained from each other easily. Therefore, the key used in decryption and encryption processes should be kept as a secret between the communicating parties. Moreover, this key must be distributed between the sender and the receiver via a secure channel before data transmission. The conventional approach is the classical approach in cryptography and its history started in Ancient Rome. In today's world, the most widely used private key algorithms are the Data Encryption Standard (DES) [6] by National Information Standards Institute (NIST), International Data Encryption Algorithm (IDEA) [7,8] by Swiss Federal Institute of Technology, RC5 [9] and RC2 [10] by RSA Labs.

2.2.2. Public Key Cryptosystems

Public key cryptography became popular after the revolutionary paper of Diffie and Hellman [11]. In public key algorithms, the encryption key and the corresponding decryption key are created by the key owner and they are different. Moreover, it is practically impossible to obtain the decryption key from the encryption key. Therefore, the encryption key, which is also known as *public key*, can be made public, but the decryption key, which is also known as *private key*, should be kept secret. The encryption and the decryption processes are just inverses of each other. Generally, conventional algorithms are faster than the public key algorithms. However, public key algorithms are more secure.

The most widely used public key algorithm is by Rivest, Shamir and Adleman and known as the RSA algorithm [12]. The cryptographic strength of this algorithm is based on the difficulty of the factorization of large numbers. The encryption and decryption methods use modular exponentiations in this algorithm. Therefore, it is slow. This algorithm is characterized by the modulus of the key. Larger modulus increases the strength of the method.

2.2.3. One-way Hash Functions

Hash functions are used to calculate hashes (also known as message digests) of messages. A hash function takes a variable length message as input and produces fixed size hash. The hash functions are one-way functions, such that it is infeasible to find out a message given the corresponding hash value. Moreover, it is also infeasible to find two messages with the same hash value. The hash functions are mostly used for integrity control in authentication and digital signature protocols. These functions are also used for integrity control in nested certificates.

A good overview of hash functions is given in [13]. The most widely used hash functions are Message Digest 4 (MD4) [14], Message Digest 5 (MD5) [15] and Secure Hash Algorithm 1 (SHA-1) [16]. MD4 and MD5 algorithms were developed by Rivest. SHA-1 algorithm was developed by NIST. The MD4 and MD5 algorithms produce 128-bit hashes, SHA-1 algorithm produces 160-bit hashes. Therefore, SHA-1 is more secure, but slower than MD4 and MD5.

2.3. The Use of Cryptography in Network Security

The network security services that are described in Section 2.1.2 can be realized using cryptographic operations. In this section, the general characteristics of these realizations are described.

2.3.1. Confidentiality

In conventional cryptosystems, confidentiality is achieved by encrypting plaintext using a shared secret key. In this way, a ciphertext is produced and it is sent to the receiver. The receiver decrypts the ciphertext and restores the original plaintext. Since only the

sender and the receiver know this key, no other party can see the plaintext while in transit. This model is shown in Figure 2.1.

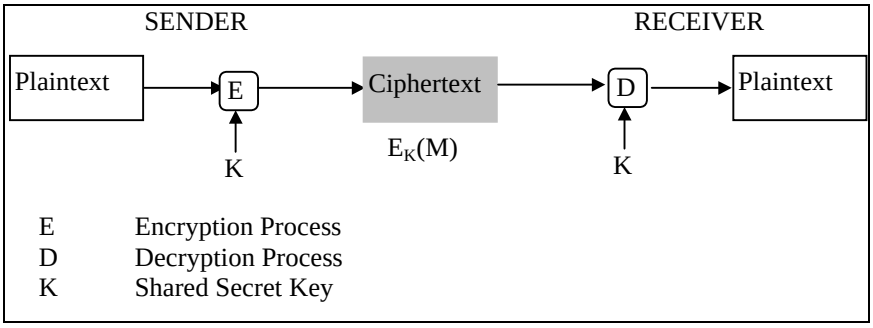

Figure 2.1. Confidentiality by using conventional cryptosystem

In public key cryptosystems, different keys are used for encryption and decryption. The sender uses the public key of the receiver to encrypt the plaintext. The receiver uses its own private key to decrypt the ciphertext and restore the plaintext. Since only the receiver knows the private key, nobody can see the plaintext during transmission. Here, it is assumed that the sender knows the correct public key of the receiver. This model is shown in Figure 2.2.

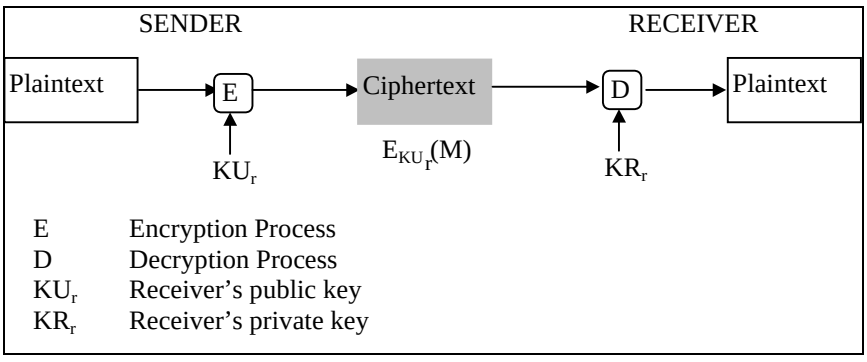

Figure 2.2. Confidentiality by using public key cryptosystem

2.3.2. Authentication

A simple message authentication scheme, which is based on conventional cryptosystem, is shown in Figure 2.3. This scheme also checks for the integrity of the message. In this scheme, the sender computes the hash of the plaintext and encrypts this hash using the shared key. Then, the sender concatenates the plaintext and the encrypted hash and sends them to the receiver. The receiver decrypts the encrypted hash using the shared key and recalculates the hash from the plaintext. Then, the verifier compares the decrypted hash with the recalculated hash. If they are the same, then the receiver concludes that the plaintext is actually sent by the sender. The reason behind this conclusion is the fact that the encrypted hash can only be created by the sender. In this method, the integrity of the plaintext is also checked automatically by the comparison at the end, because if the plaintext was modified during the transmission, then the comparison could not yield equality.

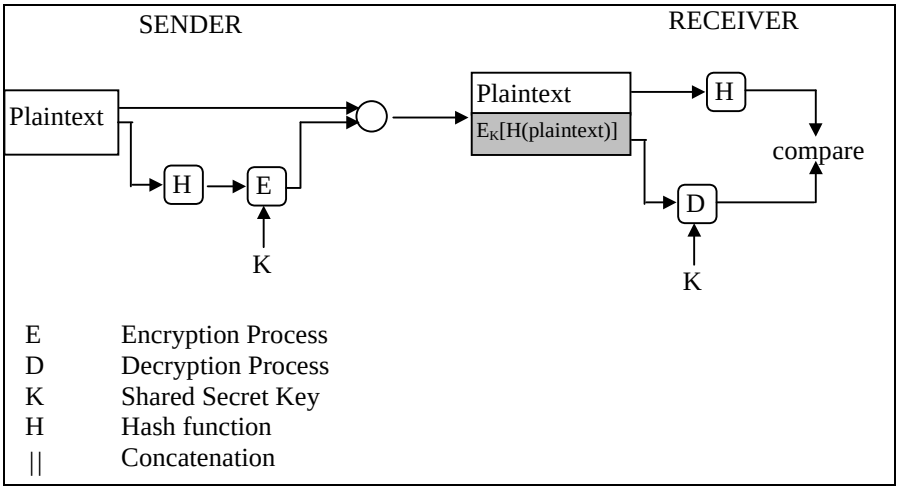

Figure 2.3. Authentication by using conventional cryptosystem

The authentication scheme, which uses a public key cryptosystem, is similar to above scheme with the exception of encryption and decryption keys. This scheme is shown in Figure 2.4. In this scheme, the sender encrypts the hash using its own private key and receiver uses the corresponding public key of the sender for verification. Since the sender's

private key is known only by the sender, the only entity that can create the encrypted hash is itself. The receiver uses this fact to ensure the authenticity of the plaintext. Moreover, the integrity of the plaintext is also assured by this method. This method is used for digital signatures also, as will be described in Section 2.3.3.

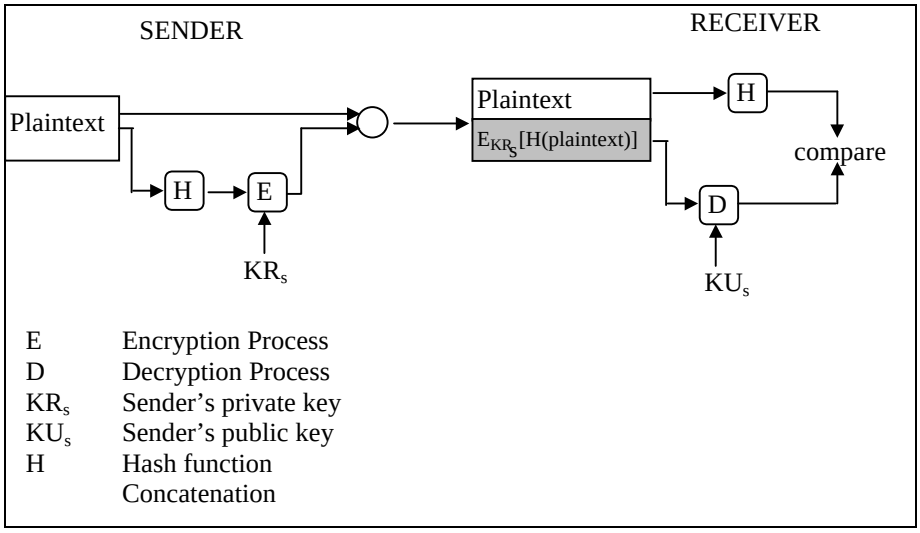

Figure 2.4. Authentication and digital signatures by using public key cryptosystem

The simple authentication schemes described above are just two basic models to achieve one-way authentication, in which only the receiver authenticates the sender. Another issue regarding authentication is the *mutual authentication* in which two parties authenticate each other. Moreover, a complete authentication system must also solve two other basic problems. These problems are the *replay* and the *key distribution* problems. The replay problem is basically the problem of recording a message and resending it at a later time by a malicious intruder. A complete authentication system must take precautions to such replay attacks. Gong has detailed some other issues regarding the replay problem in [17].

The two basic message authentication schemes described above assume that the necessary keys are obtained by the sender and the receiver a priori. That is, the sender and the receiver have agreed on a key, *K*, before the communication in the former scheme. Similarly, it is assumed that the receiver has obtained the public key of the sender in the

latter scheme. Similar problems are valid for the confidentiality models that have been given in Section 2.3.1. This problem is the *key distribution problem*. A complete authentication system should also solve the key distribution problem.

The general tendency is to link authentication and key distribution, since key distribution is the prerequisite of authentication. Indeed, there are several mutual authentication protocols that contain solutions to the replay and key distribution problems as in [18,19,20,21,22,23,24,25,26,27,28,29,2]. Key distribution is a side product of most of these protocols. Moreover, there are some other mechanisms for only secure key distribution. Diffie-Hellman key exchange mechanism [11], which uses public key cryptography, is the first and most popular such mechanism. A comprehensive discussion on key distribution and management is given by Fumy and Landrock in [30].

Some of the authentication protocols, which distribute conventional keys as session keys, are given in [18,19,20,21,22,23,24,25]. They all use mutually trusted third parties for authentication and key distribution. In these protocols, the registration of the entities at those trusted authorities before the authentication steps is a must, since each entity must have a key for the secret communication between the authority and itself during the protocol. These keys are established or exchanged at the registration time.

Some other authentication and key distribution protocols are based on distribution of public keys [26,27,28,29,2]. These protocols also use trusted third parties and require prior registration. The public key distribution problem will be discussed in Section 2.3.4 in more detail.

2.3.3. Digital Signatures

Digital signature is the mechanism for the non-repudiation requirement. A digital signature is a piece of data obtained from the message sent. This data must be able to be

created only by the actual sender. The receiver or any other third party must not be able to forge the digital signature. This requirement makes the authentication scheme with conventional cryptosystems, which is explained by Figure 2.3 in Section 2.3.2, useless to obtain digital signatures. In order to use conventional cryptosystems for digital signatures, on-line trusted third parties must be involved in the protocols. Therefore, the use of conventional cryptosystems for digital signatures is not practical. On the other hand, it is possible to obtain digital signatures without having trusted third parties by employing public key cryptography. An overview of digital signature methods is given in [31].

The authentication model given in Figure 2.4 also satisfies the digital signature requirements, since the encrypted hash can be created only by the sender. The hash is encrypted using the private key of the sender and this key is known only by the sender. The encrypted hash serves as a *digital signature* over the plaintext. The digital signatures obtained in this manner are plaintext dependent. A new hash calculation and encryption is necessary for each message signed. In the verification phase, the receiver uses the public key of the sender in order to decrypt hash. The receiver recomputes the hash from the plaintext once more. If the outcomes of these two steps match, then the receiver concludes that the signature over the plaintext is legitimate. This verification can actually be performed by anyone who knows the correct public key the sender. That means, the digital signatures are not specific to receivers.

The public key cryptosystems used for digital signatures are the RSA [12] and the Digital Signature Algorithm (DSA) cryptosystems. The DSA cryptosystem is described in Digital Signature Standard (DSS) [32] by NIST. This algorithm is a variant of El Gamal algorithm [33]. The security of DSA is directly related to the size of the key. The size of the signatures produced by this algorithm is not dependent on the key and the message size. The signatures have two 160-bit parts. On the other hand, the size of the signatures produced using RSA algorithm is the same as the key modulus size.

2.3.4. Key Distribution in Public Key Cryptosystems and Certificates

In the digital signature scheme described above, it is assumed that the receiver has obtained the public key of the sender beforehand. Similarly, the sender must know the public key of the receiver to achieve confidentiality using public key cryptosystem. At the first glance, the distribution of the public keys does not seem to be as difficult as conventional key distribution, since the public keys are not secret and can be transmitted over the insecure networks. However, there is still public key distribution problem here. The solution of Diffie and Hellman [11] to this problem is to use commonly accessible "Public Files" in which each entity puts its public key. Diffie and Hellman also stated that such files must be protected from unauthorized modification. Indeed, nowadays it is possible to download the public key of an entity from a publicly accessible bulletin boards and directories via WWW, FTP or another data transfer protocol. Alternatively, the public key owner may send its public key via electronic mail. However, these public key sources may be vulnerable to tampering and/or modification in the transmission time.

Suppose an entity *A* wants to obtain the correct public key of *B*. If *B* hands in its public key to *A* personally by verifying its identity, then *B* can make sure that this public key really belongs to *A*. However, *A* cannot make sure about the correctness of the public key of *B*, if *A* gets it over the network. The claimed public key may actually belong to another entity *C*, since there is no guaranteed binding between public key and the identity of its owner within a public key. In this way, *C* can masquerade as *B*. This problem is called as *name spoofing* problem. This is another important problem that was not discussed by Diffie and Hellman in [11]. The possible harms of name spoofing are the followings.

1. Suppose *C* has intercepted the encrypted messages from *A* to *B*. *A* has encrypted these messages using the fake public key of *B*. However, this public key actually belongs to *C*. Moreover, *C* knows the corresponding private key. Therefore, *C* is now capable of decrypting these messages.

2. Suppose *B* has digitally signed a message using its private key and sent it to *A*. Since *B*'s public key that *A* knows is not the correct one, *A* cannot verify the signature over the message, even if it is legitimate.

There is another problem in public key distribution, even if *A* gets the public key of *B* personally. The aim of the digital signatures is to prevent non-repudiation. However, non-repudiation may not be satisfied always using the digital signatures in above manner. As an example case consider the following scenario. Suppose, *A* gets the public key of *B* personally and makes sure about its correctness. Later *B* signs a message and sends it to *A*. *A* verifies the signature using the public key of *B*. Then, *B* repudiates sending such a signed message and argues that the public key that *A* used does not belong to itself. Now, *A* must be able to prove to a third party that it used the correct public key of *B* for verification. However, *A* cannot prove this fact, since it has no evidence to show to a third party.

In order to solve these problems, a trusted third party must be involved in the system. Popek and Kline [26] proposed a method in which a trusted third party distributes the public keys of *A* and *B* to *B* and *A* respectively. In this method, it is assumed that the trusted third party holds the correct public keys of *A* and *B*. Moreover, everyone knows the correct public key of the trusted third party. In order to distribute the public key of *A* to *B*, the trusted third party digitally signs both the identity and the public key of *A* and sends this signed information to *B*. The same method is applied to distribute the public key of *B* to *A*. The entities make sure about the correctness of the public keys of each other, since they are able to verify the signature of the trusted third party. Moreover, an entity, say *A*, cannot later repudiate its correct public key, since the signature of the trusted third party over the public key of *A* is the evidence for *B* to prove that it has the correct public key.

The overhead of above method is the on-line availability necessity of the trusted third party. Therefore, it has not been accepted widely and has not been implemented. An alternative approach is the *certification* mechanism. As discussed in [34,35,36], the certification mechanism is first suggested by Kohnfelder in 1978. A *certificate* is a digitally signed binding between the identity and the public key of an entity. *Certificate*

content contains the public key, the identity of the certificate owner and some managerial information. This content is digitally signed by a trusted *Certification Authority* (*CA*) and this signature is appended to the certificate content in order to issue a certificate. The digital signature method described in Section 2.3.3 is used.

The certificates are created on demand. An entity *B*, who wants to have a certificate, first creates a private-public key pair. *B* keeps the private key and takes the public key to the CA. The CA first validates the identity of *B* via some printed documents, like an identity card or driving licence. Then, the CA issues the certificate for *B*. This is an off-line process such that the CA need not to be available during the further communication between *B* and another entity *A*. The certificates are created once and they can be used to find out the public keys several times. The certificates can be obtained directly from the certificate owner or from publicly accessible bulletin boards. It is important to point out that the certificate distribution need not be secure, but has to be authentic. If another entity *A* wants to find out the correct public key of *B*, then first *A* obtains the certificate of *B*. Next, *A* tries to verify the signature, which is issued by the CA, over the certificate of *B*. *A* has to know the public key of the CA in order to verify this signature. If the verification is successful and *A* trusts CA, then *A* finds out the correct public key of *B*: *A* should trust the CA considering that the CA is an honest entity and does not issue certificates to entities who are unknown to CA. Therefore, CAs are considered as the trusted third parties in the certification systems.

For a small environment a single CA may suffice. However, for larger networks, like the Internet, there must be numerous CAs. Moreover, there must be a network of certificates to connect the CAs and consequently, the users certified by the CAs. In this way, the users, who have certificates from different CAs, can verify each other's certificates. The certificate network of CAs is called the *Public Key Infrastructure* (*PKI*).

In order to verify a certificate and to find out the public key of the target entity *B*, the verifier *A* must know the correct public key of the CA for the certificate of *B*. If *A* does not know it, then it has to verify the certificate for that CA. To do so, *A* has to know the public

key of the CA of the CA of *B*. This loop goes on until *A* is faced with a certificate that it can directly verify (i.e., it knows the public key of the corresponding CA). The certificates in that loop constitute a path, which is called *certificate path*. This path is a directed one and the starting point is a CA of which *A* knows the public key and the ending point is *B*. In this path, each certificate is verified to find out the public key of the next CA and each public key is used to verify the next certificate. The certificates of such a path are drawn from the PKI.

The certificate concept introduced here will be so called *classical certificate* and the certification authorities will be called as *classical certification authorities*. The reason for this renaming is to differentiate between the classical certificates and the nested certificates, which is the contribution of this thesis.

2.4. Classical Certificate Systems and PKIs

After the initial proposal of Kohnfelder in 1978, Denning [27] added the currency requirement for certificates in 1983. Since the certificates bring an extra overhead for public key management, the concepts of public-key cryptography were not deployed until the end of 1980's. Beginning with the development of the Internet, the demand for the data confidentiality and authenticity has increased significantly. The first action in the deployment of public key cryptography, for the use of confidentiality and digital signatures by using the classical certificates, has been taken by the International Telecommunications Union (ITU, formerly CCITT) and International Standards Organization (ISO). These organizations proposed X.509[1] standard [37] as a certificate based public key authentication framework. It was first announced in 1988. Later, some security defects in

[1] This is actually the ITU name, the ISO equivalent of X.509 is 9594-8. However since X.509 is the widely accepted name, this name will be used throughout the thesis.

X.509 were reported in [38,39]. A revised version of X.509 [40] was issued in 1993. The third version of X.509, X.509v3 [2], was published in 1997. The basic difference in X.509v3 is the optional extensions in the certificates. These extensions allow achievement of both authorization and authentication via certificates. Moreover, more flexible name, trust and certificate path processing have come with X.509v3. X.509 standards are also supported by RSA Inc. and they are included in the Public Key Cryptography Standards (PKCS) [41,42,43,44] of RSA Inc.

Another important phenomenon in public key distribution is the Pretty Good Privacy (PGP) software by Zimmermann [4,5]. Although it is used for e-mail security and authentication, the embedded certificate based public key distribution and the trust manipulation make it an important milestone in classical certification.

The X.509 and PGP systems are actually two extreme cases in the certificate world in terms of trust manipulation. PGP supports personal liberation, however, X.509 structures are more centralized and hierarchical. Trust manipulation, certificate and PKI structures are discussed for both X.509 and PGP cases in the next two subsections. Some other certificate and PKI approaches are given in Section 2.4.3.

2.4.1. X.509

X.509 [2] is the ITU standard for the public key based authentication framework. Classical certification is an inevitable piece of this framework. The certificate structure in X.509 is determined by the standard. A X.509 certificate is basically the guarantee of the binding between the identity and the public key of an entity. Both real and network identities are included in the certificates. The network identity of an entity may be the E-mail address, URL address or both of them. The certificates are issued by commonly known and trusted Certification Authorities (CAs). By issuing a certificate, the CA assures that the public key within the certificate belongs to the claimed entity. A X.509 certificate basically contains the following information:

- Version of the certificate
- Serial number for the certificate
- Certificate Issuer
- Certificate validity period
- The name of the certified entity
- The public key certified
- Optional extensions
- Algorithm identifiers used for signing
- Signature

X.509 is part of the X.500 [3] series of recommendations that define a directory service. The directory is a distributed set of servers that maintains a database of information about users. In X.509, the directory serves as a repository of classical certificates. However, there is a high cost of complexity associated with use of X.500 directory technologies and its Directory Access Protocol (DAP). Therefore, a relatively low cost protocol is designed mostly for client applications. This protocol is the Lightweight Directory Access Protocol (LDAP) version 3, which is published by Wahl, Howes and Kille as an RFC [45].

The X.509 standard defines general concepts. In order to have a functional system in compliance to X.509, a detailed design must be produced for this system. This functional system may be either for specific or for general use. Privacy Enhanced Mail (PEM) [46] is the first functional X.509 based system. It is aimed to construct confidential and authentic e-mail transfer between the users. Secure Electronic Transaction (SET) [47] is another X.509 based system. SET is a joint effort of Visa and MasterCard to construct an infrastructure for secure credit card based payment over insecure Internet. The United States Postal Service (USPS) initiated the Information-Based Indicia Program (IBIP) [48] to support new public key cryptographic methods of applying postage over the Internet. The proposed infrastructure for IBIP is a three level X.509 based hierarchical PKI. Public Key Infrastructure for X.509 certificates (PKIX) [49,50] is a general certificate infrastructure for the Internet.

Secure/Multipurpose Internet Mail Extensions (S/MIME) [51] is a secure Internet mail system. The certification infrastructure of S/MIME is based on PKIX infrastructure. Chokhani [52] proposed a X.509 based national PKI for U.S. Levi and Çağlayan [53] proposed a X.509 based multi-purpose PKI for Turkey.

X.509 standard does not enforce any topology for the PKIs that use X.509 certificates. Although X.509 standard does not specify and enforce any topology for a standard PKI, the general characteristics of the X.509 based PKIs are to be hierarchical and centralized. The general characteristics of a X.509 based PKI are: (i) strict distinction between a CA and the end user, that is end users cannot issue certificates (ii) a tree hierarchy with 4-7 levels, (iii) forming optional CA networks via cross certificates (not applicable for PEM). A typical X.509 PKI is given in Figure 2.5. The nodes in that figure represent the CAs and the end users, the arcs represent the certificates from the CAs to the certified entities.

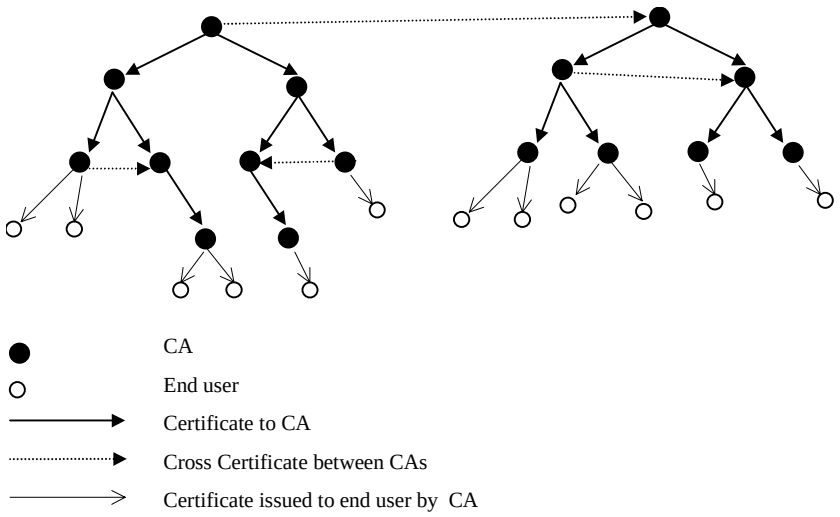

Figure 2.5. A typical X.509 based PKI

The general certificate and certificate path verification methods, which have been introduced in Section 2.3.4 and will be detailed in Section 2.4.5, apply for X.509 based systems.

In the X.509 based systems, every CA is a potentially trusted entity, but there are some mechanisms to avoid “blind trust” to CAs. In the first and second versions of X.509 [37,40] the users had almost no initiative about their trust and certification policies. The users had to trust all of the CAs. However, In the third version of X.509 [2], optional *policy identifiers* were added to the certificate structure as an optional extension. They can be used for user initiated trust management. The *policy identifier* is a value assigned by the CA of the certificate and implicitly explains how the CA has verified the identity of the certified entity and how reliable the CA is for certificate issuance. Several CAs can use the same policy identifiers in their certificates, if their certification policies are the same. On the other hand, a particular CA can use several policy identifiers, if that CA issues different types of certificates based on different policies. The verifier of a X.509 certificate initially determines a set of policy identifiers, *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. In this way, the verifier specifies the class of trusted CAs, and somehow determines its trust policy. Since the trust concept is transitive in X.509, the verifier is also bounded by the allowed policy identifiers for the trusted CAs on the certificate path. However, the policy identifier constraints of the intermediate CAs cannot loose the initial constraints of the verifier. Such a trust mechanism cannot be considered as “complete freedom” about choosing CAs to trust, because the verifier is not able to choose the CAs to trust personally. The advantage of such a scheme is the fact that the verifier does not need to know the identities of the trusted CAs. It has to know only the trustworthy policy identifiers. Moreover, 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).

2.4.2. PGP

The hierarchical and regulative behavior of X.509 certificates bothered people who want to decide on their own trust policies. Zimmermann, who is one of those people,

developed Pretty Good Privacy (PGP) [4,5]. PGP is actually a public key cryptography based e-mail security software. However, it has a typical certification and public key distribution mechanism. Moreover, it is being used by a large community. Therefore, PGP is remarkable.

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. Every user can issue a certificate to another user. The PKI of PGP is a directed graph, in which every node is a PGP key owner and every arc is a certification relationship between two users. The source of an arc is the CA and the target is the certified user. A typical PGP PKI is given in Figure 2.6.

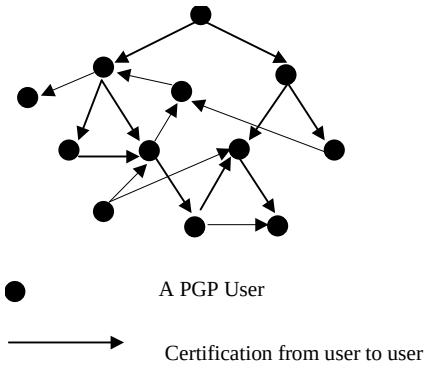

Figure 2.6. A Typical PKI of PGP

The trust model in PGP takes care of personal freedom more than the X.509 based PKIs. In PGP, everyone is free to choose its own trusted CAs. The basic consideration here is the fact that the only ultimately trusted entity for a PGP user is the user itself. Moreover, each PGP certificate path starts with the verifier. However, a PGP user may assign a level of trust to another PGP user to introduce other PGP users to it, but the trust concept in PGP is not transitive. That means, a PGP user can make sure about the validity of the certificates issued by the trusted CAs, but need not trust those trusted CAs to introduce other CAs. The reason behind this situation is not a design or implementation bug, but to give enough freedom to a PGP user to choose its own trusted parties. The lack of trust

transitivity does not mean that the length of a certificate path is at most two. The verifier may find a longer certificate path from itself to the target entity. In this path, each entity certifies the next. However, in order to verify the target entity, the verifier must assign enough trust to all of the intermediate CAs beforehand. Although the certificates are public, the trust information is kept in the local databases of the PGP users, since that information is considered as private information. However, in X.509, the trust information is somehow written in the certificates. The advantage of the trust scheme of PGP is to provide to users more freedom to constitute their trust policies independent of any authority. On the other hand, this characteristic of PGP can be considered as a disadvantage, since the users need to determine the identities of the trusted CAs by themselves.

In PGP, each user assigns some levels of trust to the CAs. PGP supports 4 trust levels: *complete*, *marginal*, *untrusted* and *unknown*. Moreover, multiple CAs may sign a certificate content, but each CA 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 *N* legitimate certificates issued by *N complete*ly trusted CAs, or *M* legitimate certificates issued by *M marginal*ly trusted CAs. The default values for *N* and *M* are one and two respectively, but they can be adjusted by the verifier. The certificates issued by *untrusted* and *unknown* CAs are rejected.

Each PGP certificate contains the following information:

- Public key serial number
- Date of issuance
- The certified public key
- Identity of the public key owner
- Signatures over the certificate content

Moreover, each user holds the following information about the CAs and the certificates in its database:

- Trust level for the public key owner
- Correctness level for the certificate
- Trust levels for the signers (CAs) of the certificate

Each PGP user holds a database for the public keys together with the certificates on them. This database is called as *public key ring*. In a public key ring, the public keys and certificates of the users, to/from whom the ring owner regularly sends/receives messages, are stored. In PGP, *public key servers* [54] are used in order to find out the public keys and the certificates of the users, if they are unknown to the public key ring owner. The public key servers are distributed worldwide and each holds the same key database. When a user wants to obtain the public key of another user or to submit a public key for distribution, it applies to one of the key servers via Internet and stores or gets public keys. The key servers communicate among themselves for public key database synchronization.

The process for the verification of a signature over a certificate is similar to X.509 except the trust manipulation as explained above. Certificate path processing in PGP is also similar to X.509. All of the certificates of the path must be verified one by one sequentially. However, since it may be necessary to have more than one certificate from different CAs because of the trust requirements, more than one certificate can be verified at one step in the path. Moreover, the PGP certificate paths must start with the verifier. Furthermore, each of the CAs on the path must have a level of trust assigned by the verifier. Moreover, the verifiers can restrict the length of the PGP certificate paths by specifying the maximum path length. The total number of certificates of the path cannot exceed this maximum value. Since the PKI of PGP has a graph structure, there can be multiple paths between any two users. These multiple paths help to improve the reliability of the verification mechanism and the availability of the public keys. The multiple paths in PKI of PGP have been analyzed by Reiter and Stubblebine in [55]. They also developed a tool, called PathServer, for PGP path analysis.

2.4.3. Other Certification Systems and PKIs

The ICE-TEL project [34] is a pan-European project to build a X.509 based PKI throughout Europe. The importance of ICE-TEL project is that it employs the optional extensions of X.509v3 to have more flexible and user-oriented trust model. In ICE-TEL, each user can specify the authorities that it trusts and can start the certificate path from these trusted authorities.

Distributed Authentication Security Service (DASS) [28] is an authentication system that uses classical X.509 certificates for public key distribution. DASS also allows threaded links between the users and non-parent CAs, that is, a user is free to obtain certificates from CAs other than its parent CAs. Although DASS is based on X.509, it does not use X.500 directories, since the inventors of DASS believe that X.500 directories will not be deployed. In DASS, certificates are distributed via Certificate Distribution Center (CDC). CDC does not only store certificates, it also stores the encrypted private keys to which the owners reach with a password. The DASS architecture is implemented by Digital Equipment Corporation [29].

The Certificate Management System (CMS) [56] is a X.509 based global network system whose primary services are generation, distribution, storage and verification of certificates. The PKI introduced in CMS is hierarchical. CMS has a built-in certificate storage and distribution mechanism like DASS. Although these are the duties of X.500 directories, the unavailability of such directories forced the CMS engineers to provide such a built-in mechanism. In CMS, the end user certificates are stored by User Certification Authorities (UCA). The certificate retrieval, as all other functions of CMS, is done by a protocol between the UCA and the requesting entity.

An important point that is ignored in all of the certification schemes is dispute resolution. In case of a dispute or repudiation, one must decide on who is telling the truth. Moreover, the trusted third parties, that is CAs, may intentionally or unintentionally

misbehave. That means, the ultimate trust to CAs is not so practical. Crispo and Lomas have addressed this problem for the certification systems and proposed an audit trail solution in [57]. In this solution, each critical activity is logged in safe audit trails. Moreover, each log entry is digitally signed by the participants of the activity. In case of any dispute, these logs help a third party to find out the culprit.

A relatively modern certification approach is not to use the certificates only as the binding between the identity and the public key, but also for authorization, that is granting permission to the certificate owner to accomplish a task. This approach introduces authorization certificates as a new concept. For example, a certificate to write electronic checks, a certificate to drive car, a certificate to prove the age and marital status are such authorization certificates. Ellison has given an interesting list of possible certificate usages in [58]. ICE-TEL [34] provides authorization by using the optional extensions of X.509v3. Some other works related to authorization certificates are also in progress. Blaze, Feigenbaum and Lacy from AT&T proposed a tool called *Policymaker* [59] in which the authorization information is kept in a certificate. The Simple Public Key Infrastructure (SPKI) [60,61] was designed by an Internet Engineering Task Force (IETF) working group led by Ellison. The aim of SPKI is to develop certificate format and associated protocols that are simple to understand, implement and use. The SPKI certificates also contain authorization information and bind the keys to that information. Rivest and Lampson proposed Simple Distributed Security Infrastructure (SDSI) [62]. SDSI combines a simple public key infrastructure design with a means of defining groups and issuing group membership certificates. The certificate structure of SDSI is also simple to understand and implement. SDSI's groups provide simple, clear terminology for defining access control lists and security policies. The designs of SDSI and SPKI have merged in SDSI 2.0 version.

Another IETF working group works on the security extensions of the Domain Name System (DNS). In simple terms, basic DNS [63] is a mechanism to find out the IP address and some other information given a DNS host name. DNS security extensions is developed by Eastlake and Kaufman and published as an IETF Request for Comments (RFC) [64].

The extensions provide authentication of the DNS information. Moreover, the security extensions also provide storing and distributing the signed public keys, i.e. certificates. This mechanism is highly similar to X.509 except two main differences. (i) DNS uses its own naming mechanism, whereas X.509 uses Distinguished Names (DN), (ii) in DNS case, signed public keys are stored in and distributed from the distributed DNS databases, whereas the distributed directories are used in X.509 for the same purpose. The public keys stored in DNS may belong to zones, hosts or individual users, but only the zones can sign public keys. That is, zones are CAs here. The signed public keys in DNS security extensions can be used either by authentic DNS information transmission or by any other security system that needs authentic public key distribution. The security extensions are built over the existing hierarchical structure of the basic DNS. Therefore, the PKI of the DNS security extensions is also hierarchical.

2.4.4. Some Applications of Certificates

In this subsection, some implemented and/or standardized applications that use classical certificates are briefly introduced.

The Secure Socket Layer (SSL) [65] is a session layer and client-server type of mechanism, which is used for confidentiality, integrity and authentication. It is developed by Netscape Corp. and it is a widely used security mechanism for all type of application layer protocols, like HTTP and NNTP, that use TCP/IP. SSL provides confidentiality and integrity during the session and authentication before the session starts. SSL uses public key cryptosystem and certificates for public key distribution, and consequently, for authentication and session key distribution. Session keys are conventional cryptosystem keys and they are used to encrypt the data sent between the client and the server during the session. Authentication of the server to the client is a must in the protocol. Therefore, each server has to have a certificate. However, authentication of the clients is optional, since client authentication requires the clients to obtain certificates from trusted certificate authorities and this case would have a negative effect on the deployment of the SSL system. SSL system uses X.509 certificates, but has no predefined PKI topology. The

public keys of some trusted CAs are embedded in the in most of the Internet browsers. The client may also add some other trusted public keys there manually. Moreover, those browsers are also capable of running the SSL protocol. Therefore, the browser of the client can verify any certificate path starting with a CA whose public key is known by the browser.

Another World Wide Web (WWW) security protocol is the Secure HyperText Transfer Protocol (S-HTTP), which is being standardized by IETF [66]. This protocol is an application layer protocol and provides confidentiality, integrity and authentication for HTTP, as SSL does. The algorithmic characteristics and the usage of certificates in S-HTTP are similar to SSL. The only difference is that S-HTTP is only designed for HTTP as an application layer protocol, whereas SSL is a multipurpose session layer protocol.

The IETF has developed two Internet Protocol (IP) layer cryptographic security mechanisms. One of them is the Authentication Header (AH) [67], which is used for integrity control and authentication. Second mechanism is the Encapsulated Security Payload (ESP) [68], which is used for integrity control, authentication and confidentiality. However, these mechanisms do not have embedded key exchange protocols. Therefore, IETF have developed some authenticated key exchange protocols to be used with AH and ESP. One of these protocols is the Oakley [69], which is a key exchange protocol based on Diffie-Hellman technique [11]. The certification mechanism and infrastructure of the DNS security extensions [64] are used in Oakley. IETF has also developed a complementary protocol that serves as a general framework for the secure manipulation of other security association attributes of the key created using Oakley. The name of this protocol is Internet Security Association and Key Management Protocol (ISAKMP) [70]. These two protocols, namely Oakley and ISAKMP, have merged into Internet Key Exchange (IKE) protocol [71]. Although IP Security working group in IETF encourages Oakley and DNS style certification system, Photuris [72], which is a X.509 and PKIX [49] based key distribution mechanism, has also been proposed by Simpson to serve as the key distribution mechanism for AH and ESP.

Another important application of the certificates and PKIs is the electronic payment mechanisms. SET [47] is one of these methods, which is based on credit card payment. There are some other credit or currency based systems, which are reviewed in [73,74].

2.4.5. Classical Certificate and Classical Certificate Path Verification Algorithms

Several approaches to certificate and PKI structures are examined above. In this thesis, the X.509 approach will be followed for classical certificates. The nested certification mechanism, which is the contribution of the thesis, will be built upon X.509. Therefore, in the rest of the thesis, the terms *classical certificate* and *X.509 certificate* will be used interchangeably. Similarly, *classical certificate path* and *X.509 certificate path* will also be used synonymously. Classical certificate path structure, certificate verification and certificate path verification methods were briefly explained in Sections 2.4 and 2.4.1. In this subsection, the certificate and certificate path verification methods will be given formally. The verification methods that are explained here are quantified as "cryptographic" to differentiate between the verification methods due to nested certification.

The basic rule of the cryptographic certificate verification is the existence of a legitimate digital signature, which has been issued by a trusted CA, over the certificate content. The trust to the CA is determined by employing policy identifiers in X.509 as explained in Section 2.4.1. The digital signatures are issued and verified by employing a public key cryptosystem. First, the CA calculates the hash of the certificate content using a one way hash algorithm. Then, the CA signs that hash to issue a certificate. In order to verify a classical certificate, *cert*, using cryptographic certificate verification mechanism, the verifier must know the correct public key of the issuer of *cert*. Assuming that this public key is known by the verifier, the verifier follows the following steps to cryptographically verify *cert*.

1. The verifier first applies the one way hash algorithm to the content of *cert*.

2. The verifier applies the public key cryptosystem based signature verification procedure to the signature part of *cert* using the public key of the issuer of *cert*.
3. The verifier compares the outcomes of above steps. If they are the same, then the verifier makes sure about:
   (i) the integrity of the content of *cert* and
   (ii) the legitimacy of the signature of the issuer of *cert* over the content of *cert*.
4. If the issuer of *cert* is trusted for the verifier, then the above two results of the verification implies that:
   (iii) the information given in *cert* is correct. In other words, the public key of the entity specified in *cert* is legitimate.

Classical certificate path is a chain of classical certificates. A generic classical certificate path is shown in Figure 2.7. The classical certificate paths are formed to verify the correctness of the public key of a target entity *T*. The verifier *V* verifies all of the certificates one by one sequentially. The verification starts with the first certificate of the path and *V* must know the correct public key of the first CA. However, *V* may not be the first CA, it may be any user. On the other hand, the trust of *V* to all of the CAs on the path is essential in order to verify the path. The cryptographic certificate verification steps are applied for the verification of all of the certificates on the path. Each certificate verification yields a public key, which is to be used to verify the next certificate of the path. This loop goes on until the target entity is reached. In order to verify the public key of the target entity, all of the certificate signatures must be legitimate and all of the CAs must be trusted by the verifier.

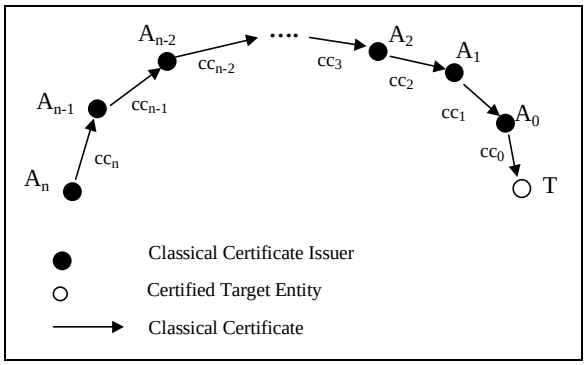

Figure 2.7. A generic certificate path

The cryptographic certificate and certificate path verification methods will be explained formally. The notation used for the classical certificate representation is given in Table 2.1.

Table 2.1. The notation used for classical certificate representation

Notation	**Meaning**
X_p	The public key of X
X_s	The secret (private) key of X
$X_s[I]$	The signature of X over the information I. The signature is issued by X_s.
$X_p[I]$	Inverse operation of $X_s[I]$. Expected to return I', if $I= X_s[I']$.
H[I]	Hash (message digest) of information I.
$I_1 \vert I_2$	Concatenation of the information I_1 and I_2.
$CC_{CA}(Y)$	The classical certificate, which is issued by *CA*, for the user *Y*.
Cnt_{cert}	Content of certificate cert.
Sig_{cert}	Signature over Cnt_{cert}.

Using the above notation, the classical certificate for the user *Y*, which is issued by *CA*, is denoted as follows.

$$CC_{CA}(Y) = Cnt_{CC} \vert Sig_{CC} = Cnt_{CC} \vert CA_s[H[Cnt_{CC}]] \quad (2.1)$$

where Cnt_{CC} contains the identity, public key of *Y* and other classical certificate fields. Cryptographic classical certificate verification is briefly outlined in Figure 2.8. The classical certificate path verification method is given in Figure 2.9. In these methods, it is assumed that the verifier trusts the CAs.

Given: a classical certificate, $CC_{CA}(Y) = Cnt_{CC}|CA_s[H[Cnt_{CC}]]$ issued by a trusted *CA*, and the correct public key *CA*, CA_p.

The verifier applies the following algorithm to verify CC.

$Verified\_Hash \leftarrow CA_p[CA_s[H[Cnt_{CC}]]]$

$Calculated\_Hash \leftarrow H[Cnt_{CC}]$

IF *Calculated_Hash* = *Verified_Hash* THEN

CC becomes verified

ELSE

CC has not been verified

Figure 2.8. Cryptographic classical certificate verification algorithm

Given: a classical certificate path, $cc_n,cc_{n-1},\ldots,cc_2,cc_1,cc_0$, with n+1 certificates issued by trusted authorities $A_n,A_{n-1},\ldots..,A_2,A_1,A_0$, respectively and the legitimate public key of A_n, the issuer of cc_n. The certificate cc_0 is the certificate for the target entity *T*.

The verifier applies the following algorithm to verify the classical certificate path and find out the public key of *T*.

```
success := TRUE
FOR i:= n TO 0 DO
        cc_i is verified cryptographically using the public key of A_i
                IF cc_i is not valid THEN
                        success := FALSE
                ELSE
                        The public key of A_{i-1} is found out
IF success THEN
        certificate path is valid and the public key of T has found out
ELSE
        certificate path is invalid
```

Figure 2.9. Classical certificate path verification algorithm

2.5. Certificate Revocation

X.509 [2] based classical certificates have validity periods. A certificate is valid only within this period. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Such circumstances might include change of name, change of association between the certificate owner and CA and compromise or suspected compromise of the corresponding private key. Under such circumstances, the CA or the certificate owner needs to revoke the certificate.

The method of X.509 for the solution to the certificate revocation problem is to employ *Certificate Revocation List* (*CRL*). A CRL is a signed list of unexpired but revoked certificates. In a CRL, serial numbers and the revocation dates of the revoked certificates are listed. This list is signed by the corresponding CA. CRLs are issued periodically by the CAs. Each CRL invalidates the previous one. The certificate entries in a CRL are removed when the expiration dates of the revoked certificates are reached. A certificate, for which the validity period is valid, should not appear in the most recent CRL of the issued CA in order to be verified successfully. X.509 based PKIs and certificate management systems, such as PEM [46], ICE-TEL [34], CMS [56] and PKIX [49,50], use the CRL approach of X.509.

A slightly different approach to CRL is proposed by Micali [75]. In this approach, Micali proposed *Minimal CRL* (*MCRL*). The aim of MCRL is to minimize the amount of data within the revocation lists. In this way, both storage and transmission costs are minimized.

The approach of Crispo and Lomas to certificate revocation in [57] is to separate the certificate issuance and certificate revocation duties. In their system, *revocation authorities* are responsible for certificate revocation. Moreover, one revocation certificate is issued for each revoked certificate. The revocation certificate is issued by the revocation authority and includes the revocation request issued by the certificate owner. The revocation request is signed by the certificate owner and contains the whole certificate.

PGP [4,5] public keys and certificates may optionally have validity periods, but the current custom is not to have validity periods and make them non-expiring. Therefore, the PGP public keys and certificates are valid until they are revoked. PGP public keys can be revoked only by the public key owner and by issuing a key revocation certificate. Similarly, certificates can be revoked by certificate revocation certificates issued by the certificate signers. These revocation certificates invalidates the public keys/certificates, however the revoked objects do not disappear. Moreover, there is no revocation list concept in PGP. Revocation certificates are distributed like the public keys and certificates.

The most common approach is to use public key servers. The revocation certificates are kept in the public key servers together with the keys and certificates. If a key or certificate has a revocation certificate, then the verifier understands that this key or certificate is invalid.

The common approach of the authorization certificates [61,62] is not to have the certificate revocation concept at all. Consequently, there is no CRL in these systems. Instead, each certificate is assigned an appropriate validity period. The certificate times out after this period and needs *revalidation*[1]. Revalidations are performed either by the certificate issuers or by specific revalidation authorities. In order to revalidate a certificate, the certificate issuer re-signs the certificate content with a new time stamp. On the other hand, revalidation authorities sign the whole certificate content and the original signature on it, in order to revalidate a certificate.

2.6. Nested Signatures

Nested certificates are based on *nested signatures*. Nested signature is the signature over a document and other signatures over the same document that had been issued a priori. Since the scope of a nested signature contains previously issued signature(s) over the same document, the nested signatures are sometimes called *cascaded signatures* in the literature.

In the literature there are some applications, which use nested/cascaded signatures. Low and Christianson [76, 77] proposed the Self Authentication Proxies (SAProxies) to combine authentication with access control and resource management. The SAProxies use

[1] "Revalidation" is the term used by SPKI. SDSI uses "reconfirmation" instead of "revalidation".

classical certificates. The certificates for both the delegator/requestor and the delegatee certificates are bound into SAProxies by inclusion of the certificate signatures. Low and Christianson argue that the process of binding the certificate of the delegator/requestor to a SAProxy or to a request gives the self-authenticating property to the system. Nevertheless, the binding of a certificate to a SAProxy constitutes extra protection against bogus certificates, since a reference to the valid certificate is included in the SAProxy. Moreover the SAProxies can be cascaded by including a SAProxy into another one. In this system, all of the certificates for the SAProxy and the request signers must be issued by a commonly known CA and those certificates must be verified before the verifications of the SAProxies and requests.

The World Wide Web Consortium (W3C) Digital Signature (Dsig) project [78] aims to achieve a standard format for the trust of the addressable world wide web objects. The structure used in Dsig is called the *signature labels*. A signature label has the rating about an information source and is digitally signed by the labeller. The signature labels are nothing but signed assertions. In Dsig, it is possible to issue a signature label for another signature label. In this way, cascaded signatures become possible in Dsig. In the cascaded case, the sequential verifications of all of the signature labels are necessary to successfully find out the rating of the signed object.

In SDSI [62], signatures are exceptionally flexible and it is possible to have cascaded signatures for an object. In SDSI, cascaded signatures are cumulative, they effectively sign all previous material. An application of cascaded signatures is digital timestamping, where the second signature provides evidence that the first signature had already been created at the time the second signature was applied. Another use would be where an application program running on behalf of a principal signs the signature created by the principal, so that the server knows that the request not only came from an authorized principal, but from an authorized principal running the correct program. SDSI does not have certificate revocation lists. Instead, signatures are designated as needing periodic reconfirmation. A reconfirmation can be issued by renewing the signature of the original signer. Another approach for reconfirmation is to employ specific *reconfirmation principles* who apply

periodic cascaded signatures for the original object and its original signature. This is another application of cascaded signatures in SDSI. The Simple Public Key Infrastructure (SPKI) certificate structure [61] uses a similar approach for the revalidation of the certificates.

The concept of “signing certificate attribute” is proposed as an enhanced security service for S/MIME in [79]. The idea behind signing certificate attribute is to send the certificates, which are necessary for the verification of the signature over the signed message, to the receiver. The certificates are sent within the message that is signed by the sender. By using this method, the sender explicitly and unforgeably states the certificates to be used in signature verification. Consequently, some certificate substitution and re-issuing attacks are prevented.

3. NESTED CERTIFICATES AND NESTED CERTIFICATE PATHS

Nested certification scheme is the main contribution of the thesis. The certificates used in this scheme are called *nested certificates*. This section deals with the structure, issuance, verification and the characteristics of nested certificates and the usage of them in nested certificate paths.

3.1. Drawbacks of Classical Certificates and Motivations behind Nested Certificates

A *classical certificate* [2,4,5] is a digitally signed binding between the public key and the identity of a user. Classical certificates are signed by trusted *Certification Authorities* (*CAs*). *Classical certificate paths* are chains of classical certificates. The classical certificate and classical certificate path verification algorithms have been given in Section 2.4.5. The classical certification systems have some common characteristics that can be considered as drawbacks. Those drawbacks are listed below.

1. Classical certificate verification uses public key cryptography. Therefore, it is a time costly process. Moreover, *n* certificate verifications must be performed, in order to verify a classical certificate path with *n* certificates. Therefore, the time requirement of such a path verification is multiplied by *n*.

2. Moreover, the verifier is interested in only the public key of the target entity of a classical certificate path. However, the public keys of the intermediate CAs on the path need to be unwillingly found out by the verifier to reach the target entity. This situation unnecessarily degrades the certificate path verification time.

3. A classical certificate is issued to certify the correct public key of an entity. Naturally, such a certificate is verified to find out a correct public key. A public key is used to verify all of the certificates that had been issued by the certified entity before the certification time and to verify the certificates that will be issued after the certification time. However, some CAs may want to issue classical certificates with restricted usage. That means, a CA may want to issue a classical certificate for the public key of an entity in order to let the verification of it be useful to verify only the existing or specific certificates issued by that entity. However, classical certificates do not allow such certificate based restrictions. In this situation, the CAs may hesitate to issue classical certificates to some parties to which the reliance of the CAs is inadequate.

4. A CA, say CA_1, may issue a classical certificate for an entity by verifying another certificate, which has been issued for the same entity by another CA, say CA_2. To do so, CA_1 must trust CA_2. Otherwise, CA_1 cannot verify the existing certificate and consequently cannot issue a new certificate.

As can be seen from the above drawbacks, the classical certification scheme:

- is unnecessarily inefficient in certificate path verification (drawbacks 1 and 2),
- is unable to issue certificates with restricted usage (drawback 3),
- is unable to issue certificates in the case of missing trust information (drawback 4).

The motivation behind the nested certification scheme is to provide partial solutions for the above problems. The nested certificates can be briefly defined as “certificates for existing certificates”. They do not convey any trust information. Therefore, they can be issued in case of trust limitations. A nested certificate is not issued to certify a public key. It is issued to certify a single certificate. Therefore, the nested certificate issuer can apply certificate based restrictions. Moreover, the usage of nested certificates in certificate paths

significantly saves the overhead of the public key cryptography operations in the verification process.

3.2. Definitions and Terminology

In simple terms, a nested certificate is defined as "*a certificate for another certificate*". A classical certificate gives assurance about the correctness of the binding between the identity and the public key of an entity. Therefore, it is verified to find out correct public key of the certified entity. A nested certificate, on the other hand, certifies another certificate by assuring the legitimacy of the signature over it. Therefore, nested certificates are used to verify the signatures over other certificates. For example, certificate 1 is a classical certificate in Figure 3.1, since it is issued by *A* to verify the public key of *B*. Certificates 2 and 3 are nested certificates in Figure 3.1, since they are issued to certify other certificates. Certificate 2 is issued by *C* to certify certificate 1. Similarly, certificate 3 is issued by *D* to certify certificate 2. *Certification Authority (CA)* is the authorized issuer of a classical certificate. *Nested Certification Authority (NCA)* is the authorized issuer of a nested certificate. For example, *A* is a CA, *C* and *D* are NCAs in Figure 3.1. The certificate, which is certified by a nested certificate, is called as *subject certificate*. For example, certificate 1 is the subject certificate of nested certificate 2, in Figure 3.1. Similarly, certificate 2 is the subject certificate of certificate 3. A subject certificate can be a classical certificate or another nested certificate. For example, in Figure 3.1, the subject certificate 1 is a classical certificate and the subject certificate 2 is a nested certificate. Moreover, the nested certificate, which is used to certify the subject certificate, may be referred as a *certifier nested certificate* or an *issuer nested certificate*. For example, the nested certificates 2 and 3 are such certificates in Figure 3.1. Here it is worthwhile to remark that, certificate 2 is a both subject and certifier nested certificate. *Certificate Content* is the information part of a certificate and does not include the signature part. *Certificate Signature* is the signature of the CA or the NCA over the certificate content. The content of a classical certificate includes the identity and the public key of the certified entity. The content of a nested certificate includes the hash of its subject certificate content and the existing subject certificate signature.

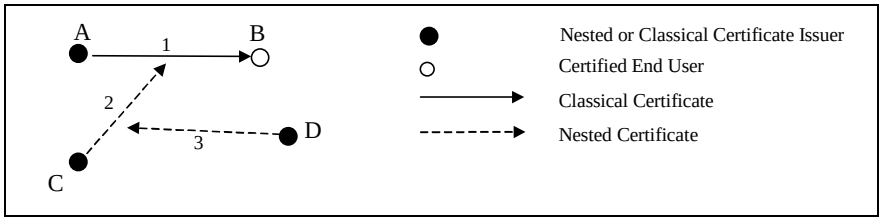

Figure 3.1. The relationships between nested and subject certificates

3.3. Structure

The nested certificate issuance is similar to classical certificate issuance. The nested certificates are issued by the digital signature of the NCA over the nested certificate content. The content of a nested certificate is related to its requirements. The two requirements of a nested certificate are:

(i) to certify that the subject certificate content has been signed by the claimed CA or NCA and
(ii) to certify that the subject certificate content has not been maliciously modified.

In order to satisfy the first requirement, a nested certificate contains the existing signature over the subject certificate content. In order to satisfy the second requirement, a nested certificate contains the hash of its subject certificate content. The hash of the subject certificate content can be obtained by applying an irreversible one way hash function [13, 15,16] to the subject certificate content. Since the subject certificate signature is contained in the nested certificate, the signature of NCA over the nested certificate content is considered as a nested signature. In the verification phase, the verifier computes the hash of the actual subject certificate and compares it with the hash of the subject certificate content stored in the nested certificate. If the comparison results in equality, then the verifier concludes that the subject certificate has not been maliciously modified.

By issuing a nested certificate, the NCA assures that the subject certificate is signed by the claimed issuer of the subject certificate and the subject certificate has not been modified maliciously. In order to issue a nested certificate, the NCA of the certifier nested certificate must have verified the signature over the subject certificate content successfully. The structure of a nested certificate is depicted in Figure 3.2. Formal representation of a nested certificate is also given below. In this representation, the notation used for classical certificates, which are explained in Section 2.4.5, is used. Given a subject certificate SC = $Cnt_{SC}|Sig_{SC}$, a nested certificate for SC issued by *NCA* is denoted as the following.

$$NC_{NCA}(SC) = Cnt_{NC}|Sig_{NC} = Cnt_{NC}|NCA_s[H[Cnt_{NC}]], \quad (3.1)$$

where Cnt_{NC} = *schash*|*scsig*|*Other*, *schash* = $H[Cnt_{SC}]$, which is the subject certificate hash, *scsig* = Sig_{SC}, which is the subject certificate signature and *Other* is the other managerial fields like the algorithms used, serial numbers, etc.

It is extremely important to realize that, the guarantee of the correctness of the information within the subject certificate content is not a requirement of a nested certificate. Therefore, the NCA of the nested certificate need not have an idea about the trustworthiness of the CA/NCA of the subject certificate, in order to issue a nested certificate for that subject certificate.

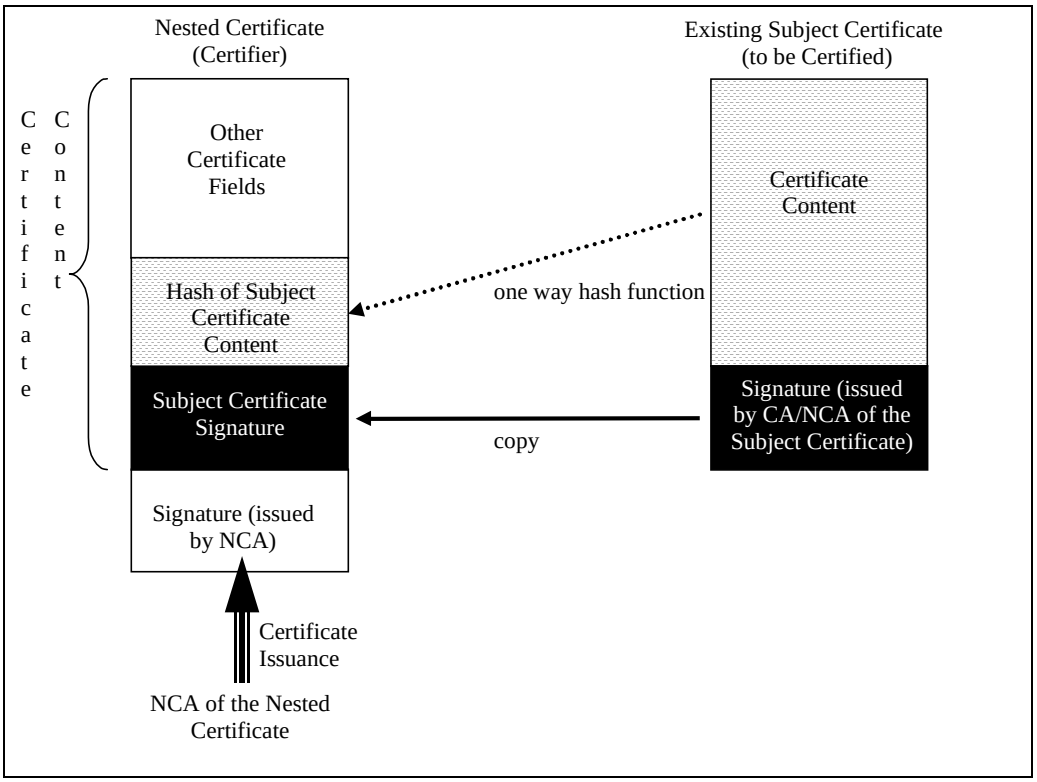

Figure 3.2. The structure of a nested certificate

3.4. Cryptographic Nested Certificate Verification Method

Since the nested certificates are issued via a digital signature over the certificate content, the cryptographic verification method, which has been explained for classical certificates in Section 2.4.5, can also be applied to nested certificate verification. That means, the digital signature over the nested certificate content is verified by employing public key cryptosystem based signature verification operations. Figure 3.3 gives the cryptographic nested certificate verification algorithm.

Given: A nested certificate, $NC_{NCA}(SC) = Cnt_{NC}|NCA_s[H[Cnt_{NC}]]$ issued by a trusted *NCA* and the correct public key of *NCA*, NCA_p.

The verifier applies the following algorithm to verify NC.

Verified_Hash $\leftarrow NCA_p[NCA_s[H[Cnt_{NC}]]]$
Calculated_Hash $\leftarrow H[Cnt_{NC}]$
IF Calculated_Hash = Verified_Hash THEN
  NC becomes verified
ELSE
  NC has not been verified

Figure 3.3. Cryptographic nested certificate verification algorithm

Verification of a nested certificate returns information about the correct hash value and signature over its subject certificate content. More formally, by the verification of a nested certificate, NC, the verifier finds out *schash*, which is equal to $H[Cnt_{SC}]$ and *scsig*, which is equal to Sig_{SC}. That information will be useful for the verification of the subject certificate SC using the *subject certificate verification* method. *Subject certificate verification* method is another verification method that can be applied to both classical and nested certificates. The subject certificate verification method is a consequence of nested certificates and will be explained in the subsequent section.

3.5. Subject Certificate Verification Method

The information found out by nested certificate verification is not sufficient to verify its subject certificate. By the verification of a nested certificate, only the correct hash value and correct signature over the subject certificate are found. In order to verify the subject certificate, the actual hash and the actual signature over the subject certificate must be compared with the ones stored in the nested certificate. Verification of a certificate as the

subject certificate of a nested certificate is called as *subject certificate verification*. Although the subject certificate verification method is a consequence of the nested certificates, it can be used to verify both the nested certificates and classical certificates, since the subject certificates can be of both types.

Given: a subject certificate, $SC = Cnt_{SC}|Sig_{SC}$, issued by a trusted authority and a legitimate nested certificate, NC, for SC, $NC_{NCA}(SC) = Cnt_{NC}|Sig_{NC} = Cnt_{NC}|NCA_s[H[Cnt_{NC}]]$,

where Cnt_{NC} contains the fields *schash*, which is equal to $H[Cnt_{SC}]$, and *scsig*, which is equal to Sig_{SC}.

The verifier applies the following algorithm to verify SC.

Calculated_Hash ← $H[Cnt_{SC}]$
IF *Calculated_Hash* = Cnt_{NC}.*schash* AND $Sig_{SC} = Cnt_{NC}$.*scsig* THEN
    SC becomes verified
ELSE
    SC has not been verified

Figure 3.4. Subject certificate verification algorithm

The subject certificate verification algorithm is given in Figure 3.4. As can be seen from this algorithm, having verified the nested certificate, *nc*, in order to verify its subject certificate, *sc*, the verifier follows the following two steps:

(a) The hash of the content of the actual *sc* is recalculated. This recalculated hash must be the same as the one stored within the *nc*.
(b) The actual signature over the content of the *sc* is compared with the subject certificate signature stored in the *nc*. These two signature values must be the same.

If the conditions in above steps are met and the issuer of *sc* is trusted as the verifier, then the verifier concludes that the *sc* is legitimate. The verifier must trust the issuer of *sc*, since the verification of *sc*, using the above steps, does not mean that the information stored within *sc* is correct. The verifier makes sure about correctness of the information contained in *sc*, if the issuer of sc is trusted. It is very important to point out that, in this way, the subject certificate *sc* becomes verified, with the same confidence, but without employing a signature verification method based on a public key cryptosystem. The phrase "the same confidence" means that the correctness level of the information found out by a subject certificate verification is the same as that of cryptographic certificate verification. Lemma 1 formalizes this conclusion. Moreover, the subject certificate can be another nested certificate. In this way, a nested certificate can be verified as the subject certificate of another nested certificate without using a cryptographic signature verification method. Lemma 2 formalizes the case where the subject certificate is another nested certificate.

Lemma 1: Suppose *A* and *B* are two authorities who are trusted to issue nested or classical certificates. Let *T* be an entity and *V* be the verifier who wants to verify the classical certificate of *T* to find out the correct public key of *T*. Suppose the authority *A* has issued a classical certificate (*cc*) for the entity *T* and the authority *B* has issued a nested certificate (*nc*) for *cc*. Figure 3.5 shows the certification relationships. If the *nc* is valid and the verifier *V* trusts both *A* and *B* as nested or classical certificate authorities, then the *cc* can be verified as the subject certificate of the *nc* and this verification has the same confidence as the cryptographic verification of the *cc*.

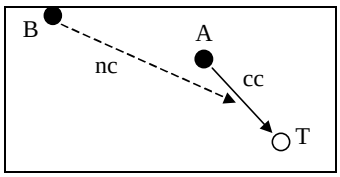

Figure 3.5. Certification Relationships in Lemma 1

Proof of Lemma 1: As described in Section 2.4.5, by the cryptographic verification of a digital signature over the *cc* using the correct public key of *A*, the verifier *V* can make sure about:

(i) the integrity of the content of *cc* and
(ii) the legitimacy of the signature of *A* over the content of *cc*.

If *A* is trusted for *V*, then the above two results of the verification imply that:
(iii) the information given in *cc* is correct. In other words, the public key of *T* specified in the *cc* is legitimate.

The belief in the integrity of the content of *cc* and the correctness of the signature over it are directly related to the one-way hash functions and the public key cryptosystem algorithms as described in Section 2.4.5. Here, it will be shown that the above three results can be obtained with the same confidence by the verification of the *cc* as the subject certificate of the *nc*.

By issuing the *nc*, *B* assures the integrity of the *cc* and correctness of the signature of *A* over it. However, this is not a direct assurance that can be verified whenever the *nc* is validated. In the *nc*, *B* gives the correct hash and the signature of *A* over the *cc*. Since the legitimacy of the *nc* and the trustworthiness of *B* are the premises of the theorem, the verifier *V* finds out the correct hash of the *cc* and the correct signature over it. Then, *V* applies the following three steps to verify the *cc* as the subject certificate of the *nc*.

(i) The verifier calculates the hash of the actual *cc*. If the calculated hash is the same as the hash within the *nc*, then *V* concludes that the *cc* has not been maliciously modified after the issuance of the *nc*, because otherwise the calculated hash would differ from the correct hash. In other words, that control verifies the integrity of the *cc*. The integrity control for the cryptographic certificate verification also relies on

the comparison of the existing and calculated hash values. Therefore, these two verification schemes have the same confidence for the integrity check.

(ii) The verifier compares the actual signature of *A* over the *cc* with the subject certificate signature of the *nc*. If both of them are the same, then *V* concludes that the *cc* has been signed by *A*. Because, *V* knew the correct signature over the *cc* due to *nc*. Therefore, if the actual *cc* bears the same signature, then *V* can infer that the *cc* contains the correct signature and that signature is issued by *A*. The signature control in this verification scheme has not been done by using public key cryptosystem based operations. However, the necessary cryptographic control had been done by *B* and assurance about it has been given in the *nc* by including the signature over the *cc* as the subject certificate signature. Since the verifier *V* trusts *B* as an authority and the *nc* is legitimate, *V* makes sure about the correctness of the signature of *A* over the *cc* with the same confidence as if *V* has cryptographically verified the *cc*.

(iii) The verifier *V* made sure about the integrity of the *cc* and correctness of the signature of *A* over it by above two steps. Since *V* trusts *A* as a CA, *V* can make sure about the correctness of the information within the *cc* and consequently finds out the correct public key of *T*.□

Lemma 2: Suppose *A* and *B* are two authorities who are trusted to issue nested certificates. Let *V* be the verifier. Suppose the authority *A* has issued a nested certificate (nc_1) and the authority *B* has issued another nested certificate for nc_1 (nc_2). Certification relationships are shown in Figure 3.6. If the nc_2 is valid and the verifier *V* trusts both *A* and *B* as the nested certificate authorities, then the nc_1 can be verified as the subject certificate of the nc_2 and this verification has the same confidence as the cryptographic verification of the nc_1.

Proof of Lemma 2: Lemma 2 is very similar to Lemma 1. The only difference is that the subject certificate is a classical one in Lemma 1, where it is a nested certificate in Lemma 2. As discussed in Section 3.4, the steps for the cryptographic verification of a nested certificate are the same for the cryptographic verification of a classical certificate. The only difference is the verified information. By the verification of a classical certificate, the public key of the certificate owner is verified. On the other hand, by the verification of a nested certificate, the correct hash and the legitimate signature over the subject certificate are found. The information within the subject certificate has not been used as a precondition in the proof of Lemma 1. Therefore, Lemma 2 can be proven by following exactly the same arguments as in the proof of Lemma 1.□

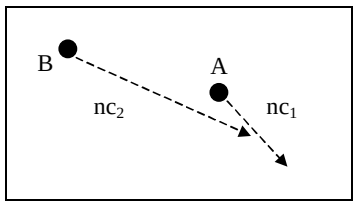

Figure 3.6. Certification Relationships in Lemma 2

3.6. Nested Certificate Paths

Nested certificates can be used to certify other nested certificates. In this way, a sequence of nested certificates is achieved. Such a sequence is called a *nested certificate series*. Each nested certificate of a nested certificate series is used to certify the next nested certificate. However, the eventual aim of using a nested certificate series is to verify a classical certificate at the end. Therefore, the last nested certificate of a nested certificate series must certify a classical certificate. A nested certificate series together with a classical certificate at the end are called a *nested certificate path*. There can be zero or more nested certificates on a nested certificate path, but there must be exactly one classical certificate at the end. This classical certificate is the certificate for the *target entity* of the nested certificate path. The target entity can be an end user or a CA/NCA.

A nested certificate path with *k* nested certificates is called a *k-nested certificate path*. A 0-*nested certificate path* (Figure 3.7a) is a nested certificate path with no nested certificates. In other words, it is just a classical certificate issued by a CA. Other example nested certificate paths with 1, 2 and 5 nested certificates are shown in Figure 3.7b, Figure 3.7c and Figure 3.7d respectively. A generic *k*-nested certificate path (the certificates nc_k,nc_{k-1},nc_{k-2} … nc_3,nc_2,nc_1,cc_0) is shown in Figure 3.8. In a *k*-nested certificate path, each nested certificate is used to verify its subject certificate. At the end of a series of subject certificate verifications, the classical certificate, cc_0, of the target entity, *T*, is verified as the subject certificate of the last nested certificate, nc_1, of the *k*-nested certificate path. Only the first nested certificate, nc_k, of a *k*-nested certificate path is verified cryptographically using the public key of its issuer, A_k. The other certificates of the path are verified as the subject certificates. Verification of a certificate as a subject certificate is faster than the cryptographic verification of the same certificate. Consequently, nested certificate path verification is more efficient than the classical certificate path verification. The details of the performance evaluation of the proposed technique will be presented in Section 5.

The natural evolution of nested certificate paths can be explained as follows. In a classical certificate path, which has been shown in Figure 2.7, each CA can validate the certificates, which have been issued by its immediate successor, since it already knows the public key of its immediate successor. Consequently, each CA can issue nested certificates for all of the certificates that had been issued by its successor. This rule can be applied to a classical certificate path to obtain a structure in which each classical and nested certificate is certified via a nested certificate. From such a structure, it is possible to extract a nested certificate path. The details on the nested certificate path formation will be discussed in Section 4.

To verify the classical certificate via such a nested certificate path, it is necessary to use the public key of the first nested certificate issuer of the path. The public keys of other nested and classical certificate issuers need not be found out. One may argue that, the classical certificate path verification has a similar property. That means, it is sufficient to know the public key of the first issuer of a classical certificate path. This argument is

correct, but in a classical certificate path, all of the certificates are verified cryptographically to find out the public key of the next issuer on the path. However, in a nested certificate path, the intermediate nested certificates are not verified cryptographically and the public keys of the intermediate issuers are not found out. In this way, the verification of the classical certificate becomes faster via a nested certificate path as compared to a classical certificate path.

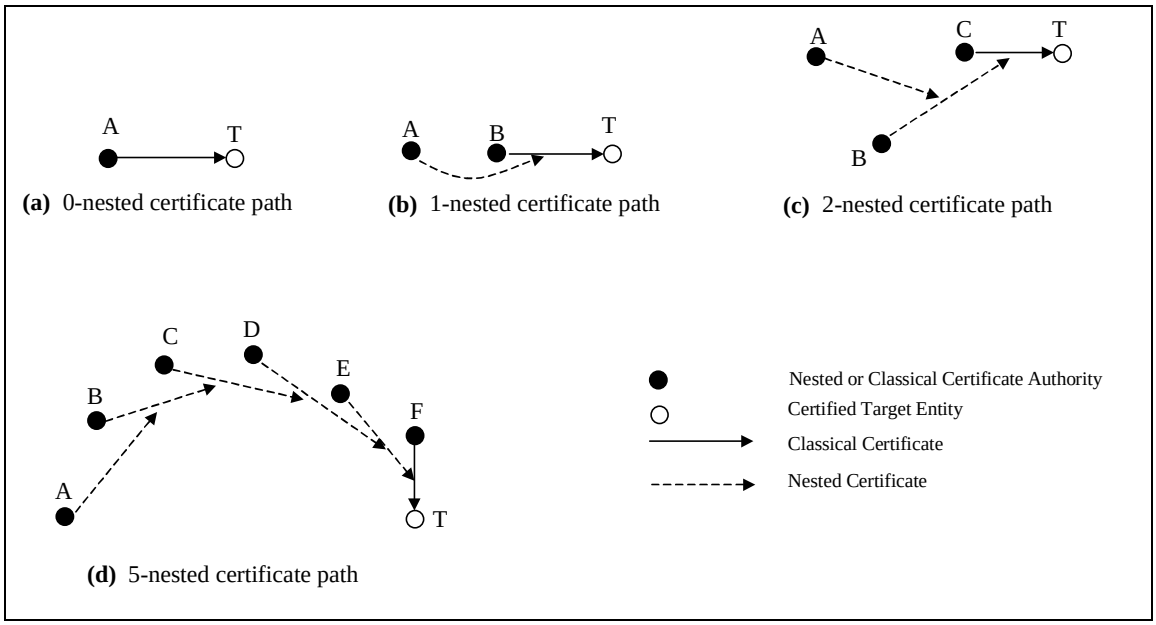

Figure 3.7. Some example nested certificate paths

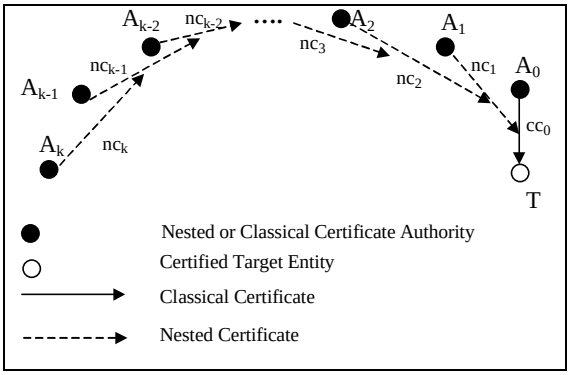

Figure 3.8. A generic *k*-nested certificate path

3.6.1. Confidence Proof of Nested Certificate Path Verification Method

In this section, it will be proven that the verification of the classical certificate of the target entity via a nested certificate path has the same confidence as the cryptographic verification of that certificate. Under this consideration, 0-nested certificate path verification is a tautology, since both 0-nested certificate path verification and the cryptographic verification are the same processes. Each nested certificate of a nested certificate path is used to verify the next certificate as a subject certificate. The first nested certificate of a nested certificate path is verified cryptographically using the public key of the first entity of the nested certificate path. Lemma 3 formalizes nested certificate path verification assuming that the first nested certificate of the nested certificate path is valid. The verification of the first certificate will be considered in Theorem 1.

Lemma 3: Consider the generic *k*-nested certificate path in Figure 3.8. Let A_0 be a CA and A_i, i=1 .. k, be k NCAs. Suppose that A_0 has issued a classical certificate (cc_0) for the target entity T and A_1 has issued a nested certificate for cc_0 (nc_1). Moreover, suppose that A_i has issued a nested certificate (nc_i) for the nested certificate that A_{i-1} has issued (nc_{i-1}), $\forall i$= 2 .. k. If the nc_k is valid and the verifier V trusts the authorities A_i, $\forall i$=0 .. k, then the classical certificate cc_0 can be verified via the *k*-nested certificate path by applying the following steps and this verification has the same confidence as the cryptographic verification of cc_0.

Steps that the verifier V must follow to verify cc_0 via the *k*-nested certificate path:

(i) V verifies nc_i as the subject certificate of nc_{i+1}, $\forall i$=k-1 .. 1,

(ii) finally, V verifies cc_0 as the subject certificate of nc_1.

Proof of Lemma 3: The proof is by induction on *k*, the total number of nested certificates on the *k*-nested certificate path. The proof uses Lemma 1 and Lemma 2 of Section 3.5

Stage 1 (*k* = 1): If *k* = 1, then there is only one nested certificate on the *k*-nested certificate path. This is the case described in Lemma 1. Therefore by Lemma 1, cc_0 can be verified via nc_1 and this verification has the same confidence as the cryptographic verification of cc_0.

Stage 2 (*k* = *n*): Assume that if there are *n* nested certificates on the *k*-nested certificate path and the nc_n is valid, then the classical certificate cc_0 can be verified via this *k*-nested certificate path starting with nc_{n-1} and this verification has the same confidence as the cryptographic verification of cc_0.

Stage 3 (*k* = *n*+1): This is the case where there are *n*+1 nested certificates on the *k*-nested certificate path. Since nc_{n+1} is valid, nc_n can be verified as the subject certificate of nc_{n+1} and this verification has the same confidence as the cryptographic verification of nc_n, by Lemma 2. Having verified the nc_n, by the assumption in the stage 2, the classical certificate cc_0 can be verified via the *k*-nested certificate path and this verification has the same confidence as the cryptographic verification of cc_0.□

Lemma 3 assumes that the first nested certificate of the *k*-nested certificate path is valid. However, the first nested certificate, nc_k in Figure 3.8, of a *k*-nested certificate path must be verified cryptographically using the public key of the first NCA, A_k in Figure 3.8. The complete *k*-nested certificate path verification is formally given by the next theorem.

Theorem 1: Consider the generic *k*-nested certificate path in Figure 3.8. Let A_0 be a CA and A_i, i=1 .. *k*, be *k* NCAs. Suppose that A_0 has issued a classical certificate (cc_0) for the target entity *T* and A_1 has issued a nested certificate for cc_0 (nc_1). Moreover, suppose

that A_i has issued a nested certificate (nc_i) for the nested certificate that A_{i-1} had issued (nc_{i-1}), $\forall i = 2 .. k$. If the verifier V knows the correct public key of A_k and trusts the authorities A_i, $\forall i = 0 .. k$, then the classical certificate cc_0 can be verified via the k-nested certificate path by applying the following steps and this verification has the same confidence as the cryptographic verification of cc_0.

Steps that the verifier V must follow to verify a k-nested certificate path are:

(i) Firstly, V verifies the nc_k by employing a public key cryptosystem based signature verification algorithm which uses the public key of A_k,
(ii) V verifies nc_i as the subject certificate of nc_{i+1}, $\forall i = k-1 .. 1$,
(iii) finally, V verifies cc_0 as the subject certificate of nc_1.

Proof of Theorem 1: Since the verifier V knows the correct public key of A_k, V can apply cryptographic signature verification algorithm over the nc_k to verify it as explained in Section 3.4. After this verification, V can make sure about the validity of nc_k. If the nc_k comes out to be valid, then V can verify cc_0 via the k-nested certificate path and this verification has the same confidence as the cryptographic verification of cc_0, by Lemma 3.□

3.7. Characteristics of Nested Certificates and the Differences between Nested and Classical Certificates

In this section, the basic characteristics of the nested certificates will be examined by comparing them with the classical certificates. These characteristics and differences will be examined in two parts: (i) structural and semantic, (ii) operational characteristics and differences. The advantages and the disadvantages of the nested certificates will be discussed in Section 3.8.

3.7.1. Structural and Semantic Characteristics and Differences

Both classical and the nested certificates are issued by trusted authorities. The contents of both types of the certificates are digitally signed by such authorities. However, the contents of the classical and the nested certificates are different. Therefore, they have different meanings.

A classical certificate is a *certificate for entity*. Therefore, it is shown as a node-to-node relationship in Figure 3.1. A classical certificate contains the identity and the public key of an entity in its content. The meaning of the CA signature over the content is the guarantee of the correctness of the binding between the public key and the identity of the certified entity.

On the other hand, a nested certificate is a *certificate for certificate*. Therefore, it is shown as a node-to-certificate relationship in Figure 3.1. The content of a nested certificate includes the hash of its subject certificate and the subject certificate signature. The NCA signs this content to assure the integrity of the subject certificate and the legitimacy of the existing signature over the subject certificate.

Moreover, the NCAs need not trust the subject certificate issuers and subject certificate holders, in order to issue nested certificates. Here, it is very important to understand that, within a nested certificate, the NCA does not give any guarantee about the correctness of the information in the subject certificate content. It guarantees only the integrity of the subject certificate content and the legitimacy of the signature over it. However, this does not mean that the subject certificate content contains incorrect information. The NCAs of the nested certificates have no assumptions about the trustworthiness of the subject certificate issuers. Therefore, NCAs cannot comment on the correctness of the information content of the subject certificates. The knowledge of the correct public key of an authority is not the same as being able to trust that authority. The NCA of the nested certificate only knows the correct public key of the CA/NCA of the

subject certificate, so that the NCA was be able to verify the signature over the subject certificate. This is a computational process and gives only the information that “the subject certificate has been signed by the claimed authority”. Although the certifier NCA has no assumption about the trustworthiness of the CA/NCA of the subject certificate, since it has made sure that the subject signature is legitimate, it will not hesitate to issue a nested certificate for this subject certificate. Since the NCA does not guarantee the correctness of the information within the subject certificate content, the verifier must also trust the issuer CA/NCA of the subject certificate to completely verify the subject certificate in the verification phase.

The scope of assurance of a classical certificate is the legitimacy of a public key. A public key is used to verify all of the certificates that has been and will be issued by the public key owner. Moreover, once a classical certificate has been issued, the CA cannot later monitor the usage of the certificate and the certified public key. On the other hand, the scope of assurance of a nested certificate is just the correct hash and the correct signature over a single subject certificate content. The verifier must also verify the subject certificate via the nested certificate and trust the CA/NCA of the subject certificate in order to proceed.

As a result of last two paragraphs, it can be said that, the scope of assurance for a nested certificate is restricted as compared to classical certificates. In other words, the guarantee given in a classical certificate is more effective than the guarantee in a nested certificate.

Nested Certificates use nested/cascaded (so called nested) signatures. Nested signature is the signature over a document and other signatures over the same document that had been issued a priori. Some systems that use nested signatures have been explained in Section 2.5. However, the application area of the nested signatures in nested certificates is different from other systems that use nested signatures. The scope of nested certificates is just certificate issuance and verification. The nested signatures over nested certificates are not used for authorization, access control delegation, revalidation, attack prevention or

authentication. The nested signature over a nested certificate is the guarantee for nothing but the legitimacy of the signature over the corresponding subject certificate. In the context of nested certificates, nested signatures do not guarantee the correctness of the information within the subject certificate. Moreover, the structure of nested certificates is quite different from the structures of other systems. SAProxies [76,77] contain only the signature part of a certificate. SDSI [62] cascaded signatures sign all of the previous material. However, a nested certificate contains the signature over its subject certificate and the hash of its subject certificate content. The latter is used for the verification of the subject certificate signature and the former is used for the integrity control. Apparently, a nested certificate is issued only for a single certificate. However, it is possible to issue a nested certificate for another nested certificate. In this way, arbitrary depth nesting becomes possible without holding all of the previous material.

The “signing certificate attributes” of S/MIME [79] is the closest study to the nested certificates in terms of the usage of nested signatures. Signing certificate attributes are used for a specific application, which is the verification of signed MIME messages. The primary aim of signing certificate attributes of S/MIME is to prevent some attacks based on substitution and re-issuing of certificates that are explained in [79]. Moreover, in this way, the sender may also specify and restrict the certificate usage in an unforgeable manner. The basic difference between the nested certificates and signing certificate attributes is the difference in the application scope. The nested certificates are to be used within PKIs, whereas the signing certificate attributes are to be used only in S/MIME. Moreover, the whole certificate path is included in signing certificate attributes, whereas a nested certificate is only for a single certificate. Here it is worthwhile to mention that the “signing certificate attributes” of S/MIME is introduced to the author at the end of his Ph.D. study. Therefore, the thesis study is entirely independent of signing certificate attributes of S/MIME.

The classical certificates, nested signatures and the nested certificates are analogous to some notary operations in the real world. A classical certificate is the digital equivalent of a printed signature declaration approved by a notary. That declaration is done by the

printed signature owner and the notary approves the legitimacy of the signature and the identity of the owner. A nested signature is the equivalent of notary approval over a personally signed document. Here, the notary approves that the document has been signed by the claimed person, does not approve the correctness of the information content of the document. Finally, a nested certificate is analogous to the approval of another notary, $notary_2$, over a $notary_1$ approved printed signature declaration of an entity. Here the $notary_2$ is analogous to the nested certificate issuer and its approval is the nested certificate. The $notary_1$ approval for the signature declaration is analogous to the subject certificate and $notary_1$ is the subject certificate issuer. The $notary_2$ approves only the signature of the $notary_1$ over the original declaration, it does not assure the correctness of the original declaration. In this way, the ones who trust the $notary_1$, but cannot verify the signature of it over the original declaration, may verify this declaration via the signature of the $notary_2$.

3.7.2. Operational Characteristics and Differences

The classical certificates can be verified cryptographically as explained in Section 2.4.5. The usage of nested certificates allows a classical certificate to be verified as the subject certificate of a nested certificate. Similarly, the nested certificates can be verified either cryptographically or as the subject certificate of another nested certificate. As explained in Section 3.5, the subject certificate verification method does not employ any signature verification operation based on a public key cryptosystem. Therefore, verification of a certificate as a subject certificate is computationally more efficient as will be discussed in Section 3.8.5.

By the verification of a classical certificate, the verifier finds out the legitimate public key of the certificate owner. On the other hand, by the verification of a nested certificate, the verifier finds out the correct hash and the correct signature over its subject certificate content.

The classical certificates have lifetimes in order to have an upperbound for the usage of the certified public key in case of a compromise of the private key corresponding to that public key. Since the nested certificates do not certify a public key directly, they need not have lifetimes and consequently they need not be renewed.

3.8. Advantages and Disadvantages of Nested Certificates

Since the nested certification scheme allows a snapshot certification only for one certificate with restricted assurance as compared to the classical certificates, it constitutes an alternative certification scheme for the certification authorities. This is an advantage of the nested certificates for the certification authorities. The most important advantage of nested certification is the significant efficiency improvement in verification processes. Moreover, the nested certificates are useful to verify certificates that had been issued with currently revoked keys. On the other hand, the disadvantage of the nested certification is the necessity to issue a large number of nested certificates. The advantages and disadvantage of nested certificates will be detailed below.

3.8.1. Flexibility in Certificate Issuance

The restricted assurance characteristic of nested certification allows the NCAs to issue nested certificates for the cases where they want to give restricted assurance. In this way, the NCAs become more flexible in certificate issuance, as will be detailed in this subsection.

By issuing a classical certificate, a CA implicitly allows the verifiers to verify all of the certificates that had been issued by the certificate owner. For example consider Figure 3.9a. By issuing certificate 1 for *B*, *A* implicitly allows the verification of the certificates 2, 3 and 4 that *B* has issued for *C*, *D* and *E* respectively. Because, by the verification of

certificate 1, the verifier finds out the correct public key of *B* and uses it to verify the certificates 2, 3 and 4.

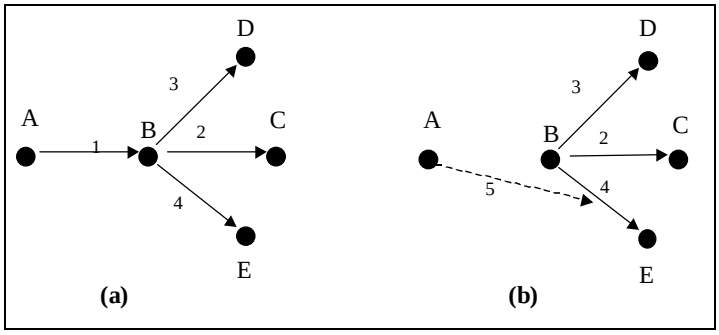

Figure 3.9. An example to show the flexibility in nested certificate issuance

The CAs may not be comfortable with this implicit allowance. They may want to specify the certificates to be certified by using the public key specified in a classical certificate. However, they cannot do anything, to avoid this implicit allowance, by using the classical certificates. Such an avoidance is possible by using the nested certificates. A nested certificate is issued to certify only a single subject certificate, not to certify a public key. Therefore, the verification of a nested certificate is useful only to verify its subject certificate. Other certificates, which have been issued by the subject certificate issuer, cannot be verified by using a single nested certificate. For example, consider Figure 3.9b. Suppose that *A* wants to issue a certificate for *B* such that it would be useful only to verify certificate 4. Therefore, instead of certificate 1 in Figure 3.9a, *A* issues a nested certificate (certificate 5) for certificate 4. In this way, the verifier, who has verified certificate 5, will be able to verify only certificate 4. The verifier will not be able to verify certificates 2 and 3 via certificate 5.

As a counter argument to the above example, one may state that “why does not *A* issue a classical certificate directly for *E*, instead of the nested certificate 5 ?”. First of all, it must be mentioned that *A* must trust *B* as an introducer, in order to issue a classical certificate for *E* via *B*. If *A* trusts *B*, then *A* can issue a classical certificate for *E*: However, if *A* does not trust *B*, then *A* cannot verify the public key of *E* and consequently, cannot

issue a classical certificate for it. Here, one may state that since *A* has issued a classical certificate for *B*, *A* has to trust *B*. This argument may not be true always, because a classical certificate issuance is just to assure the correctness of the public key of the certified entity. The classical certificate issuer need not trust the certified entity all the time. Therefore, *A* may or may not trust *B*. If A trusts B, then *A* can issue a classical certificate for *E*, but otherwise *A* cannot. On the other hand, in order to issue a nested certificate, the NCA need not trust anyone including the subject certificate issuer. Therefore, whatever the trust information is, *A* can issue the nested certificate 5 for the classical certificate for *E*. As a result, it can be said that the nested certificates can be issued independent of trust information. This advantage will be utilized in the nested certificate propagation strategies in NPKI, which will be described in Sections 4.6 and 4.7.

As can be seen above, the authorities will be able to reflect their choices in certificate basis rather than public key basis by using nested certificates. This enhances the granularity of the system and allows the authorities to customize their requirements. In this way, the authorities become more flexible.

3.8.2. Flexibility in Verification of Revoked Classical Certificates

Besides the validity of the signature over a classical certificate, the verifier should also check whether that classical certificate is revoked or not. Certificate revocation is generally maintained by Certificate Revocation Lists (CRLs). A detailed explanation of certificate revocation in X.509 can be found in [2]. By the revocation of a classical certificate, the public key within that certificate becomes invalid. Therefore, the certificates, which had been issued by the corresponding private key of this revoked public key before the revocation, could not be verified after the revocation time. However some verifiers may prefer to be able to verify such certificates, since they had been issued before the revocation time. Such certificates can be verified via nested certificates. If a certificate, which had been issued by an authority whose public key is revoked, has a nested certificate, then that certificate can be verified as the subject certificate of the nested certificate. In this case, the verifier does not need to know and use the public key of the

certificate issuer to verify it. Consequently, the invalidity of this public key does not cause any problem.

3.8.3. Unnecessity of Renewal

The classical certificates have lifetimes in order to have an upperbound for the usage of the certified public key in case of a compromise of the private key corresponding to the public key in the certificate. Since the nested certificates do not certify a public key directly, they do not need to have lifetimes and consequently they do not need to be renewed.

3.8.4. Alternative Certificate Paths

The differences between the classical certificates and the nested certificates make the nested certification scheme an alternative certification model. This alternative is advantageous for the certificate authorities as explained in Section 3.8.1. The nested certificates are also advantageous for the verifiers, since the verifiers can form alternative certificate paths by using the nested certificates.

3.8.5. Efficient Subject Certificate and Nested Certificate Path Verification

The most important advantage of the nested certification scheme is the computational efficiency of the subject certificate verification method as compared to the cryptographic certificate verification method. The reason behind this efficiency is the difference between the cryptographic certificate verification and the subject certificate verification methods. As explained in Section 3.5, subject certificate verification method does not employ any public key cryptosystem operation. A single hash computation and two comparisons are enough to verify a subject certificate via a nested certificate. Hash calculation is also a part of the cryptographic certificate verification method. Therefore, the time spent for the

public key cryptosystem operation is the saving of the subject certificate verification method.

The computational performances of the different hash algorithms are examined in [80,81,82,83]. Moreover, the computational performances of the public key cryptosystem operations of different algorithms are examined in [82,84]. Those results show that hash computation is more efficient than the public key cryptosystem based signature verification. Therefore, the relative efficiency improvement of the subject certificate verification over the cryptographic certificate verification is remarkable.

Similarly, nested certificate path verification is also more efficient than classical certificate path verification. All of the certificates of a classical certificate path are verified using cryptographic certificate verification method. On the other hand, in the nested certificate paths, only first certificate is verified using the cryptographic certificate verification method. Other certificates are verified using subject certificate verification method. Since subject certificate verification method is more efficient than the cryptographic certificate verification method, nested certificate path verification method is also more efficient than the classical certificate path verification method.

In order to present the relative computational efficiency improvement of the subject certificate verification method over the cryptographic certificate verification method, analytical and simulation studies are performed. Similar analytical and simulation studies are performed for the relative computational efficiency improvement of the nested certificate path verification method over the classical certificate path verification method. The results obtained from these studies will be given in Section 5.

3.8.6. Nested Certification Overhead

A classical certificate is verified to validate a public key. Moreover, that public key is used to verify all of the certificates that had been issued by the owner of that public key. However, a nested certificate is used to eventually verify only one certificate. The advantages of this situation have been explained previously. However, if a NCA wants to certify different certificates issued by an authority, it has to issue separate nested certificates. Moreover, in order to benefit from the efficiency of the subject certificate verification method in certificate paths, several nested certificates must be issued. This situation causes the NCAs to issue a large number of nested certificates.

On the other hand, the usage of nested certificates improves the time complexity of the certificate verification as discussed in Section 3.8.5. From this point of view, the nested certification can be thought as a way to convert the time complexity for the verifiers into time complexity for the authorities. Since the certificates are issued once and the authorities are dedicated machines, this situation can be preferred in many systems for which minimization of the time complexity for verifiers is more important. The analysis for the number of nested certificates that must be issued and the analysis of the trade-off between the time complexities will be given in more detail in Section 5.4.

4. NPKI: NESTED CERTIFICATE BASED PUBLIC KEY INFRASTRUCTURE

Public Key Infrastructure (*PKI*) is a network of classical certificates. In a PKI, the nodes are the CAs and the end users, where the arcs are the classical certification relationships among them. The PKIs are necessary to make the public keys of all of the users available to each other. Classical certificate paths are drawn from PKIs and verified by the verifiers to find out public keys of different users. The X.509 [2] based PKI structures are mostly hierarchical as discussed in Section 2.4.1. On the other hand, the PKI incorporated in PGP [4,5] has a graph structure as discussed in Section 2.4.2.

Nested certificates must be embedded into the PKIs for their usages to become widespread. A PKI, which is capable of handling nested certificates as well, is called the *Nested certificate based PKI* (*NPKI*). In NPKI, the classical certificates are used together with the nested certificates. There are two main approaches for the construction of NPKI. One of them is *free certification* approach. Other approach is *transition from an existing PKI* approach.

1. In the construction of a NPKI using the *free certification* method, each CA/NCA is free to issue classical certificates to other CAs/NCAs/end users or nested certificates to other certificates. Therefore there is no enforcement to issue nested certificates in this method. The primary aim of this method is to construct a flexible NPKI. The efficiency in path verification time is just a consequence of nested certificates used in NPKI. The characteristics, structure and other details of this method will be explained in Sections 4.1 through 4.5. Moreover, the NPKI referred in these sections must be regarded as the NPKI formed by this method.

2. The construction of a NPKI by *transition from an existing PKI* approach deals with a systematic deployment of nested certificates. In this method, each authority, *A,* is required to issue nested certificates for all of the certificates that have been issued by

the authorities for which *A* had issued classical certificates. The only goal of this method is to deploy the nested certificates throughout the infrastructure and to have quickly verifiable certificate paths. The details and the variations of this method will be described in Sections 4.6 and 4.7.

4.1. General Characteristics and Assumptions

In this subsection, some basic design criteria and assumptions about NPKI will be listed. The design criteria of NPKI are to be *generic*, *flexible* and to have a *user oriented trust-management*. Those criteria are detailed below:

Generic: The NPKI is not aimed to work for a specific application like e-mail, electronic payment, etc. Instead, it is designed to demonstrate the results and the advantages of adding the nested certificate concept to the classical PKIs. That is why a generic basic design has been preferred and application specific design decisions have been avoided. The NPKI can be considered as a framework to adopt the nested certificates into PKIs. It is believed that, specific NPKI designs for different applications will be developed in the future. Secure and private e-mail, electronic commerce and payment are the most suitable systems to have a NPKI.

Flexible: NPKI is a general system. Therefore, it should be flexible in order to port it to a specific application. Since no application specific design decision has been assumed in NPKI, it can be easily adapted to a specific application. Moreover, one of the aims of the nested certificate usage in the NPKI is to provide more flexible certification and verification structure.

User oriented trust management: The key component of the NPKI is the user. The CAs, certified objects and the verifiers are the users of NPKI. NPKI will be a X.509 based PKI. Because of the regulative behavior of the standard X.509 [2] certificate structure, the

users are retained in order to have a high degree of freedom and flexibility to construct their own trust policies. The NPKI will allow the highest degree of freedom, flexibility and ease that X.509 allows. For example, policy identifiers and cross certificates are such X.509 structures. Furthermore, the users of NPKI will have extra flexibility to constitute their own trust and certification policies because of the use of nested certificates.

The general design assumptions about the classical certificates of NPKI are as follows:

Certificate Structure: In NPKI, the classical certificate structure will conform to X.509 standard. The reason is that X.509 is a widely accepted international standard in the area of digital certificates and the new products which uses digital certificates are getting more and more X.509 compliant. However, the X.509 standard has been developed only for the classical certificates. The X.509 conformance issues regarding the nested certificates will be discussed in Section 4.8.

Topology: There is no restriction for the topology of the certificate network in NPKI. It can be a tree or a directed graph structure. The graph structure, if any, can be constructed using cross certificates or by having possessed several certificates from different CAs. In the design of NPKI, the cross certificates will not be differentiated from the classical certificates.

Trust and certificate path processing: In NPKI, the users will have partial freedom to choose the trusted CA groups via policy identifiers that X.509 allows. Moreover, the users will be able to specify their trusted CAs with the corresponding public keys, in their local databases. In this way, the users will be able to specify their directly trusted starting CAs of the certificate paths. The standard X.509 certificate path processing rules will be used in NPKI.

Network address types: Since NPKI is not for a specific application, there is no need to restrict the type of the network address. Thus, in NPKI, the network addresses specified in the classical certificates can be any address (e-mail, URL, etc.) that the X.509 standard allows.

Who signs the certificate? There are two main approaches about the executive object for the digital signatures. One of them is that the digital signatures are issued by the *keys*. Second approach is that the signatures are issued by *individuals* who own the keys. Reiter and Stubblebine [85] have given a good discussion of this subject and some other design criteria for the PKIs. The widely accepted answer to this question is to have the keys as the executive signing object for the digital signatures and this answer will be accepted for NPKI. Therefore, we will try to validate the keys, rather than the owners, within the certificates as the signer of the next certificate on the certificate path.

Certificate Revocation: Certificate revocation in NPKI is not detailed in the thesis. CRLs can be used for classical certificate revocation. The intuition is that the nested certificates need not revoke, since they do not directly certify a public key. However, a more detailed analysis must be done to confirm this intuition. Detailed nested certificate revocation analysis is left as a future research topic.

The assumptions about the nested certificates and the NCAs in NPKI are explained below.

A new type of certificate: The content and the usage of a nested certificate are different from a classical certificate. Thus, in NPKI, the nested certificates are considered as a new certificate type.

CAs and NCAs are the same entities: Although the nested certificates are different from the classical certificates, they can be issued by the same authorities. Therefore, the CAs and NCAs can be the same NPKI entities, but an end user cannot be a CA or a NCA.

Trust Considerations for NCAs: In NPKI, an authority can issue a classical certificate as well as a nested certificate, because of the above assumption. However, a user may trust an entity differently as a CA and as a NCA, since assurances within classical and nested certificates are different each other. In NPKI, the trust to the CA and the trust to the NCA are distinguished. That means, a user will have two attributes for the trust information of an authority. One of them is about trust information as a CA; the other is for the trust information as a NCA.

X.509 compliance: The X.509 standard has not been designed for certificates other than the classical certificates. Therefore, the implementation of NPKI with the above assumptions in compliance with X.509 requires some minor modifications and interpretation changes to the X.509 standard. Those modifications and changes will be discussed briefly in Section 4.8.

4.2. General Structure of NPKI

The NPKI employs both the classical certificates and the nested certificates. Actually, NPKI is a network of those types of certificates. The cross certificates are considered as classical certificates in NPKI. The nodes of the NPKI network are the end users, CAs and NCAs. In NPKI, the end users are distinguished from CAs and NCAs, since X.509 certificate structure enforces such discrimination. However, the CAs and the NCAs can be the same entities. Therefore, the CAs and the NCAs are not differentiated in NPKI.

The arcs of the NPKI network will be the certificates. A classical certificate is a certificate from a CA to an end user or to another CA, so it is a node-to-node arc in the representation. On the other hand, a nested certificate is issued to certify a subject certificate. Therefore, a nested certificate is a certificate from a NCA to another certificate. Thus, a nested certificate is represented as an arc from a node to another arc. Moreover, the NPKI also supports the hierarchical nested certificates. A *hierarchical nested certificate* is a nested certificate of which the subject certificate is another nested certificate. The representation of a hierarchical nested certificate is again an arc from a node to another arc. However, the subject arc for a hierarchical nested certificate is another nested certificate. An example NPKI network is depicted in Figure 4.1. The hierarchical and normal nested certificates do not have different structures. Therefore, they have not shown differently in Figure 4.1.

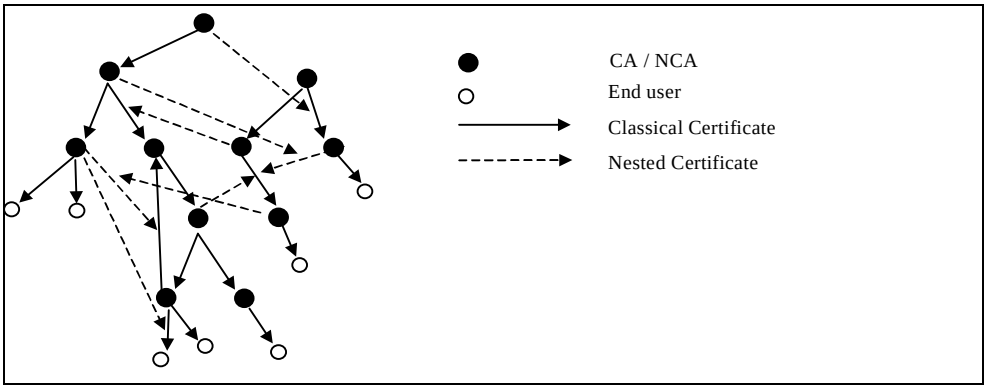

Figure 4.1. An example NPKI network

4.3. Certificate Path Processing in NPKI

In NPKI, in order to verify the binding between the identity and the correct public key for a target entity, the verifier must find a valid certificate path for which the end point is the target entity. Such a path is named as the *NPKI certificate path* and it is derived from the NPKI. The starting point of a NPKI certificate path is either the verifier itself or another CA or NCA to whom the verifier directly trusts. To have a CA or NCA other than the verifier as the starting point of the NPKI certificate path is allowed only if the verifier

has an entry for that CA or NCA in the local *trusted authorities database*. The verifier has also to know the correct public key of the first entity of the NPKI certificate path. Moreover, the last certificate of the NPKI certificate path must be a classical certificate, because this certificate is used to verify the public key of the target entity and only classical certificates can certify public keys. The intermediary certificates of the NPKI certificate path can be either classical or nested certificates. An example NPKI certificate path is shown in Figure 4.2. In this example path, the certificates 1,3,4,7 and 8 are classical certificates, whereas the certificates 2,5 and 6 are nested certificates.

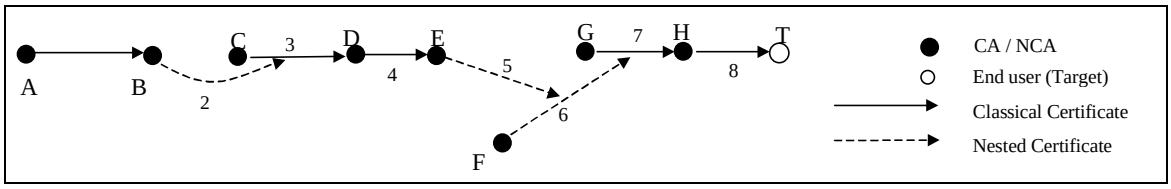

Figure 4.2. An example NPKI Certificate Path

All of the certificates of a NPKI certificate path are verified one by one sequentially. The verification process starts with the first certificate of the NPKI certificate path and ends with the certificate of the target entity. Each classical certificate is verified to find out the public key of the CA or the NCA of the next certificate. On the other hand, the nested certificates of the path are verified to validate their subject certificates. All of the certificates must be verified successfully, in order to consider the NPKI certificate path as valid and consequently to validate the public key of the target entity.

The NPKI certificate path verification algorithm is given in Figure 4.3. As can be seen from this algorithm, there are two verification methods for the certificates on a NPKI certificate path: (i) cryptographic verification method, (ii) subject certificate verification method. The former one is the classical approach to verify a public key signature, which is explained in Section 2.4.5 for classical certificates and in Section 3.4 for the nested certificates. The latter method is explained in Section 3.5. The type of the predecessor certificate determines the verification method of a certificate on a NPKI certificate path. The first certificate of a path has no predecessor and it is verified cryptographically using

the public key of its issuer. This public key must be known by the verifier. If the predecessor certificate is a classical certificate, then the current certificate (whatever its type is) is verified cryptographically using the public key of its issuer. This public key is obtained from the verification of the predecessor certificate. If the predecessor certificate is a nested certificate, then the current certificate (whatever its type is) is verified as the subject certificate of its predecessor. Therefore, the total number of subject certificate verifications is equal to the number of nested certificates on a NPKI certificate path. Consequently, the total number of cryptographic certificate verifications is equal to the number of classical certificates within the NPKI certificate path.

The algorithm given for the NPKI certificate path processing in Figure 4.3 explains the general principles of certificate processing conceptually. Actual implementation requires extra data structures, variables and predefined operations. For example, in order to check if an authority is trusted or not, the verifier checks whether the policy identifier within the certificate is in its allowed policy identifiers set or not. If so, the verifier trusts the authority, otherwise does not trust. Another predefined operation is necessary to check the validity of a signature over a certificate. All of the implementation details are defined for classical certificates in X.509 [2]. X.509 conformance issues of the nested certificates are discussed briefly in Section 4.8. Moreover, this algorithm is for a single path. In the NPKI, there may be several paths between any two entities. If one path fails to validate the target entity, then other paths should be tried.

Given: a NPKI certificate path, $c_1,c_2,c_3, \ldots ,c_{n-1},c_n$, with n certificates issued by trusted authorities $A_1,A_2,A_3, \ldots ,A_{n-1},A_n$, respectively and the legitimate public key of A_1, the issuer of c_1.

The verifier applies the following algorithm to verify the NPKI certificate path.

success := TRUE;
FOR *i*:=1 TO *n* DO
    IF *i*=1 OR c_{i-1} is a classical certificate THEN
        c_i is verified cryptographically using the public key of A_i;
        IF c_i is not valid THEN
            success := FALSE
    ELSE IF c_{i-1} is a nested certificate THEN
        c_i is verified as the subject certificate of c_{i-1} ;
        IF c_i is not valid THEN
            success := FALSE;
IF *success* THEN
    NPKI certificate path is valid
ELSE
    NPKI certificate path is invalid

Figure 4.3. NPKI certificate path verification algorithm

4.3.1. An Example on Certificate Path Verification

Figure 4.2 depicts an example of a NPKI certificate path which starts with the certificate authority *A* and ends with the target entity *T*. Other entities (*B, C, D, E, F, G and H*) are intermediate CAs and NCAs. The verifier initially knows only the correct public key of *A*. However, the verifier does not know the public keys of the other entities at the beginning. It is assumed that the verifier trusts *A, C, D, G, H* as CAs and trusts *B, E, F* as

NCAs. The aim of this path is to verify the certificate of *T* and to get the correct public key of it.

In that example, certificates 1, 2, 4, 5 and 8 are verified cryptographically, since they have either classical certificate predecessors or no predecessor. On the other hand, certificates 3, 6 and 7 are verified as subject certificates, since their predecessors are nested certificates. In that run, it is assumed that all of the certificates are valid. In other words, the signatures over the certificate contents are legitimate and the certificate contents are unmodified. If a certificate with an invalid signature or modified certificate content existed on the path, then the verifier would not be able to verify the public key of *T*. The step-by-step verification of this example NPKI certificate path is explained below.

1. First the verifier verifies certificate 1 cryptographically. Since the verifier trusts *A* as a CA then, it validates the certificate and finds out the public key of *B*.

2. Using the public key of *B*, the verifier verifies certificate 2 cryptographically. Since it trusts *B* as a NCA, the verifier finds out the correct signature and the hash over certificate 3 after this verification.

3. Certificate 3 is verified as the subject certificate of certificate 2. To do so, the verifier first computes the hash of the actual certificate 3. Then, the verifier compares this calculated hash with the one that it has found from the previous step. Next, the verifier compares the signature that it has found from the previous step with the actual signature over certificate 3. Since these comparisons yield equality and the verifier trusts *C* as a CA, the verifier verifies certificate 3 without using the public key of *C* and finds out the correct public key of *D*.

4. Then, the verifier simply verifies certificate 4 cryptographically using the public key of *D* that it has found from the previous step. Since the verifier trusts *D* as a CA, it gets the correct public key of *E*.

5. Using the public key of *E*, the verifier verifies certificate 5 cryptographically. The verifier trusts *E* as a NCA. Therefore, the verifier gets the correct signature and hash over the certificate 6.

6. The nested certificate 6 is verified as the subject certificate of certificate 5. To do so, the verifier first computes the hash of the actual certificate 6 content. Then, the verifier compares this calculated hash with the one that it has found from the previous step. Next, the verifier compares the signature that it has found from the previous step with the actual signature over certificate 6. Since these comparisons yield equality and the verifier trusts *F* as a NCA, the verifier verifies certificate 6 without using the public key of *F* and finds out the correct signature and hash over certificate 7.

7. The nested certificate 7 is verified as the subject certificate of certificate 6. To do so, the verifier first computes the hash of the actual certificate 7 content. Then, the verifier compares this calculated hash with the one that it has found out in the previous step. Next, the verifier compares the signature that it has found out in the previous step with the actual signature over certificate 7. Since these comparisons yield equality and the verifier trusts *G* as a CA, it verifies certificate 7 without using the public key of *G* and finds out the correct public key of *H*.

8. Finally, the verifier verifies certificate 8 cryptographically by using the public key of *H* that it had found out in the previous step and finds out the public key of the target entity, *T*, since it trusts *H* as a CA.

4.4. Structure of NPKI Certificate Paths

A NPKI certificate path is a sequence of nested certificate paths. The nested certificate paths were defined in Section 3.6. To recap, a nested certificate path is a sequence of nested certificates with a classical certificate at the end. In a nested certificate path, each nested certificate certifies the next certificate.

For example, consider the example certificate path in Figure 4.2. In this path, certificates 1, 4 and 8 are three separate 0-nested certificate paths. Certificates 2 and 3 constitute a 1-nested certificate path. Certificates 5, 6 and 7 form a 2-nested certificate path. Therefore, this example NPKI certificate path contains 6 sequential nested certificate paths. The last nested certificate path certifies the target end user of the NPKI certificate path. The intermediate nested certificate paths certify the first CA/NCA of the next nested certificate path. The public key of the first CA/NCA of the NPKI certificate path, which must be known by the verifier, is used to start verification process of the first nested certificate path. The public keys of the initial CAs/NCAs of the other nested certificate paths are found by the verification of the preceding nested certificate paths.

The correctness proof of the verification of a nested certificate path was given in Section 3.6.1. Since the verification of a NPKI certificate path is a sequence of nested certificate path verifications, it is straightforward to prove the legitimacy of the NPKI certificate path verification process.

4.5. Advantages and Disadvantages of NPKI

The advantages and the disadvantages of NPKI are the generalizations of the advantages and the disadvantages of the nested certificates that had been explained in Section 3.8.

An authority may prefer to issue a nested certificate instead of a classical one, since the assurance of a nested certificate is more restricted than a classical certificate. Moreover, a nested certificate is only for a single certificate and its verification leads to the verification of its subject certificate only. Therefore, the authorities of NPKI are more flexible in terms of certificate issuance. Such a flexibility in the certificate issuance allows extra certification relationships among the NPKI entities. Moreover, In NPKI, even if the public key of a CA is not known or revoked, the certificates issued by that CA can still be validated by using the nested certificates. In this way, the verifiers can combine disconnected classical certificate paths via nested certificates and form alternative certificate paths that cannot be constructed using only classical certificates. For example, the 1-nested certificate path, which contains certificate 2, in Figure 4.2 and the 2-nested certificate path, which contains the certificates 5 and 6, in Figure 4.2 connect three disconnected certificate paths.

As explained in Section 3.8.5, the verification of a certificate as a subject certificate is always computationally more efficient than the cryptographic verification of the same certificate. Similarly, NPKI certificate path verification is also more efficient than classical certificate path verification, since nested certificates are used in NPKI certificate paths. All of the certificates of a classical certificate path are verified using cryptographic certificate verification method. On the other hand, in the NPKI certificate paths, some of the certificates are verified using the subject certificate verification method. The total number of subject certificate verifications in a NPKI certificate path is the same as the total number of nested certificates on the path. Therefore, the relative efficiency improvement of the NPKI certificate path verification method over the classical certificate path verification method is directly and linearly related to the number of nested certificates on the NPKI certificate path. In a NPKI certificate path, there must be at least one classical certificate, which is for the target entity of the path. Moreover, there may be some other classical certificates as exemplified in Figure 4.2. In order to present the relative computational efficiency improvement of the NPKI certificate path verification method over the classical certificate path verification method, analytical and simulation studies are performed. The results obtained from these studies will be given in Section 5.

The nested certification overhead for the NCAs is the most important disadvantage of the NPKI, if the NPKI is formed by using the transition from an existing PKI method. In this method, in order to take the efficient verification advantage of the NPKI, several nested certificates must be issued. This overhead is especially significant for the upper level authorities of the infrastructure. Consequently, more storage space is necessary for them. However, the usage of nested certificates improves the execution time of the certificate path verification process. Therefore, NPKI is useful for the systems where the time complexity for the verifiers is a more important bottleneck than the time complexity for the NCAs. However, the balance between these time complexities can be adjusted by the amount of nested certificate usage in NPKI. Sections 4.6 and 4.7 explain the PKI to NPKI transition methods. The trade-off between these time complexities will be analyzed in Section 5.4.

4.6. Transition from PKI to NPKI

As discussed above, a NPKI can be constructed using the free certification method from zero. In such a NPKI, each CA/NCA is free to issue classical or nested certificates. Another approach is to construct a NPKI by converting from an existing PKI. In this method, there is a systematic nested certification enforcement. The sole aim of this method is to have fast certificate path verification. In order to construct NPKI from an existing PKI, there must be some rules about nested certificate issuance. In this section, the ways of transition from PKI to NPKI will be discussed. As in the free certification method, the CAs behave as NCAs here also. Moreover, the nested certificates are also node-to-arc arcs and hierarchical nested certification is still possible in this method.

The basic rule behind the ability of the nested certificate issuance in PKI to form a NPKI is briefly explained as follows. Let A be an authority and A^c be the set of authorities that have been certified by A. A can validate the certificates, which had been issued by the authorities in A^c, since A already knows the public keys of them. Consequently, A can issue nested certificates for all of the certificates (nested or classical) that had been issued by the

authorities in A^c. The above condition is necessary but not a sufficient condition. That means, for some other reason, A may not want to issue nested certificates for some certificates, which had been issued by the authorities in A^c.

The method, in which all of the authorities issue nested certificates for all of the cases described by the above rule, is called as *full nested certificate propagation*. An example application of the full nested certificate propagation method over a tree-shaped PKI is given by Figure 4.4. On the other hand, the method, in which some authorities prefer to issue nested certificates for some certificates, is called as *partial nested certificate propagation*. The way of nested certification in partial nested certificate propagation methods must also be well defined. A partial nested certificate propagation method is given later in this subsection.

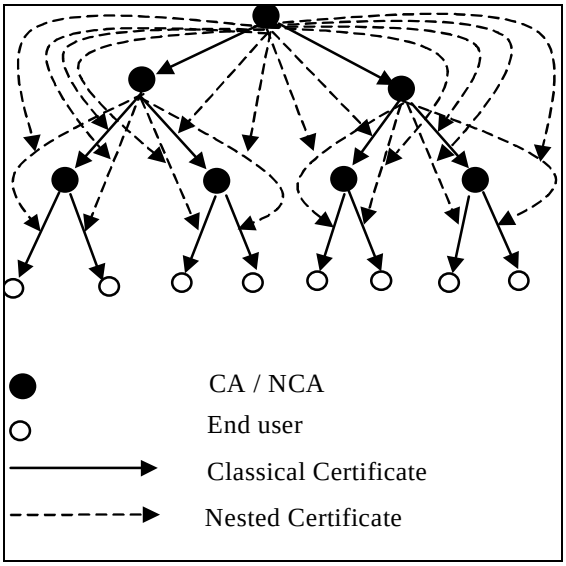

Figure 4.4. An example of the application of full nested certificate propagation method

By full nested certificate propagation method, it is possible to have a nested certificate path between each node pair for which there was a classical certificate path in the PKI before nested certificate issuance. This is actually the most important advantage of full nested certificate propagation method, since the method improves the average certificate path verification time significantly. However, a large volume of nested

certificate issuance is necessary here. This overhead is more significant for the top level authority. For example, as can be seen in Figure 4.4, the top-level authority has to issue 12 nested certificates. Moreover, in the same example, each second level authority has to issue two nested certificates. Therefore, it may not be practical to use this method for crowded hierarchies. Moreover, this method is an extreme case for nested certificate propagation in PKIs.

The certificate paths that can be extracted from a NPKI formed by the partial nested certificate propagation method are, actually, the NPKI certificate paths. The NPKI certificate paths, which have been explained in Section 4.3, contain both classical and nested certificates. The average improvement for the certificate path verification time in this case is less than the full nested certificate propagation method. Moreover, the number of nested certificates can be adjusted and kept to a reasonable level. A partial nested certificate propagation method, namely *nested certificate propagation towards end users* method, will be defined in the next subsection.

4.6.1. Nested Certificate Propagation Towards End Users

The full nested certificate propagation method creates one nested certificate path for each classical certificate path of the original PKI. However, it is not a common practice to verify the certificate of a CA/NCA by another CA/NCA. The common practice is to verify the certificate of an end user starting with a CA/NCA. Indeed, the X.509 standard enforces end user – CA distinction. The full nested certificate propagation method can be easily modified to have nested certificate paths only towards end users starting with any CA/NCA from which there exists a path in the original PKI. This is actually a kind of partial nested certificate propagation method. This new method is called as *nested certificate propagation towards end users*. In the nested certificate propagation towards end user method, each CA issues nested certificates as in the full nested certificate propagation method except that the CAs do not issue nested certificates for the classical certificates which belong to other CAs. However, the classical certificates, which belong to the end users, are certified. All of the nested certificates, if possible, are also certified. In

this way, a nested certificate path is produced for each classical certificate path from a CA to an end user. In this method, since no nested certificate is produced for the certificates towards CAs/NCAs, the total number of nested certificates is less than the full nested certificate propagation case. The analysis of the number of nested certificates to be produced in this method will be given in Section 5.4.

An example application of the nested certificate propagation towards end user method over a tree-shaped PKI is given by Figure 4.5. As can be seen in this figure, there is a nested certificate towards each end user from CAs/NCAs. However, as expected, there is no nested certificate path for the classical certificate paths between CAs/NCAs. Moreover, in this example case, the number of nested certificates is less than the example case of full nested certificate propagation method shown in Figure 4.4. For the same PKI, full nested certificate propagation requires 20 nested certificates, whereas nested certificate propagation towards end user method requires 16 nested certificates.

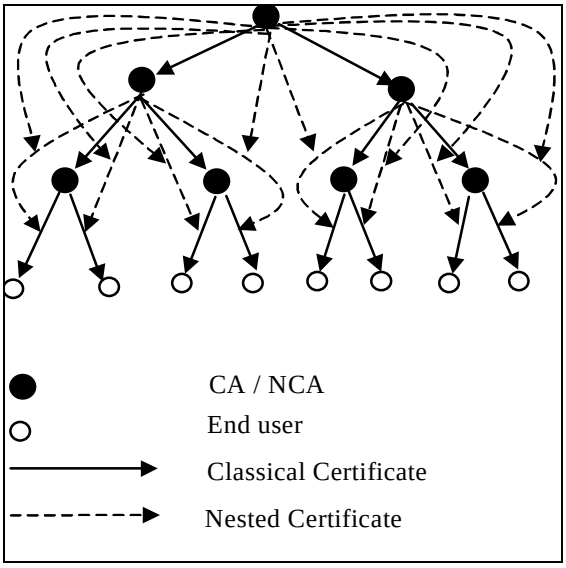

Figure 4.5. An example application of nested certificate propagation towards end user method

4.6.2. The Comments on Nested Certificate Propagation Methods

There are two important comments on the nested certificate propagation methods. One of them is about the availability requirements of the authorities and propagation delay. This is explained in this subsection. The other comment will be explained in Section 4.7.

Once an end user, say *E*, is issued a classical certificate by a CA, say *C*, some classical certificate paths towards *E* are automatically created depending on the topology of the PKI. In order to create certificate paths with nested certificates corresponding to these classical certificate paths, nested certificate propagation is necessary. This propagation may be full, partial or towards end user depending on the choice in the NPKI. At the first glance, the authorities seem to be available on-line during the propagation process. Although the authorities must issue nested certificates for propagation, they need not accomplish this task just after the classical certificate issuance for *E*. The propagation process may be completed in time. The nested certificate propagation is carried out only to form efficiently verifiable nested certificate paths towards *E*. However, the absence of some nested certificates on these paths does not cause non-verifiability of *E*, since there are compensating classical certificates in the PKI. Only the efficiency improvement of the nested certificate path verifications is not fully utilized temporarily for these cases.

Whatever the propagation method chosen, it is sufficient for the NCAs to periodically check their neighbor CAs/NCAs for new certificates to issue nested certificates for them. The length of this period is related to the time to the completeness of the nested certificate propagation for *E*. If long time periods are chosen by the NCAs for periodic nested certificate issuance, the completeness of the nested certificate propagation for *E* takes long time. However, even if the nested certificate propagation does not complete, each nested certificate issuance improves the average certificate path verification time.

Another important point about nested certificate propagation is that the propagation methods are sequential, not parallel. That means, the nested certificates must be issued starting with the authorities which are closer to *E*. Each authority should wait for the authorities which are closer to *E*. Because, each authority should certify the nested certificate, which has been issued by the authority which is closer to *E*. Therefore, the propagation delay is cumulative.

4.7. Nested Certificate Propagation versus Classical Certificate Propagation

Nested certificate propagation methods are described in Section 4.6. An alternative approach to nested certificate propagation is the following: Each CA, say CA_1, verifies all the paths in the PKI starting with CA_1. Having verified each path, CA_1 issues a classical certificate for the verified target entity. This method is theoretically possible in PGP [4,5] and ICE-TEL [34] systems. This is called *full classical certificate propagation*. In this way, it is possible to have just a single certificate between each node pair for which there was a classical certificate path in the PKI beforehand. Therefore, the certificate path verification is reduced to single certificate verification. This is, actually, an advantage of this method over nested certificate propagation. That means, the average certificate path (actually, single certificate) verification time for full classical certificate propagation method is less than average certificate path (actually, nested certificate path) verification time for full nested certificate propagation method. The full classical certificate propagation method can have a *partial* version in which some CAs can verify some paths of PKI to issue direct classical certificates for some other CAs.

The total number of nested certificates that must be issued in the full nested certificate propagation method is equal to the total number of extra classical certificates that must be issued in the full classical certificate propagation method. Moreover, the total numbers of verifications that must be performed by each CA of PKI for propagation are the same in both methods. Thus, it can be said that “why must one insist on nested certificate propagation method, since it has no advantage over classical certificate

propagation?”. Although the classical certificate propagation method is more advantageous, it is not possible to apply that method all the time. In the subsequent subsections, the motivations behind the nested certificate propagation will be discussed.

First, it is worthwhile to mention that the philosophy of nested certification does not discourage classical certificate issuance. On the contrary, the classical certification must be used together with nested certificates. If possible, classical certificate propagation must be performed. However, it is not always possible to do so. The nested certification is designed to close the gap of classical certification, especially classical certificate propagation.

There are mainly three motivations behind nested certificate propagation. These are the followings.

1. Trust information-free certificate issuance.
2. Preservation of trust relationships.
3. Appropriateness for distributed implementations.

Above three motivations will be examined in detail below.

4.7.1. Trust Information-Free Certificate Issuance

In order to verify a public key through a classical certificate path, the verifier must trust all of the intermediate CAs on the path. If even one of those CAs is not trusted, then the verifier cannot verify the path and consequently, the public key of the target of the path. Therefore, in order to realize full classical certificate propagation, every CA of the PKI must trust everyone that can be reached starting with itself. Certainly, this is not a realistic assumption. On the other hand, partial classical certificate propagations are possible for the CAs, for which it is possible to verify some classical certificate paths that contain trusted CAs. Indeed, this should be done in a PKI. That means, if it is possible to

verify the public key of a target entity for CA_1, then CA_1 should issue a classical certificate for that target entity. The necessary condition for this issuance is that CA_1 must trust all of the intermediate CAs on the certificate path towards the target entity. On the other hand, if there is at least one untrusted (or the trust information is unknown) CA on the path, then CA_1 will not be able to verify the path and will not be able to issue a classical certificate for the target entity.

On the other hand, in order to issue a nested certificate for a subject certificate, the NCA does not need to trust anyone. Therefore, nested certificate propagation is possible all the time, especially for the cases where classical certificate propagation is not possible because of trust reasons. In order to issue a nested certificate, the NCA should only know the correct public key of the subject certificate issuer and should verify the signature over the subject certificate. Moreover, as described in Section 4.6, it can be easily assumed that the NCAs know the public keys of the subject certificate issuers to verify subject certificates in full and partial nested certificate propagation methods. The NCA does not need to trust the subject certificate issuer or the entity certified within the subject classical certificate. Because, by definition, a nested certificate does not guarantee the information correctness of the subject certificate content. Having verified a nested certificate and its subject certificate via that nested certificate, the verifier must also trust the subject certificate issuer, in order to completely verify the subject certificate.

4.7.2. Preservation of Trust Relationships

Classical certificate propagation methods spoils the existing trust relationships in the PKI, since a CA forces the verifiers to by-pass a certificate path by issuing a direct classical certificate for the target entity. Although this situation does not seem to be a problem at the first glance, it may cause a problem for the strictly hierarchical PKIs, like the PKIs of DNS security extensions [64], PEM [46] and SET [47] systems. In such strictly hierarchical PKIs , it is not possible to by-pass intermediate CA levels and consequently, classical certificate propagation is not possible for these PKIs.

For strictly hierarchical PKIs, the trust relationships must be preserved such that the authorities of the PKI paths, which the verifiers trust, must be kept the same after the propagation. Nested certificate propagation methods preserve trust relationships in this manner as is described below.

As described earlier, the verifiers should also trust the subject certificate issuers in order to verify a subject certificate via a nested certificate. Moreover, each authority deals with its neighbor in the propagation phase. Therefore, the nested certificate propagation methods do not spoil the trust relationships for the verifiers. In this way, trust relationships within the PKI are preserved in the NPKI, which is constructed via nested certificate propagation. This discussion will be generalized to the full and partial nested certificate propagation cases.

By the full nested certificate propagation method, one nested certificate path is created for each classical certificate path in the PKI. Moreover, a nested certificate path passes through the CAs that its corresponding classical certificate path passes. As a consequence, in the nested certificate path verification process, the verifier must trust exactly the same CAs of the corresponding classical certificate path. In this way, the existing trust relationships in the PKI are preserved in the NPKI. For example, consider Figure 4.6. In this figure, both a classical certificate path, which is the certificate sequence cc_6, cc_5, cc_4, cc_3, cc_2, cc_1, cc_0, and its corresponding nested certificate path, which is the certificate sequence nc_6, nc_5, nc_4, nc_3, nc_2, nc_1, cc_0, are shown together. The verifier has to trust the authorities A_1, A_2, A_3, A_4, A_5 and A_6 in order to verify both classical and nested certificate paths. Moreover, in both classical and nested certificate path verifications, the verifier must know the public key of A_6 a priori. That means, the prerequisites of both path verifications are the same. On the other hand, since a large amount of subject certificate verifications are performed, the average path verification time in NPKI is significantly less than the average classical certificate path verification time in PKI.

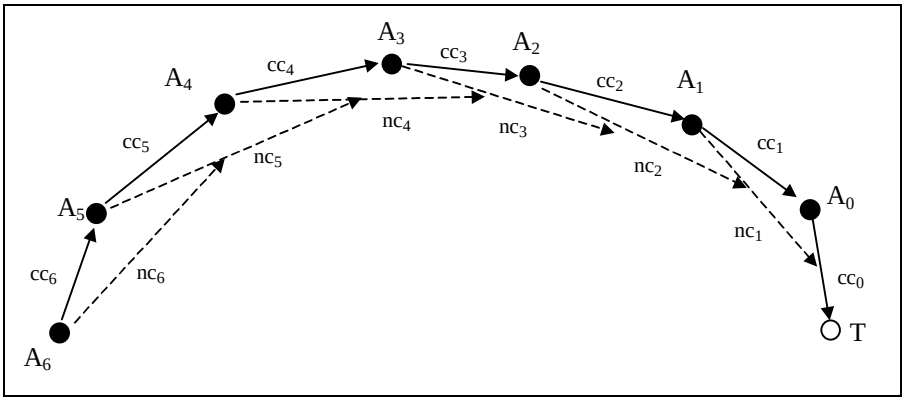

Figure 4.6. An example classical certificate path and its corresponding nested certificate path produced by full nested certificate propagation

The NPKI certificate paths formed by the partial nested certificate propagation methods have both classical and nested certificates as the intermediate certificates. Therefore, the verification times of them will not be as small as the nested certificate paths. Nevertheless, the average NPKI certificate path verification time in this case is better than the classical certificate paths. Moreover, the NPKI certificate paths formed by a partial nested certificate propagation method also pass through the exactly the same CAs that the corresponding classical certificate paths pass. Therefore, the trust relationships in PKI are also preserved here. For example, in Figure 4.7, both a classical certificate path (the certificate sequence cc_6, cc_5, cc_4, cc_3, cc_2, cc_1, cc_0) and a corresponding NPKI certificate path produced by partial nested certificate propagation (the certificate sequence nc_6, nc_5, cc_4, cc_3, nc_2, cc_1, cc_0) are shown together. As can be seen from this figure, the verifier must know the public key of A_6 and must trust all of the authorities, A_1, A_2, A_3, A_4, A_5 and A_6, for the verification of both paths. In other words, the trust relationships in the classical certificate path of the PKI are also preserved in NPKI certificate path produced via partial nested certificate propagation.

To summarize, it can be said that nested certificate propagation methods preserve the existing trust relationships in the PKI. In this way, the trust structure of the PKI is not spoiled.

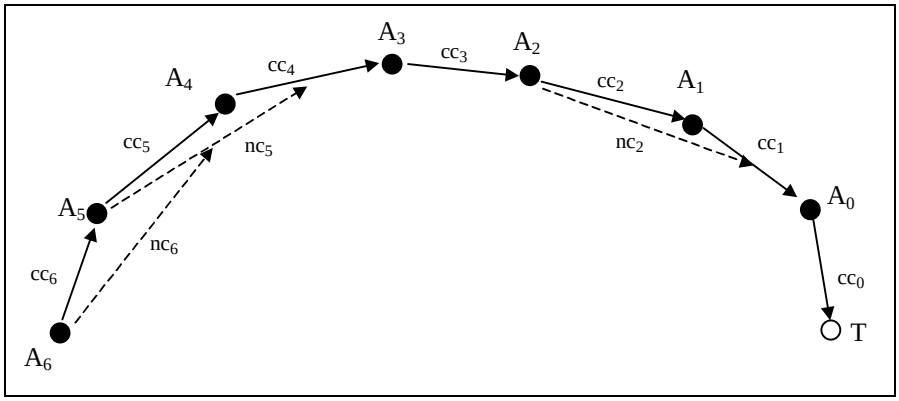

Figure 4.7. An example classical certificate path and its corresponding nested certificate path produced by partial nested certificate propagation

4.7.3. Suitability for Distributed Implementations

One should verify the paths starting from itself in classical certificate propagation methods. Such an approach is centralized. At least, classical certificate propagation methods require enumerating all of the paths starting from a node in the global PKI. However, in nested certificate propagation method, everyone is responsible to issue nested certificates to the certificates that its certified nodes have issued. This approach is more suitable to distributed applications. Moreover, in nested certificate propagation method, one need not enumerate paths to issue nested certificates. It is sufficient for an authority to check its certified authorities regularly for new certificates and issue nested certificates for these new certificates.

4.8. X.509 Conformance

One of the basic design goals of the NPKI is that NPKI is X.509 compliant. However, the X.509 standard [2] has been designed for classical certificates, not for nested certificates. Therefore, the X.509 standard should be modified to accommodate the nested certificates as well as the classical certificates. In this section, proposals for these modifications will be discussed.

4.8.1. Different Attributes for the Certified Entity

The most important difference between a classical certificate and a nested certificate is the difference in content. A classical certificate certifies the binding between the name and the public key of an entity. Therefore, it contains a name and a public key as the attributes of the certified entity. On the other hand, a nested certificate certifies the binding between a certificate and its claimed signature. Thus, it contains a certificate identifier and a hash and signature combination as the attributes of the certified entity. Therefore, there are two subproblems regarding the representation of these two attributes of the nested certificates in X.509. These are *the problem of the representation of the claimed hash and signature* and *the problem of the representation of the subject certificate identity* in X.509. Solutions to these problems are sketched below.

Representation of the claimed hash and signature: In classical X.509 [2] certificates, the public key of the certified entity is stored as a bitstring. In nested certificates the hash and the signature combination can also be represented as bitstring. Therefore, there is no structural problem to represent the claimed hash and signature of a nested certificate in X.509. The only difference is in the name and the interpretation of the certified object field. For classical certificates, the certified object is a public key, whereas in nested certificates, the certified object is a hash and signature combination. This problem can be solved by renaming the **subjectPublicKeyInfo** field of a classical X.509 certificate as **certifiedObject** and interpreting that field differently for classical and nested certificates. Once the type of the certificate (classical or nested) is identified, the **certifiedObject** field will be able to be interpreted either as the public key of the certified entity (for classical certificates) or as the claimed hash and signature combination of the subject certificate (for nested certificates).

Representation of the subject certificate identity: The **Name** structure of the classical X.509 certificates cannot be used to represent the identity of the subject certificate in the nested certificates. A certificate identity is of type integer. In the nested certificates, the identity of the subject certificate can be represented in the **otherName** choice of the

Subject Alternative Name (**subjectAltName**) extension field of the X.509 structure. For the nested certificates, instead of the **name** field, the **otherName** choice of the **subjectAltName** extension will be processed to extract the identity of the subject certificate that had been certified by the nested certificate.

4.8.2. The Problem of Differentiation in the Certificate Types

In order to use the solutions described in Section 4.8.1, the certificate type must be identified a priori. In this subsection, three alternative solutions are proposed to identify whether a certificate is a classical certificate or a nested certificate.

Alternative 1: There is no certificate type concept in the X.509 standard. Different types of certificates, like user certificates, CA certificates and cross certificates, are stored with different attributes in the directories. The nested certificates can be stored with a new attribute also. In this way, the certificate processor can understand whether a certificate is a nested certificate or not.

Alternative 2: However, for some applications those directories are not used. In order to differentiate a nested certificate from a classical certificate in case of absence of those directories, a certificate type field might be added to the certificate structure. Depending on the value of that field, the certificate can be interpreted either as a classical or a nested certificate.

Alternative 3: The second alternative above requires a change in the structure of the X.509 certificates, therefore that solution is costly. Moreover, the first one is not useful for all of the applications. The third alternative uses the *Extended Key Usage* (**extKeyUsage**) extension field of the classical X.509 certificate structure. That field is used to assign an extra function to the certified public key of a classical certificate via object identifiers. That field can also be used to qualify a certificate as a nested certificate. A publicly known

object identifier can be assigned as the value for the **extKeyUsage** field for the nested certificates and the certificate processor checks the value of that field to understand whether a certificate is a nested certificate or not. Moreover, the value of that field is assigned at the issuance time of a nested certificate. Therefore, this solution does not require a design modification for the X.509 certificate structure, so it is the least costly solution for the problem of differentiation in the certificate types.

4.8.3. CA – NCA Differentiation and Trust Considerations

In the previous sections, it is assumed that a single entity can behave both as a CA and as a NCA. The only difference is in their trustworthiness. Since the trust information is kept as policy identifiers in X.509, there must be different policy identifiers for the NCAs. This is neither a design nor an implementation problem, because the policy identifiers are assigned at the issuance time. However, the certificate structure should allow checking the trustworthiness of the CAs and the NCAs differently. Therefore, the CAs for the classical certificates must be interpreted as the NCAs for the nested certificates. Once the type of the certificate is identified by using one of the methods described in Section 4.8.2, this interpretation problem becomes an issue of implementation. Therefore, there is no need to change the fields related to CAs in the X.509 certificate structure.

5. PERFORMANCE EVALUATION

The main advantage of the nested certification scheme is the computational efficiency of the subject certificate verification method as compared to the cryptographic certificate verification method. Consequently, nested certificate path and NPKI certificate path verification methods are more efficient than the classical certificate path verification method. Moreover, a significant amount of nested certificates must be issued to convert a PKI into NPKI and there is a trade-off between the number of nested certificates and efficiency improvement in certificate path verification. In this section, the analytical and simulation studies will be detailed to support the claimed performance characteristics. The performance measure that is used in these analyses is the *speed-up* factor. The speed-up factor is the ratio for the verification improvement of using nested certificates over the classical techniques.

In Section 5.1.1, analytical formulation of the speed-up factor for the subject certificate verification method is derived and it is proven that the subject certificate verification method is always faster than the cryptographic certificate verification method. The graphical analysis of the subject certificate verification speed-up factor is given in Section 5.1.2. The formulation of the speed-up factor for the NPKI certificate path verification method is derived in Section 5.1.3. It is proven that the NPKI certificate path verification method performs faster than the classical certificate paths of the same length and characteristics. The analysis of the NPKI certificate path verification speed-up factor is given in Section 5.1.4. Analytical comparison of the subject certificate verification and the NPKI certificate path verification speed-up factors is given in Section 5.1.5. It is concluded that the NPKI certificate path verification speed-up factor is always less than the corresponding subject certificate verification speed-up factor, because of the classical certificates in the NPKI certificate paths. The NPKI certificate path verification speed-up factor approaches to the subject certificate verification speed-up factor, as the proportion of number of nested certificates increases. The nested certificate path verification speed-up factor is analyzed in Section 5.1.6. It is observed that the nested certificate path verification

speed-up factor increases, as the number of nested certificates on the nested certificate path increases.

Simulation results regarding subject certificate verification, NPKI certificate path verification and nested certificate path verification are given in Sections 5.2.1, 5.2.2 and 5.2.3 respectively. The characteristics of the simulation results are similar to the analytical results except that the simulation results are less than the analytical results, as discussed in Section 5.3.

Nested certification overhead, which is the number of nested certificates to be issued by the authorities in NPKI, is analyzed for a tree shaped topology in Section 5.4. The trade-off between the nested certification overhead and the certificate path verification efficiency is exemplified for an example case in the same section. It is concluded that, although the nested certificate propagation improves the NPKI certificate path verification significantly, it causes an overhead especially for the upper level authorities in NPKI. However, this overhead is tolerable to have efficiently verifiable certificate paths.

5.1. Analytical Performance Evaluation

The relative performance improvement of the subject certificate verification method over cryptographic certificate verification method and the improvement of nested and NPKI certificate path verification methods over the classical certificate path verification method are formulated and their graphical analyses are given in this subsection..

5.1.1. Formulation of Subject Certificate Verification Performance

Subject certificate verification method, which is explained in Section 3.5, is more time-efficient than the cryptographic certificate verification method, which is explained in

Sections 2.4.5 and 3.4. Cryptographic certificate verification method employs both public key cryptosystem operation and hash calculation. On the other hand, the subject certificate verification method does not employ any public key cryptosystem operation. Only hash computation and comparisons are enough to verify a subject certificate via a nested certificate. Hash calculation is also a part of the cryptographic certificate verification method. Therefore, the time spent for the public key cryptosystem operation is the saving of the subject certificate verification method.

In this subsection, the relative computational speed-up factor for the subject certificate verification time over the public key cryptosystem based certificate verification time is formulated. The speed-up factor indicates how many times the subject certificate verification method is faster than the cryptographic certificate verification method. Moreover, it will be proven that the subject certificate verification is always computationally more efficient than the cryptographic certificate verification. In order to formulate the computational speed-up factor, the total cryptographic certificate verification time and the total subject certificate verification time are also formulated.

As explained in Sections 2.4.5 and 3.4, the first step of the cryptographic verification of a certificate is the hash computation of the certificate content. Then, the public key cryptosystem based signature verification operation is applied to the signature part of the certificate. In this way, the verifier finds out the correct hash that the CA had signed. Finally, the computed hash is compared with the verified hash for equality. $T_{pkc}(h, c, cert)$ is the total time of the cryptographic verification of the certificate *cert* which uses the hash algorithm *h* and the public key cryptosystem *c*. Neglecting the time for the comparison, $T_{pkc}(h, c, cert)$ is formulated as:

$$T_{pkc}(h, c, cert) = t_h + \frac{S(cert)}{\lambda_h} + t_c \tag{5.1}$$

where, $S(m)$ is the *size-of* function and returns the bit length of its argument *m*. *h* is the hash algorithm used, like the MD5 [15] or SHA-1 [16] algorithms. t_h is the fixed setup time for the hash algorithm *h* in milliseconds (msec). λ_h is the throughput of the hash algorithm *h* in bits/msec. *c* is the public key cryptosystem used, like the RSA [12] or DSA [32] cryptosystems. t_c is the time necessary for the verification in the public key cryptosystem *c* in msec. The *cert* is the certificate content, which is to be verified.

The size of the signature over the hash *cert* does not depend on the size of *cert* and the hash algorithm used. It solely depends on the cryptosystem used for signing. For example, in the RSA cryptosystem, the signature size is the same as the modulus size of the signing key, for the DSA cryptosystem, the signature has two 160-bits long parts. That is why t_c does not depend on $S(cert)$ and the hash size.

The time necessary to compute the hash of a certificate content has two parts: (i) fixed setup time, t_h, which is for the initializations; (ii) hash calculation time, which depends on the size of the certificate content and calculated as $\frac{S(cert)}{\lambda_h}$.

On the other hand, the subject certificates are verified by one hash computation and two comparisons as described in Section 3.5. $T_{s(z,\,scert)}$ is the total time of the verification of the certificate s*cert* as a subject certificate using the hash algorithm *z*. Neglecting the time for the comparisons, $T_{s(z,\,scert)}$ is formulated as:

$$T_{s(z,\,scert)} = t_z + \frac{S(scert)}{\lambda_z} \tag{5.2}$$

where, $S(m)$is the *size-of* function and returns the bit length of its argument *m*. *z* is the hash algorithm used, like the MD5 or SHA-1 algorithms. t_z is the fixed setup time for the hash algorithm *z* in msec. λ_z is the throughput of the hash algorithm *z* in bits/msec. The *scert* is the subject certificate content that is to be verified.

The relative improvement of the subject certificate verification over the cryptographic certificate verification must be analyzed for the same certificates, that means, *cert = scert*. Further assume that, the same hash algorithms are employed for both type of verifications, that means, *h = z*. $f_{(h,c,cert)}$ is the relative computational speed-up factor for the verification of the certificate *cert* as a subject certificate over the cryptographic verification of it. *h* is the hash algorithm and *c* is the public key cryptosystem used in the verifications. $f_{(h,c,cert)}$ is formulated as:

$$f_{(h,c,cert)} = \frac{T_{pkc}(h,c,cert)}{T_s(h,cert)} = \frac{t_h + \frac{S(cert)}{\lambda_h} + t_c}{t_h + \frac{S(cert)}{\lambda_h}} = 1 + \frac{t_c}{t_h + \frac{S(cert)}{\lambda_h}} = 1 + \frac{1}{\frac{t_h}{t_c} + \frac{S(cert)}{\lambda_h \cdot t_c}} \quad (5.3)$$

The values of t_h, t_c, λ_h and $S(cert)$ are all positive. Therefore, as Equation 5.3 implies, the relative speed-up factor $f_{(h,c,cert)}$ is always larger than 1. This conclusion proves the following theorem.

Theorem 2: Verification of a certificate as a subject certificate is always computationally more efficient than the verification of the same certificate by employing a signature verification scheme based on a public key cryptosystem, assuming that the same hash algorithms are used in both cases.

5.1.2. Graphical Analysis of Subject Certificate Verification Speed-up Factor

The hash computation is a more computationally efficient process than the cryptographic verification of a signature. Bosselaers et. al. [80,81] reported that λ_{MD5} = 136 Kbits/msec, λ_{SHA-1} = 55 Kbits/msec on a 90 Mhz Pentium computer. On the other hand, Wiener [84] has reported that $t_{RSA1024}$ = 0.6 msec, $t_{DSA1024}$ = 27 msec on a 200 Mhz Pentium computer. However, such performance values for hash functions and public key cryptosystem operations are not obtained on the same platform. For our analyses, we have measured the execution times of the MD5 [15] and SHA-1 [16] hash functions and the RSA [12] and DSA [32] cryptosystems with different key sizes. The measurements are obtained by running on a 166 Mhz Pentium computer. The cryptographic library of the SECUDE toolkit [86] was used. Table 5.1 gives these measurements.

Table 5.1. Execution times for public key cryptosystem verifications and hash calculations

Algorithm	$t_{Algorithm}$(msec)	$\lambda_{Algorithm}$(bits/msec)
DSA512	33.909	-
DSA1024	113.968	-
RSA512	1.256	-
RSA1024	4.457	-
RSA2048	17.152	-
SHA-1	0.0093	36057.0
MD5	0.009	68267.0

The behavior of the speed-up factor, $f_{(h,\,c,\,cert)}$, versus the certificate size, $S(cert)$, for different cryptosystem and hash algorithm combinations is given by the graphs in Figure 5.1 and Figure 5.2. These graphs are drawn by using the formula given in Equation 5.3 and the t_c, t_h, λ_h values of Table 5.1.

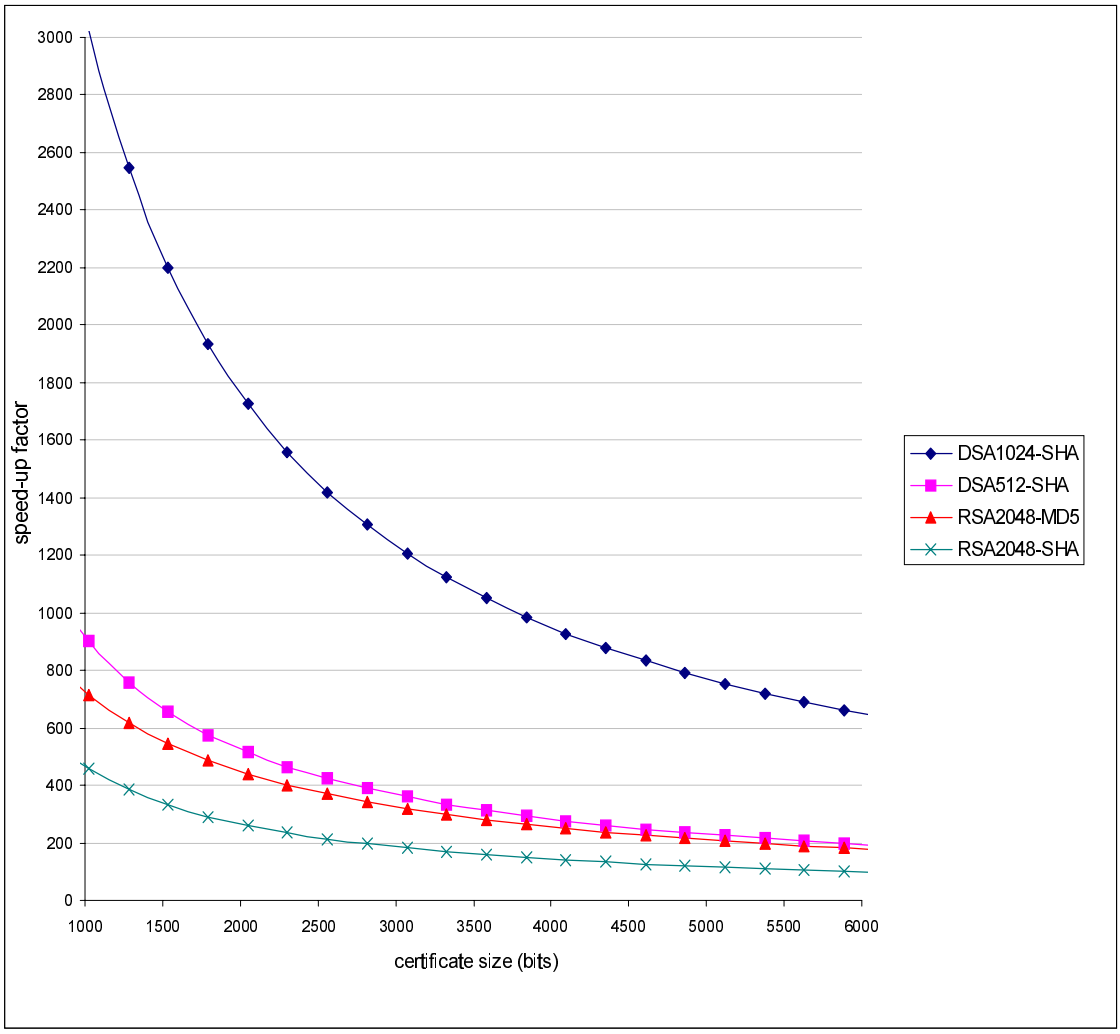

Figure 5.1. Change of speed-up factor with respect to certificate size for different algorithm combinations

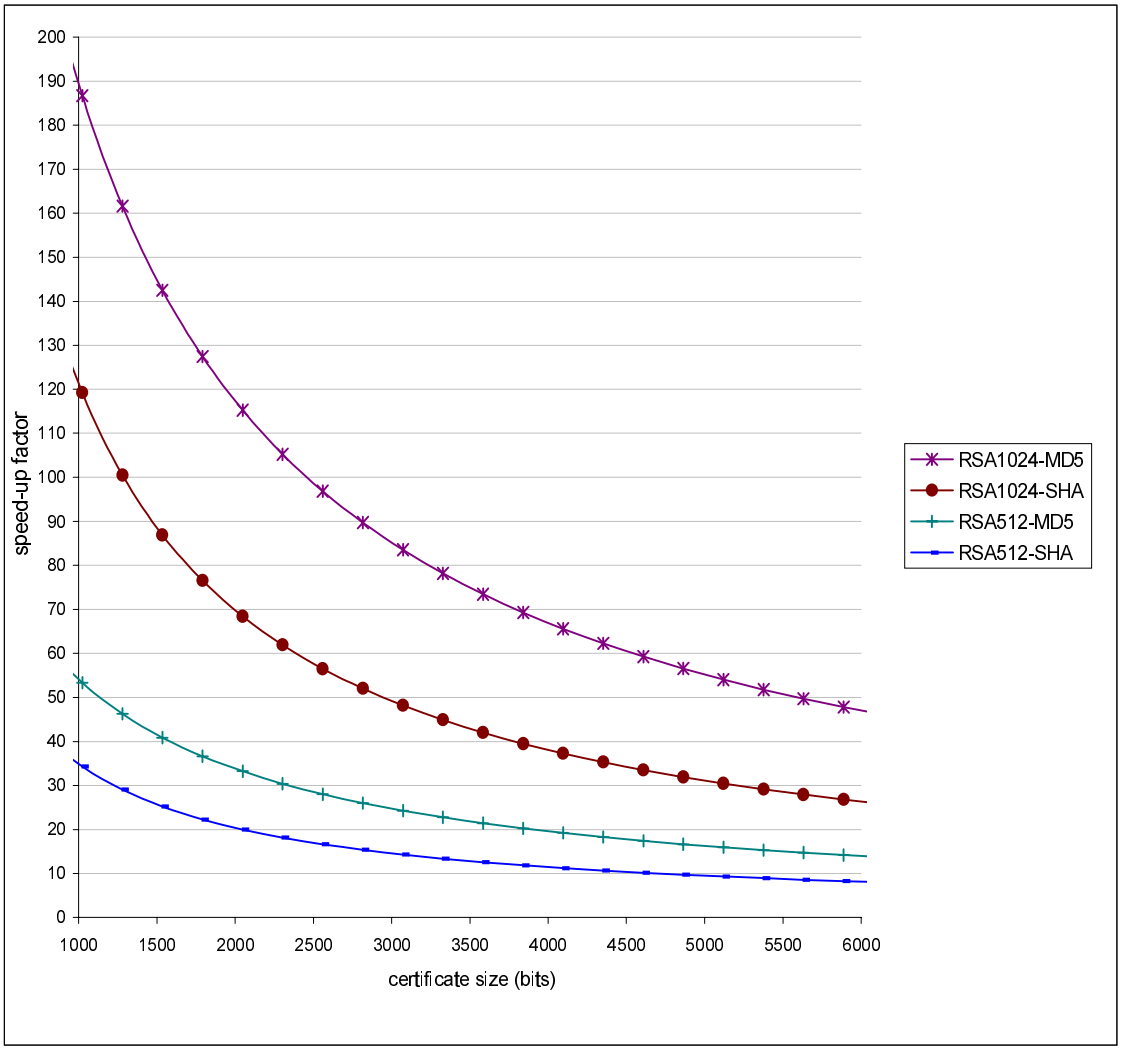

Figure 5.2. Change of speed-up factor with respect to certificate size for different algorithm combinations

The speed-up factor in Equation 5.3, $f_{(h,c,cert)}$, is directly related to cryptographic verification time, t_c. The speed-up factor is inversely related to the certificate size, $S(cert)$. Those conclusions can be seen from Figure 5.1 and Figure 5.2. The speed-up factor varies between 8 and 3000 for the public key cryptosystem - hash function pairs considered. The speed-up factor becomes larger for the slower cryptosystems like the DSA512, DSA1024 and RSA2048, as shown in Figure 5.1. Moreover, the speed-up factor decreases while the certificate size increases. However, there is a remarkable improvement even for the large certificate sizes. For example, where $S(cert)$ = 6000 bits, then the speed-up factor varies between 8 and 640, in other words, the subject certificate

verification is 8 to 640 times faster than the cryptographic verification. On the other hand, the typical certificate sizes are 3000 to 4500 bits. The speed-up factors for the typical certificate sizes are between 10 and 1220, depending on the algorithms used.

The speed-up factor for the RSA512 cryptosystem is the one with the smallest values as shown in Figure 5.2. However, the average speed-up factor for RSA512 is 20. That means, the subject certificate verification method is 20 times faster than the RSA512 based cryptographic certificate verification. Although this factor is smaller than other cryptosystems, it is still a good improvement.

The hash algorithms also effect the speed-up factor. However, since the hash algorithms are much faster than the public key cryptographic operations, the effect of the hash computation time over the speed-up factor is not as significant as the execution time of the public key cryptosystem operations.

As can be seen from Figure 5.1 and Figure 5.2, the effect of the MD5 hash algorithm over the speed-up factor is better than the effect of the SHA-1 hash algorithm for the same cryptosystems. The throughput of the MD5 algorithm is greater than the one of the SHA-1 algorithm and the setup times are almost the same in both cases, as can be seen from Table 5.1. The effect of the MD5 algorithm over the speed-up factor is better, since the speed-up factor given by Equation 5.3 is directly related to the hash throughput.

5.1.3. Formulation of NPKI Certificate Path Verification Performance

Verification of a certificate as a subject certificate is more efficient than the cryptographic verification of the same certificate as given by Theorem 2 of Section 5.1.1. In this subsection, the relative speed-up factor for the NPKI certificate path verification time over the classical certificate path verification time is formulated. The speed-up factor gives an idea about how many times the NPKI certificate path verification method is faster

than the classical certificate path verification. Moreover, it will be proven that the NPKI certificate path verification is always computationally more efficient than the classical certificate path verification. In order to formulate the speed-up factor, the NPKI and classical certificate path verification times are also formulated.

Let *N-path* be a NPKI certificate path. In *N-path*, all of the certificates (nested or classical) must be verified either cryptographically or as subject certificates. In the *N-path* verification time formulation, three certificate sets will be used. These sets are $SubjectCerts_{N\text{-}path}$, $CryptCerts_{N\text{-}path}$ and $AllCerts_{N\text{-}path}$. The definitions of them are as follows:

$$SubjectCerts_{N\text{-}path} = \left\{ cert_i \,\middle|\, i \text{ is the index of the certificates to be verified as subject certificates} \right\} \quad (5.4)$$

$$CryptCerts_{N\text{-}path} = \left\{ cert_i \,\middle|\, i \text{ is the index of the certificates to be verified cryptographically} \right\} \quad (5.5)$$

$$AllCerts_{N\text{-}path} = SubjectCerts \bigcup CryptCerts \quad (5.6)$$

In the *N-path*, the certificates with nested certificate predecessors are verified as subject certificates. Therefore, the total number of subject certificate verifications is the total number of nested certificates. Consequently, the total number of cryptographic verifications is the total number of classical certificates of the *N-path*. Thus, the numbers of elements of the above declared sets are given as:

$$\left| SubjectCerts_{N\text{-}path} \right| = n_{nc}^{N-path} \quad (5.7)$$

$$\left| CryptCerts_{N\text{-}path} \right| = n_{cc}^{N-path} \quad (5.8)$$

$$\left|AllCerts_{N\text{-}path}\right| = n_{total}^{N-path} \tag{5.9}$$

where n_{cc}^{N-path} is the total number of classical certificates, n_{nc}^{N-path} is the total number of nested certificates of the *N-path*. $n_{total}^{N-path} = n_{nc}^{N-path} + n_{cc}^{N-path}$, i.e. the total number of certificates of the *N-path*.

$T_{N-path}(h,c)$ is the total *N-path* verification time. For the sake of the simplicity of the analysis, it is assumed that the same hash algorithm *h* and the same public key cryptosystem *c* are used for all of the certificates of the *N-path*. Under these assumptions, $T_{N-path}(h,c)$ is formulated as:

$$T_{N-path}(h,c) = \sum_{cert_p \in CryptCerts_{N\text{-}path}} T_{pkc}(h,c,cert_p) + \sum_{cert_t \in SubjectCerts_{N\text{-}path}} T_s(h,cert_t) \tag{5.10}$$

$T_{pkc}(h,c,cert_p)$ and $T_s(h,cert_t)$ are the times for the verification of a certificate using a public key cryptosystem and the verification of a certificate as a subject certificate respectively. Substituting Equations 5.1 and 5.2 of Section 5.1.1 for $T_{pkc}(h,c,cert_p)$ and $T_s(h,cert_t)$, the total *N-path* verification time, $T_{N-path}(h,c)$, becomes:

$$T_{N-path}(h,c) = \sum_{cert_p \in CryptCerts_{\text{N-path}}} \left(t_h + \frac{S(cert_p)}{\lambda_h} + t_c \right) + \sum_{cert_t \in SubjectCerts_{N\text{-}path}} \left(t_h + \frac{S(cert_t)}{\lambda_h} \right)$$

$$= \sum_{cert_p \in CryptCerts_{\text{N-path}}} t_c + \sum_{cert_p \in CryptCerts_{\text{N-path}}} \left(t_h + \frac{S(cert_p)}{\lambda_h} \right) + \sum_{cert_t \in SubjectCerts_{N\text{-}path}} \left(t_h + \frac{S(cert_t)}{\lambda_h} \right)$$

$$= \sum_{cert_p \in CryptCerts_{\text{N-path}}} t_c + \sum_{cert_i \in AllCerts_{\text{N-path}}} \left(t_h + \frac{S(cert_i)}{\lambda_h} \right)$$

$$= \sum_{cert_p \in CryptCerts_{\text{N-path}}} t_c + \sum_{cert_i \in AllCerts_{\text{N-path}}} t_h + \frac{n_{total}^{N-path}}{\lambda_h} \cdot \frac{\sum_{cert_i \in AllCerts_{\text{N-path}}} S(cert_i)}{n_{total}^{N-path}} \quad (5.11)$$

It has been assumed that the same hash algorithm *h* and the same cryptosystem *c* are used for all of the certificates of the *N-path*. Therefore, it can be deduced that:

$$\sum_{cert_p \in CryptCerts_{N-path}} t_c = n_{cc}^{N-path} \cdot t_c \quad (5.12)$$

$$\sum_{cert_i \in AllCerts_{\text{N-path}}} t_h = n_{total}^{N-path} \cdot t_h \quad (5.13)$$

Furthermore, $\frac{\sum_{cert_i \in AllCerts_{\text{N-path}}} S(cert_i)}{n_{total}^{N-path}}$ is the average size of all of the certificates of the *N-path* and this average value is denoted as $S\left(cert_{avg}^{N-path}\right)$. Under these considerations, Equation 5.11 becomes:

$$T_{N-path}(h, c) = n_{cc}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h} \quad (5.14)$$

In order to examine the speed-up factor of the NPKI certificate path verification over the classical certificate path verification, the classical certificate path verification time should also be formulated. In a classical certificate path, all of the certificates are classical

certificates that must be verified cryptographically. Therefore, a classical certificate path can be considered as a special case of a NPKI certificate path where there are no nested certificates. That is, for a classical certificate path, *C-path*:

$$SubjectCerts_{C-path} = \{\ \} \tag{5.15}$$

$$CryptCerts_{C-path} = AllCerts_{C-path}\,. \tag{5.16}$$

From Equations 5.15 and 5.16, it can be deduced that:

$$n_{nc}^{C-path} = 0 \tag{5.17}$$

$$n_{cc}^{C-path} = n_{total}^{C-path} \tag{5.18}$$

Under these considerations, the verification time of *C-path*, $T_{C-path}(h,c)$, is given by Equation 5.19.

$$T_{C-path}(h,c) = n_{total}^{C-path} \cdot t_c + n_{total}^{C-path} \cdot t_h + n_{total}^{C-path} \cdot \frac{S\left(cert_{avg}^{C-path}\right)}{\lambda_h} \tag{5.19}$$

$f_{path}(h, c, N\text{ - }path)$ is the speed-up factor for the verification of *N-path* over the verification of *C-path* in terms of the time spent for the verifications. $f_{path}(h, c, N\text{ - }path)$ must be analyzed where the path lengths and the average certificate sizes of the *N-path* and *C-path* are the same. Moreover, it is also assumed that the same hash algorithm *h* and the same cryptosystem *c* are used in both *N-path* and *C-path*. These assumptions imply that:

$$n_{total}^{C-path} = n_{total}^{N-path} \tag{5.20}$$

$$S\left(cert_{avg}^{N-path}\right) = S\left(cert_{avg}^{C-path}\right) \tag{5.21}$$

Under these considerations, $f_{path}(h, c, N\text{ - }path)$ is given by Equation 5.22. The derivation is as follows:

$$f_{path}(h, c, N\text{ - }path) = \frac{T_{C-path}(h, c)}{T_{N-path}(h, c)}$$

$$= \frac{n_{total}^{C-path} \cdot t_c + n_{total}^{C-path} \cdot t_h + n_{total}^{C-path} \cdot \dfrac{S\left(cert_{avg}^{C-path}\right)}{\lambda_h}}{n_{cc}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}}$$

$$= \frac{n_{total}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}}{n_{cc}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}}$$

$$= \frac{n_{total}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}}{\left(n_{total}^{N-path} - n_{nc}^{N-path}\right) \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}}$$

$$= \frac{n_{total}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}}{n_{total}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h} - n_{nc}^{N-path} \cdot t_c}$$

$$= \frac{1}{1 - \frac{n_{nc}^{N-path} \cdot t_c}{n_{total}^{N-path} \cdot t_c + n_{total}^{N-path} \cdot t_h + n_{total}^{N-path} \cdot \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}}}$$

$$= \frac{1}{1 - \frac{n_{nc}^{N-path} \cdot t_c}{n_{total}^{N-path} \cdot \left(t_c + t_h + \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h}\right)}}$$

$$= \frac{1}{1 - \frac{n_{nc}^{N-path}}{n_{total}^{N-path}} \cdot \frac{1}{\left(1 + \frac{t_h}{t_c} + \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h \cdot t_c}\right)}} \tag{5.22}$$

The values of t_h, t_c, λ_h and $S\left(cert_{avg}^{N-path}\right)$ are all positive and finite. Therefore, $\frac{1}{\left(1 + \frac{t_h}{t_c} + \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h \cdot t_c}\right)}$ is between, but excluding, 0 and 1. Moreover, $n_{nc}^{N-path} < n_{total}^{N-path}$. Thus, the denominator of Equation 5.22 is also between, but excluding, 0 and 1. Therefore, provided that n_{nc}^{N-path} is not zero, the relative speed-up factor $f_{path(h,\,c,\,N\text{ - }path)}$, which is

given by Equation 5.22, is always larger than 1. This conclusion proves the following theorem.

Theorem 3: Let *N-path* be a NPKI certificate path with at least one nested certificate on it. Let *C-path* be a classical certificate path with the same number of certificates as the *N-path*. Assuming that the average certificate sizes, the public key cryptosystems and the hash functions used are the same in both paths, the verification of the *N-path* is always computationally more efficient than the verification of the *C-path*.

5.1.4. Analysis of NPKI Certificate Path Verification Speed-up Factor

The speed-up factor $f_{path}(h, c, N\text{-}path)$ given by Equation 5.22 is dependent on three parameters. These are the followings.

1. The cryptosystem – hash function pair. The t_h, t_c and λ_h values are determined depending on the cryptosystem and hash function used. t_h is inversely, t_c and λ_h are directly related to speed-up factor.

2. The degree of nested certification. This value, which is denoted as $\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}$ in Equation 5.22, is the fraction of the number of nested certificates over the number of all certificates on the *N-path*. The degree of nested certification is a value between 0 and 1, but it cannot be 1 since there must be at least one classical certificate on the *N-path*. If it is 0, then it means there is no nested certificate on the path, therefore the path becomes a classical certificate path. The degree of nested certification is directly related to the speed-up factor.

3. The average certificate size $S\left(cert_{avg}^{N-path}\right)$. The average certificate size is inversely related to the speed-up factor.

The NPKI certificate path verification speed-up factor is also relevant with the subject certificate verification speed-up factor. As an example case, the behavior of the both speed-up factors will be analyzed comparatively for the case where *h* = SHA-1 and *c* = RSA-512. The behavior of the speed-up factors $f_{(SHA1,\, RSA512,\, cert_{avg}^{N\,-\,path})}$ and $f_{path\,(SHA1,\, RSA512,\, \text{N - path})}$ versus the average certificate size, $S\left(cert_{avg}^{N-path}\right)$, and the degree of nested certification, $\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}$, is given in Figure 5.3. In Figure 5.3, the upper curve, with all '*' characters, is the curve of $f_{(SHA1,\, RSA512,\, cert_{avg}^{N\,-\,path})}$ versus $S\left(cert_{avg}^{N-path}\right)$ and it is drawn by using Equation 5.3. The mesh below this curve is the mesh for $f_{path\,(SHA1,\, RSA512,\, \text{N - path})}$ versus $S\left(cert_{avg}^{N-path}\right)$ and $\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}$, which is drawn by using Equation 5.22. The t_{SHA1}, t_{RSA512} and the λ_{SHA1} values of Table 5.1 are used. As can be seen from Figure 5.3, the $f_{path\,(SHA1,\, RSA512,\, \text{N - path})}$ values are always less than the $f_{(SHA1,\, RSA512,\, cert_{avg}^{N\,-\,path})}$ values for all of the average certificate sizes. Moreover, $f_{path\,(SHA1,\, RSA512,\, \text{N - path})}$ values approach $f_{(SHA1,\, RSA512,\, cert_{avg}^{N\,-\,path})}$ values as $\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}$ approaches to 1.

Moreover, as can be seen from Figure 5.3, the change in speed-up factor $f_{path\,(SHA1,\, RSA512,\, \text{N - path})}$ is more sensitive to the degree of nested certification as compared to average certificate size. For example, $f_{path\,(SHA1,\, RSA512,\, \text{N - path})}$ is 5.13, where average certificate size is 5000 and the degree of nested certification is 0.9. For the same average certificate size, $f_{path\,(SHA1,\, RSA1024,\, \text{N - path})}$ becomes 2.68 and 1.82, where the degree of nested certification is 0.7 and 0.5 respectively. On the other hand, in the case where the degree of

nested certification is 0.9, $f_{path}(SHA1, RSA1024, \text{N - path})$ becomes 5.60 and 6.18, where the average certificate size is 4000 and 3000 respectively.

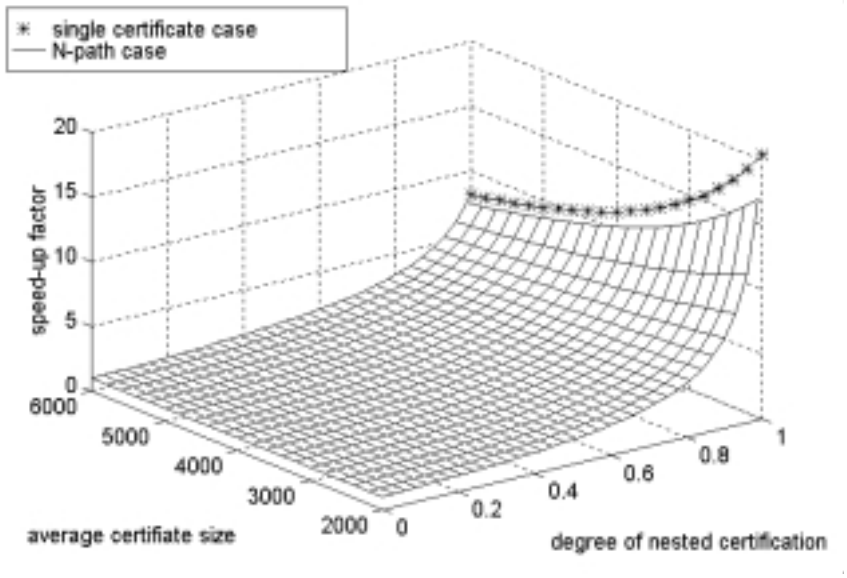

Figure 5.3. The speed-up factors of single subject certificate verification and the NPKI certificate path verification over the cryptographic certificate verification and the classical certificate path verification, where SHA-1 and RSA with 512-bit modulus are used

5.1.5. Analytical Comparison of Subject Certificate Verification and NPKI Certificate Path Verification Speed-up Factors

As graphically discussed above, the NPKI certificate path verification speed-up factor is always less than the subject certificate verification speed-up factor for all certificate sizes and for a given cryptosystem – hash function pair. In this subsection, this conclusion will be proven analytically. Moreover, it will also be shown that the speed-up factor for the NPKI certificate path verification only approaches to the corresponding subject certificate verification speed-up factor.

In order to compare the speed-up factor for the verification of a certificate *cert* as a subject certificate with the speed-up factor for the verification of a NPKI certificate path *N*-

path, the same cryptosystem *c* and hash function *h* are assumed to be used in both cases. Moreover, the size of *cert* is assumed to be the same as the average certificate size of *N-path*. That means, $S(cert)$ is assumed to be equal to $S(cert_{avg}^{N-path})$. The ratio of the speed-up factor for the verification of a certificate *cert* as a subject certificate over the speed-up factor for the verification of a NPKI certificate path *N-path* is denoted as $r_{(h,c,\text{N - path})}$. Under these assumptions, $r_{(h,c,\text{N - path})}$ is given by Equation 5.23. The derivation is as follows.

$$r_{(h,c,\text{N - path})} = \frac{f_{(h,c,cert)}}{f_{path}(h,c,\text{N - path})} = \frac{1+\dfrac{1}{\dfrac{t_h}{t_c}+\dfrac{S(cert)}{\lambda_h \cdot t_c}}}{1-\dfrac{n_{nc}^{N-path}}{n_{total}^{N-path}}\cdot\dfrac{1}{\left(1+\dfrac{t_h}{t_c}+\dfrac{S(cert_{avg}^{N-path})}{\lambda_h \cdot t_c}\right)}}$$

$$= \frac{1+\dfrac{1}{\dfrac{t_h}{t_c}+\dfrac{S(cert_{avg}^{N-path})}{\lambda_h \cdot t_c}}}{1-\dfrac{n_{nc}^{N-path}}{n_{total}^{N-path}}\cdot\dfrac{1}{\left(1+\dfrac{t_h}{t_c}+\dfrac{S(cert_{avg}^{N-path})}{\lambda_h \cdot t_c}\right)}} = \frac{\dfrac{t_h}{t_c}+\dfrac{S(cert_{avg}^{N-path})}{\lambda_h \cdot t_c}+1}{\dfrac{t_h}{t_c}+\dfrac{S(cert_{avg}^{N-path})}{\lambda_h \cdot t_c}+1-\dfrac{n_{nc}^{N-path}}{n_{total}^{N-path}}}$$

$$= \frac{\dfrac{1}{\dfrac{t_h}{t_c} + \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h \cdot t_c}}}{\dfrac{1}{\dfrac{t_h}{t_c} + \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h \cdot t_c} + 1 - \dfrac{n_{nc}^{N-path}}{n_{total}^{N-path}}}} = \frac{\dfrac{t_h}{t_c} + \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h \cdot t_c} + 1 - \dfrac{n_{nc}^{N-path}}{n_{total}^{N-path}}}{\dfrac{t_h}{t_c} + \dfrac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h \cdot t_c}} \tag{5.23}$$

As discussed in Section 5.1.3, the degree of nested certification $\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}$ is between, but excluding, 0 and 1. Therefore, as Equation 5.23 implies, $r_{(h, c, \text{N - path})}$ is always greater than 1. The implication of this conclusion is as follows.

$$r_{(h, c, \text{N - path})} > 1 \Rightarrow \frac{f_{(h, c, cert)}}{f_{path\,(h, c, \text{N - path})}} > 1 \Rightarrow f_{(h, c, cert)} > f_{path\,(h, c, \text{N - path})}$$

That means, the speed-up factor for subject certificate verification is always greater than the speed-up factor for NPKI certificate path verification. In other words, the speed-up factor for the NPKI certificate path verification cannot reach the speed-up factor for the subject certificate verification. This conclusion proves the following theorem.

Theorem 4: Let *N-path* be a NPKI certificate path with at least one nested certificate on it. Let *cert* be a certificate path of which the size is equal to the average certificate size of *N-path*. Assuming that the same public key cryptosystems and the hash functions are used in both cases, the speed-up factor for the verification of *N-path* is always less than the speed-up factor for the verification of *cert*.

Although the speed-up factor is smaller for NPKI certificate paths, there is always improvement as proven by Theorem 3 and this speed-up factor is greater for the paths with large degree of nested certification. As discussed in Section 5.1.4, the speed-up factors for

the NPKI certificate path verification approach to the speed-up factors for subject certificate verification as the degree of nested certification approaches to 1. This intuition is proven here. In order to prove it, the limit of the NPKI certificate path speed-up factor will be calculated as the degree of nested certification approaches to 1.

$$\lim_{\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}\to 1} f_{path}(h,c,\text{N - path}) = \lim_{\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}\to 1} \frac{1}{1-\frac{n_{nc}^{N-path}}{n_{total}^{N-path}}\cdot\frac{1}{\left(1+\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h\cdot t_c}\right)}}$$

$$= \frac{1}{1-1\cdot\frac{1}{\left(1+\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h\cdot t_c}\right)}} = \frac{1}{\frac{\left(1+\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h\cdot t_c}\right)-1}{\left(1+\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h\cdot t_c}\right)}}$$

$$= \frac{\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h\cdot t_c}+1}{\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h\cdot t_c}} = 1+\frac{1}{\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h\cdot t_c}} \quad (5.24)$$

The limit given by Equation 5.24 is nothing but the speed-up factor for the verification of $cert_{avg}^{N-path}$ as a subject certificate. Substituting this speed-up factor as given by Equation 5.3, Equation 5.24 becomes as the following.

$$\lim_{\frac{n_{nc}^{N-path}}{n_{total}^{N-path}} \to 1} f_{path}(h, c, \text{N - path}) = 1 + \frac{1}{\frac{t_h}{t_c} + \frac{S\left(cert_{avg}^{N-path}\right)}{\lambda_h \cdot t_c}} = f_{(h, c, cert_{avg}^{N-path})} \quad (5.25)$$

As can be seen from Equation 5.25, the speed-up factor for the NPKI certificate path verification approaches to the speed-up factor for a single subject certificate verification, as the degree of nested certification approaches to 1. This conclusion proves the following theorem.

Theorem 5: Let *N-path* be a NPKI certificate path and *cert* be a certificate, of which the size is the average certificate size of *N-path*. The speed-up factor for the verification of *N-path* approaches to the speed-up factor for the verification of *cert* as a subject certificate, while the degree of nested certification of *N-path* approaches to 1.

5.1.6. Analysis of Nested Certificate Path Verification

The nested certificate path concept has been explained in Section 3.6. The nested certificate paths can be considered as special NPKI certificate paths on which only the last certificate is a classical certificate, other certificates are nested certificates. Therefore, for a nested certificate path, *ncpath*:

$$n_{cc}^{ncpath} = 1 \quad (5.26)$$

$$n_{total}^{ncpath} = n_{nc}^{ncpath} + n_{cc}^{ncpath} = n_{nc}^{ncpath} + 1 \quad (5.27)$$

where n_{cc}^{ncpath} is the total number of classical certificates, n_{nc}^{ncpath} is the total number of nested certificates of *ncpath*. n_{total}^{ncpath} is the total number of certificates of *ncpath*. Under

these considerations, the degree of nested certification in *ncpath* can be formulated as Equation 5.28.

$$\frac{n_{nc}^{ncpath}}{n_{total}^{ncpath}} = \frac{n_{nc}^{ncpath}}{n_{nc}^{ncpath}+1} \tag{5.28}$$

By using Equation 5.22, the speed-up factor for the verification of nested certificate path, *ncpath*, can be formulated as follows.

$$f_{path}(h, c, \text{ncpath}) = \frac{1}{1-\frac{n_{nc}^{ncpath}}{n_{total}^{ncpath}} \cdot \frac{1}{\left(1+\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{ncpath}\right)}{\lambda_h \cdot t_c}\right)}}$$

$$= \frac{1}{1-\frac{n_{nc}^{ncpath}}{n_{nc}^{ncpath}+1} \cdot \frac{1}{\left(1+\frac{t_h}{t_c}+\frac{S\left(cert_{avg}^{ncpath}\right)}{\lambda_h \cdot t_c}\right)}} \tag{5.29}$$

As can be seen from Equation 5.28 easily, the degree of nested certification in *ncpath* approaches to 1, while the number of nested certificates on *ncpath*, n_{nc}^{ncpath}, approaches to infinity. Therefore, by Theorem 5, the speed-up factor for the verification of *ncpath* approaches to the corresponding speed-up factor for subject certificate verification. This conclusion proves the following corollary.

Corollary 1: Let *ncpath* be a nested certificate path and *cert* be a certificate, of which the size is the average certificate size of *ncpath*. The speed-up factor for the

verification of *ncpath* approaches to the speed-up factor for the verification of *cert* as a subject certificate, while the number of nested certificates on *ncpath* approaches to infinity.

The behavior described in Corollary 1 is exemplified in Figure 5.4 for different algorithm combinations. In this figure, the behavior of the speed-up factor $f_{path}(h, c, \text{ncpath})$ versus the number of nested certificates on *ncpath*, n_{nc}^{ncpath}, is given. The t_h, t_c and the λ_h values of Table 5.1 are used. The average certificate size of *ncpath*, $S\left(cert_{avg}^{ncpath}\right)$, is assumed to be fixed and 3000 bits for all certificates on the paths.

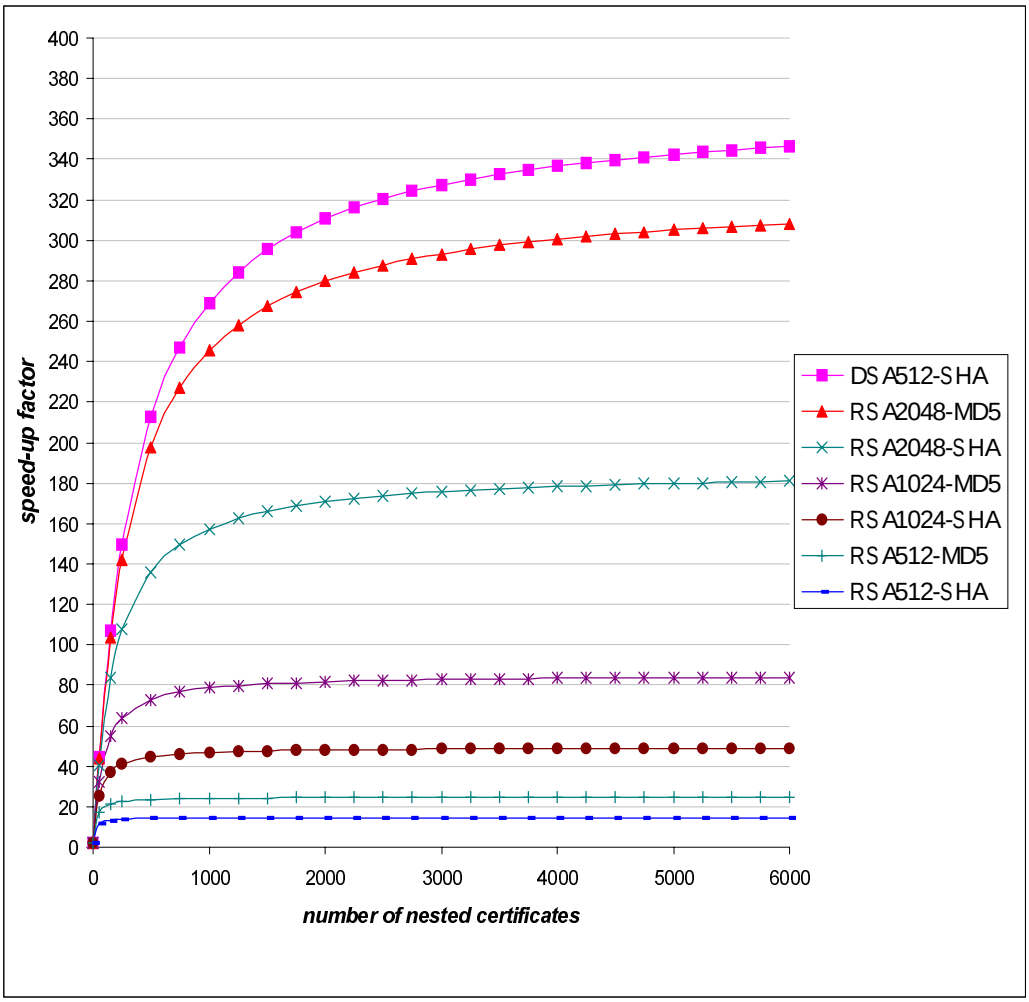

Figure 5.4. Change of speed-up factor with respect to number of nested certificates on nested certificate paths for different algorithm combinations

As can be seen from Figure 5.4, the speed-up values increase asymptotically as the number of nested certificates increases. Moreover, the values that the nested certificate path verification speed-up factors approach are the subject certificate verification speed-up factors for 3000 bits certificate sizes, which are given by Figure 5.1 and Figure 5.2. For example, as can be seen from Figure 5.4, for RSA2048-SHA1 algorithm pair and 3000 bits certificate size, the nested certificate path speed-up factor approaches to 185. As expected, for the same algorithm pair and the certificate size, the subject certificate verification speed-up factor is also 185, as can be seen from Figure 5.1. These results are, actually, confirmation of Corollary 1.

As can be seen from Figure 5.4, the speed-up factors approach their maximum values for large number of nested certificates. However, it is not practical to have even hundreds of nested certificates on a nested certificate path. The practical speed-up factors have to be analyzed for smaller (like 1 to 8) number of nested certificates. Such an analysis is given by Figure 5.5. In this analysis, several algorithm pairs are used. The t_h, t_c and the λ_h values of Table 5.1 are used. The average certificate size is taken 3000 bits as in the previous analysis. As can be seen from Figure 5.5, the speed-up factors for smaller number of nested certificates are not as large as the ones for large number of nested certificates. However, the improvement is still acceptable. For the cases considered, the speed-up factors are between 1.87 and 8.94. That means, the nested certificate path verification performs 1.87 to 8.94 times faster than classical certificate path verification.

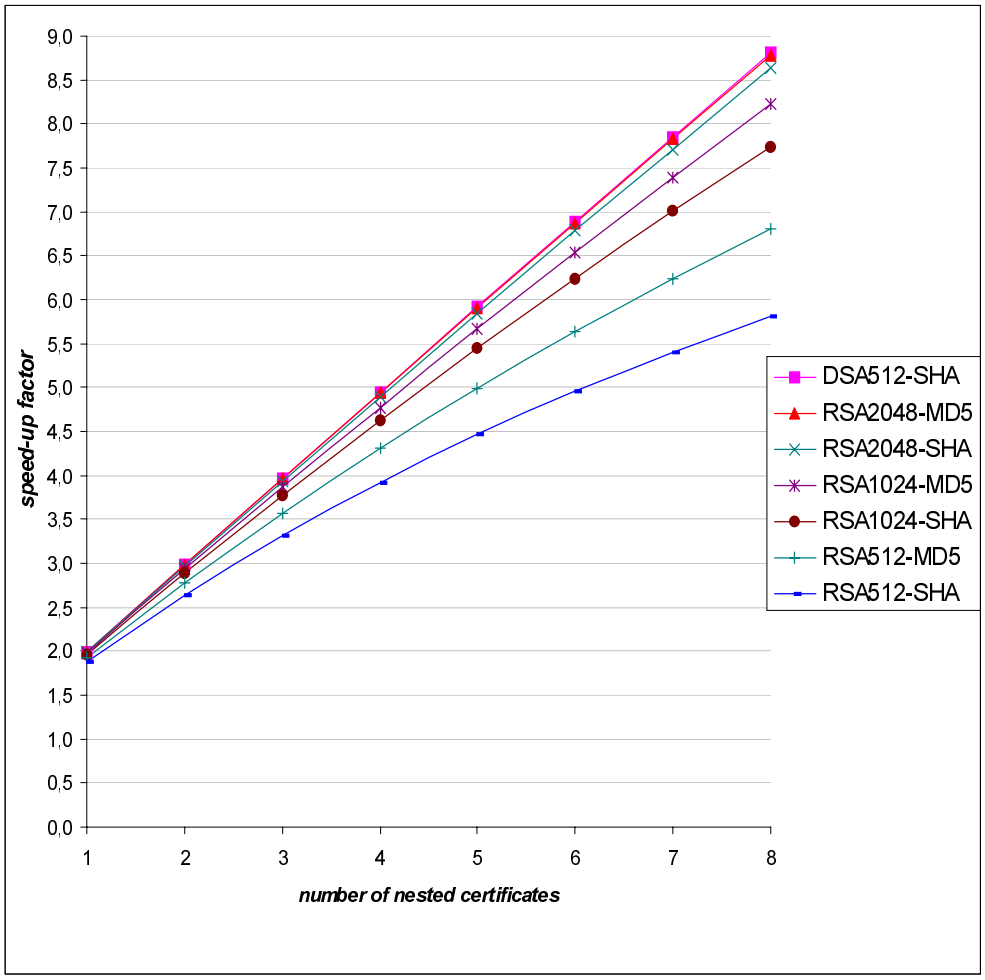

Figure 5.5. Change of speed-up factor with respect to number of nested certificates on nested certificate paths for different algorithm combinations for small number of nested certificates

5.2. Simulation based Performance Evaluation

In this section, the simulation results will be given for the relative efficiency improvement values of the subject certificate verification and NPKI certificate path verification methods over the cryptographic certificate verification and the classical certificate path verification methods respectively. Moreover, relative efficiency improvement of the nested certificate path verification method over classical certificate

path verification method will be analyzed for short paths. The simulations were performed on a Pentium 166 computer using the cryptographic library of the SECUDE toolkit [86].

5.2.1. Performance Evaluation of Subject Certificate Verification Method

In this subsection, the simulation results for the efficiency improvement of the subject certificate verification method over the cryptographic certificate verification method will be given. In the simulation studies, a certificate, *cert*, is verified both cryptographically and as a subject certificate of a nested certificate. For the cryptographic verification, it is assumed that the correct public key of the issuer of *cert* is known. Similarly, for the subject certificate verification, it is assumed that there is a legitimate nested certificate of which the subject certificate is *cert*. The important parameters that effect the efficiency are the hash algorithm and the public key cryptosystem used in cryptographic verification as well as the hash algorithm used in the subject certificate verification. The hash algorithm used in the cryptographic verification may be different from the hash algorithm used in the subject certificate verification. In the simulations, the RSA [12] and DSA [32] cryptosystems with different key sizes, MD5 [15] and SHA-1 [16] hash algorithms are used. The size of *cert* is assumed to be fixed and 300 octets.

In Table 5.2, execution times of cryptographic and subject certificate verification methods are given for different cryptosystem and hash algorithm combinations. Two relative improvement measures are used to compare the performances of the cryptographic and subject certificate verification methods. One of them is the *time saving*, which indicates the amount of the time saved by using the subject certificate verification method instead of the cryptographic certificate verification method. Second improvement measure is the *speed-up factor*, which indicates how many times the subject certificate verification method is faster than the cryptographic verification method.

As can be seen from Table 5.2, there is a remarkable improvement in the subject certificate verification method as compared to the cryptographic verification. For the

algorithm combinations given in Table 5.2, the subject certificate verification method is 13.3 to 1587.4 times faster than the cryptographic certificate verification method. The primary factor that effects the speed-up factor is the public key cryptosystem used for the cryptographic verification. The reason is that the hash algorithms are much faster than the public key cryptosystem operations, so that the effects of the hash algorithms over the execution times and the speed-up factors are not as significant as the public key cryptosystems. Moreover, hashing is also employed in the subject certificate verification method, whereas public key cryptosystem operations are not. Therefore, the time spent for public key cryptosystem operation is the saving of the subject certificate verification method. That is why the time saving and speed-up factor values are larger for the slower cryptosystems, like DSA512, DSA1024 and RSA2048.

Table 5.2. Performance values for cryptographic and subject certificate verification methods

Algorithms			Verification Time (milliseconds)			
Cryptosystem for Cryptographic Verification	Hash Function for Cryptographic Verification	Hash Function for Subject Certificate Verification	Cryptographic Verification (***CV***)	Subject Certificate Verification (***SCV***)	Time Saving ***CV-SCV***	Speed-up Factor ***CV/SCV***
DSA512	SHA-1	SHA-1	33.800	0.102	33.698	331.4
DSA512	SHA-1	MD5	33.726	0.071	33.655	475.0
DSA1024	SHA-1	SHA-1	113.688	0.102	113.586	1114.6
DSA1024	SHA-1	MD5	112.707	0.071	112.636	1587.4
RSA512	MD5	MD5	1.343	0.070	1.273	19.2
RSA512	MD5	SHA-1	1.345	0.101	1.244	13.3
RSA512	SHA-1	MD5	1.376	0.071	1.305	19.4
RSA512	SHA-1	SHA-1	1.374	0.101	1.273	13.6
RSA1024	MD5	MD5	4.573	0.071	4.502	64.4
RSA1024	MD5	SHA-1	4.572	0.103	4.469	44.4
RSA1024	SHA-1	MD5	4.600	0.072	4.528	63.9
RSA1024	SHA-1	SHA-1	4.602	0.101	4.501	45.6
RSA2048	MD5	MD5	17.224	0.071	17.153	242.6
RSA2048	MD5	SHA-1	17.223	0.102	17.121	168.9
RSA2048	SHA-1	MD5	17.221	0.072	17.149	239.2
RSA2048	SHA-1	SHA-1	17.219	0.104	17.115	165.6

As can be seen from Table 5.2, there is a significant improvement in the subject certificate verification method. However, these results depend on the assumption that the nested certificate for *cert* is legitimate. Nevertheless, this nested certificate must also be verified either cryptographically or as a subject certificate of another nested certificate. This reasoning eventually yields nested or NPKI certificate paths. Simulation results regarding the NPKI certificate path verification are given in the next subsection. Simulation results regarding the nested certificate path verification are given in Section 5.2.3.

5.2.2. Performance Evaluation of NPKI Certificate Path Verification

In this subsection, the simulation results for the efficiency improvement of the NPKI certificate path verification method over the classical certificate path verification method is given.

In the simulation study, the verification time for a classical certificate path with 5 certificates is compared with the verification time of NPKI certificate paths of the same length. For each classical certificate path, four NPKI certificate paths are produced with 1, 2, 3 and 4 nested certificates over a total of 5 certificates. Other certificates of the NPKI certificate paths are the classical ones. Figure 5.6a shows the generic classical certificate path used in the simulation. Figure 5.6b, Figure 5.6c, Figure 5.6d and Figure 5.6e show the NPKI certificate paths produced from the classical certificate path with 1, 2, 3 and 4 nested certificates respectively. Different simulation runs are performed by using the different combinations of RSA [12] and DSA [32] cryptosystems with different key sizes, MD5 [15] and SHA-1 [16] hash algorithms. For the sake of simplicity, in a single simulation run, the same cryptosystem and the same hash algorithm are used for all of the cryptographic verifications of the classical and nested certificates on the paths. Moreover, the same hash algorithm is used for all of the subject certificate verifications on the NPKI certificate paths of a single simulation run. However, this hash algorithm may be different from the hash algorithm used for cryptographic certificate verifications.

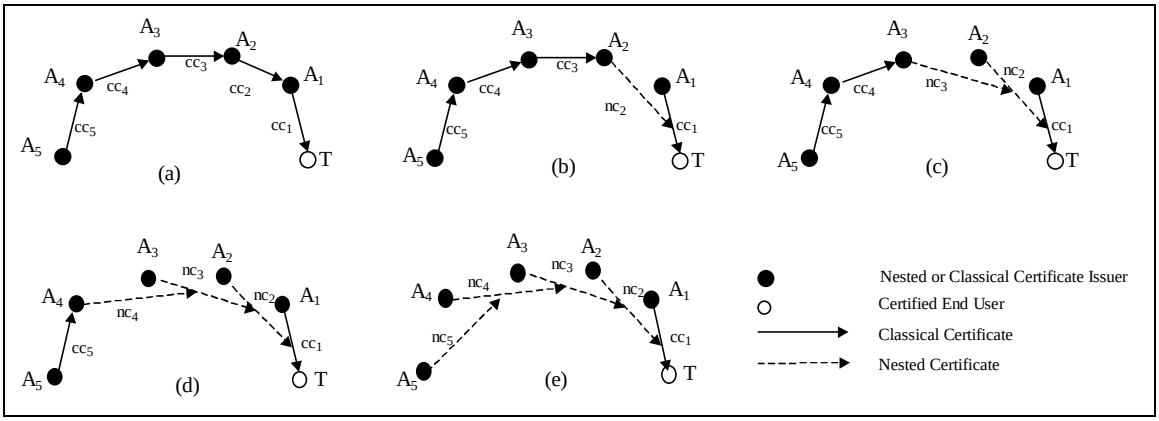

Figure 5.6. Classical and NPKI certificate paths used in the simulation

Table 5.3. Execution times and time saving values of the verification of classical and NPKI certificate paths

Algorithms			Verification Times (milliseconds)					Time Savings (milliseconds)			
Cryptosystem for Cryptographic Verifications	Hash Algorithm for Cryptographic Verifications	Hash Algorithm for Subject Certificate Verifications	Classical Path with 5 Classical Certificates (***CP***)	NPKI path with 1/5 Nested Certificate (***NCP*** $_1$)	NPKI path with 2/5 Nested Certificates (***NCP*** $_2$)	NPKI path with 3/5 Nested Certificates (***NCP*** $_3$)	NPKI path with 4/5 Nested Certificates (***NCP*** $_4$)	NPKI path with 1/5 Nested Certificates ***CP-NCP*** $_1$	NPKI path with 2/5 Nested Certificates ***CP-NCP*** $_2$	NPKI path with 3/5 Nested Certificates ***CP-NCP*** $_3$	NPKI path with 4/5 Nested Certificates ***CP-NCP*** $_4$
DSA512	SHA-1	SHA-1	167.778	133.607	100.557	67.891	34.484	34.171	67.221	99.887	133.294
DSA512	SHA-1	MD5	168.696	135.109	101.201	67.432	34.187	33.587	67.495	101.264	134.509
DSA1024	SHA-1	SHA-1	562.956	451.214	337.216	224.747	114.640	111.742	225.740	338.209	448.316
DSA1024	SHA-1	MD5	565.478	452.369	340.304	226.768	114.664	113.109	225.174	338.710	450.814
RSA512	MD5	MD5	6.705	5.378	4.105	2.834	1.560	1.327	2.600	3.871	5.145
RSA512	MD5	SHA-1	6.665	5.413	4.177	2.940	1.700	1.252	2.488	3.725	4.965
RSA512	SHA-1	MD5	6.753	5.472	4.180	2.887	1.590	1.281	2.573	3.866	5.163
RSA512	SHA-1	SHA-1	6.751	5.516	4.249	3.001	1.720	1.235	2.502	3.750	5.031
RSA1024	MD5	MD5	22.811	18.304	13.805	9.312	4.811	4.507	9.006	13.499	18.000
RSA1024	MD5	SHA-1	22.818	18.362	13.909	9.449	4.989	4.456	8.909	13.369	17.829
RSA1024	SHA-1	MD5	22.972	18.458	13.928	9.399	4.867	4.514	9.044	13.573	18.105
RSA1024	SHA-1	SHA-1	22.973	18.495	14.006	9.513	5.022	4.478	8.967	13.460	17.951
RSA2048	MD5	MD5	86.143	68.947	51.813	34.678	17.536	17.196	34.330	51.465	68.607
RSA2048	MD5	SHA-1	86.172	69.190	52.094	34.998	17.895	16.982	34.078	51.174	68.277
RSA2048	SHA-1	MD5	86.346	69.198	52.026	34.831	17.650	17.148	34.320	51.515	68.696
RSA2048	SHA-1	SHA-1	86.353	69.260	52.130	34.990	17.850	17.093	34.223	51.363	68.503

Table 5.4. Speed-up factors for the verification of NPKI over classical certificate paths

Algorithms			Speed-up Factor			
Cryptosystem for Cryptographic Verifications	Hash Algorithm for Cryptographic Verifications	Hash Algorithm for Subject Certificate Verifications	NPKI path with 1/5 Nested Certificates CP/NCP_1	NPKI path with 2/5 Nested Certificates CP/NCP_2	NPKI path with 3/5 Nested Certificates CP/NCP_3	NPKI path with 4/5 Nested Certificates CP/NCP_4
DSA512	SHA-1	SHA-1	1.26	1.67	2.47	4.87
DSA512	SHA-1	MD5	1.25	1.67	2.50	4.93
DSA1024	SHA-1	SHA-1	1.25	1.67	2.50	4.91
DSA1024	SHA-1	MD5	1.25	1.66	2.49	4.93
RSA512	MD5	MD5	1.25	1.63	2.37	4.30
RSA512	MD5	SHA-1	1.23	1.60	2.27	3.92
RSA512	SHA-1	MD5	1.23	1.62	2.34	4.25
RSA512	SHA-1	SHA-1	1.22	1.59	2.25	3.93
RSA1024	MD5	MD5	1.25	1.65	2.45	4.74
RSA1024	MD5	SHA-1	1.24	1.64	2.41	4.57
RSA1024	SHA-1	MD5	1.24	1.65	2.44	4.72
RSA1024	SHA-1	SHA-1	1.24	1.64	2.41	4.57
RSA2048	MD5	MD5	1.25	1.66	2.48	4.91
RSA2048	MD5	SHA-1	1.25	1.65	2.46	4.82
RSA2048	SHA-1	MD5	1.25	1.66	2.48	4.89
RSA2048	SHA-1	SHA-1	1.25	1.66	2.47	4.84

As in the subject certificate verification case, *time saving* and the *speed-up factor* are used as the performance improvement measures. The verification times and the time saving values are given in Table 5.3. As can be seen from this table and Table 5.2, the time saved by the verification of a NPKI certificate path is greater than or almost the same as the time saved by the subject certificate verification method of the same algorithm triple. Moreover, as can be seen from Table 5.3, the time saving values increase as the number of nested certificates within the NPKI certificate path increases. The reason for this behavior can be explained as follows. The total number of subject certificate verifications in a NPKI certificate path is the same as the total number of nested certificates on it. Therefore, if the total number of nested certificates on the NPKI certificate path is *n*, then it can be said that *n* cryptographic certificate verifications are replaced with subject certificate verifications. Moreover, since the subject certificate verification time is less than the cryptographic certificate verification time, the time saving values for the NPKI certificate path

verification method increase by the number of nested certificates within the NPKI certificate path.

The speed-up factor values for the same algorithm triples are given in Table 5.4. The values in this table are calculated using the verification time values of Table 5.3. Table 5.4 shows that the speed-up factors, as the time saving values, are also directly related to the number of nested certificates within the NPKI certificate paths. However, as can be seen from Table 5.4, these speed-up factors are not as large as the speed-up factors of the subject certificate verification case. The speed-up factors, which have been given for the subject certificate verification method in Table 5.2, would be able to be obtained in the NPKI certificate path verification method, if all of the certificates of the NPKI certificate path were verified as subject certificates. However, at least one of the certificates of a NPKI certificate path must be verified cryptographically as explained in Section 4.3. Therefore, in the NPKI certificate path verification method, it is not possible to achieve the speed-up factors of the subject certificate verification method. However, this does not mean that the NPKI certificate path verification method is inefficient, since the NPKI certificate path verification method is 1.22 to 4.93 times faster than the classical certificate path verification method for the example certificate paths considered.

5.2.3. Performance Evaluation of Nested Certificate Path Verification

Nested certificate paths were explained in Section 3.6. Analytical performance analysis of nested certificate path verification was given in Section 5.1.6. In this subsection, simulation based performance analysis of nested certificate path verification is given.

For the verification of a nested certificate path with n+1 certificates (one classical certificate + n nested certificates), one cryptographic and n subject certificate verifications are performed. On the other hand, n+1 cryptographic certificate verifications must be performed for the verification of a classical certificate path of the same length. That means,

n cryptographic certificate verifications are replaced with subject certificate verifications in the case of nested certificate path verification. Since the subject certificate verification method is more efficient than the cryptographic certificate verification, nested certificate path verification is also more efficient than the verification of a classical certificate path of the same length. Moreover, higher relative improvement is expected for the cases of larger *n*, where *n* is the number of nested certificates on the path. As a matter of fact, simulation studies show that the efficiency improvement in the nested certificate path verification method is directly related to *n*.

In the simulations, the effect of the number of nested certificates on the nested certificate paths over the relative improvement is examined. As all other analyses, the relative improvement measure is the *speed-up factor*, which is the ratio of the verification time of a classical certificate path over the verification time of a nested certificate path of the same length. In other words, the speed-up factor indicates how many times the nested certificate path verification is faster than the classical certificate path verification. For the sake of simplicity and uniformity, the same hash algorithms and the same public-key cryptosystems are used for all of the certificates on the paths. Moreover, the number of certificates for both classical and nested certificate paths is the same. Eight sets of simulations are performed; each uses a different pair of public-key cryptosystem (RSA [12] or DSA [32] with different key sizes) and hash algorithm (MD5 [15] or SHA-1 [16]). In each set, the number of nested certificates on the nested certificate paths has been taken in the range of 1 to 8, since these lengths are practical path lengths. Since there is one classical certificate at the end of a nested certificate path, this range corresponds to 2 to 9 total (including nested certificates and the classical certificate) certificates. The results for these simulations are shown in Figure 5.7.

As can be seen from Figure 5.7, there is a remarkable improvement especially for slower cryptosystems, like DSA-512, RSA-2048 and RSA-1024. For the cases considered, the speed-up factors are between 1.87 and 8.83.

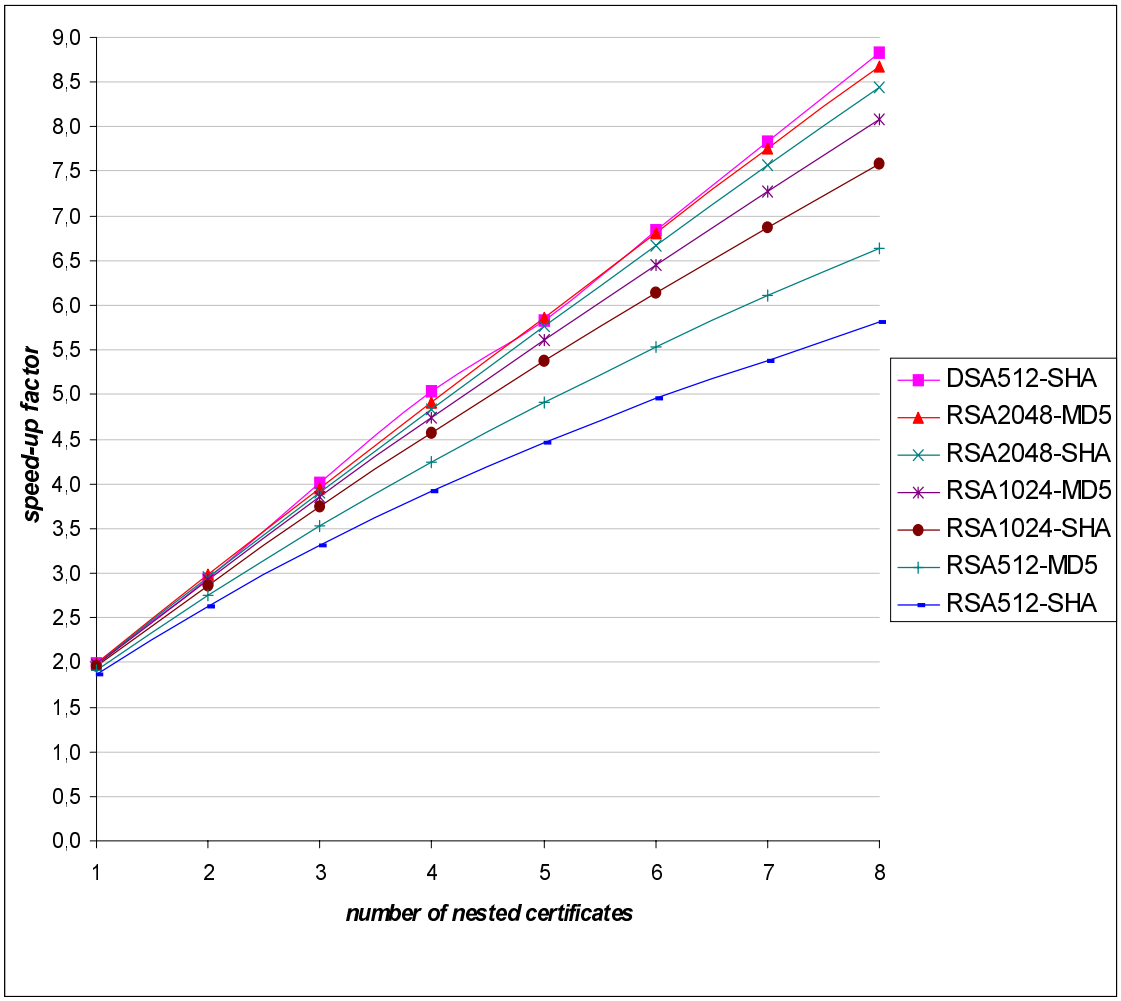

Figure 5.7. Simulation results for the change of speed-up factor with respect to the number of nested certificates on the nested certificate paths

5.3. Comparison of Analytical and Simulation Results

As can be seen from Sections 5.1 and 5.2, the general characteristics of the analytical and the corresponding simulation results are similar to each other. However, the are some differences between the corresponding analytical and simulation measurements. In this subsection, these differences and their causes are discussed.

According to the observations done, the simulation environment causes some overheads for the cryptographic and subject certificate verification methods. The overhead is 0.05 – 0.1 msec per cryptographic certificate verification, 0.01 – 0.015 msec per subject certificate verification. That means, one simulation based cryptographic verification takes 0.05 to 0.1 msec (0.5 to 4 per cent) more than the corresponding analytically computed time. Similarly, one subject certificate verification takes 0.01 to 0.015 msec (10 to 15 per cent) more in simulation environment as compared to the analytical calculations.

The reason for these overheads is the implementation overhead. The analytical values are obtained by considering only the hash calculations and public key cryptosystem signature verification operations. However, in the implementation, there are some other operations, like comparisons, subroutine calls, loops, etc. that take time.

The effects of the overheads of the simulation environment over the speed-up factors are listed as follows.

1. The subject certificate verification speed-up factor, which is analyzed in Section 5.1.2 analytically and in Section 5.2.1 using simulation, is very sensitive to the subject certificate verification time. Therefore, 10 – 15 per cent overhead in the subject certificate verification method, which is explained above, effects the subject certificate verification speed-up factor significantly. As can be seen from Sections 5.1.2 and 5.2.1, the simulation results are generally 5 to 20 per cent less than the analytical results.

2. On the other hand, more practical results were obtained with the path analyses given in Sections 5.1.4 and 5.1.6 analytically, in Sections 5.2.2 and 5.2.3 using simulation. Since the NPKI and nested certificate path verification methods include cryptographic certificate verifications, the corresponding speed-up factors are not sensitive to the subject certificate verification time as the subject certificate verification speed-up factors are. The simulation based NPKI and nested certificate path verification speed-up factors are 0 to 2 per cent less than the corresponding analytical speed-up factors.

Moreover, the DSA [32] algorithm has an interesting property that causes some inconsistencies between analytical and simulation results and some irregularities within the simulation results. This property is that a random number is associated with each message signed using DSA. Depending on the largeness of this number, signing and verification times may vary. Sometimes, the verification times become more or less than the expected values. Moreover, for some cases the analytical results are larger than the corresponding simulation results. For example, the analytical speed-up factor for the verification of the nested certificate path with 6 nested certificates using DSA algorithm with 512-bit key length is 6.89, whereas the corresponding simulation result is 6.84.

5.4. Number of Nested Certificates and Trade-off Analysis

As discussed in Section 4, there are two approaches for the NPKI formation. One of them is the *free certification* approach, second one is the *transition from an existing PKI* approach. The details of the latter approach are given in Sections 4.6 and 4.7. In the former approach, everyone is free to choose the certification method. Therefore, the application of it does not enforce any nested certificate issuance. However, the latter method enforces nested certificate issuance. The performance improvement analyses for nested and NPKI certificate path verifications were given in Sections 5.1 and 5.2. However, in order to obtain nested or NPKI certificate paths from a NPKI, several nested certificates must be issued. In this section, the analysis for the number of nested certificates will be given. Moreover, the trade-off between the number of nested certificates and the verification performance improvement will also be analyzed. For these analyses, the *nested certificate issuance towards end user* method of the *transition from an existing PKI* approach will be considered. As explained in Section 4.6.1, this method aims to construct nested certificate paths from each CA/NCA towards the end users in the NPKI.

5.4.1. Preliminaries

An *end user* is a user in PKI/NPKI, which does not issue certificates, but has certificates issued for it. The CAs/NCAs are the users of PKI/NPKI other than the end users. A *singular path* is a path with only one certificate. A *non-singular path* is a path with more than one certificate.

In the nested certificate issuance towards end user method, each CA/NCA, say *A*, issues nested certificates for the certificates that are issued by the CAs/NCAs that *A* had certified. The only restriction to the above rule is that no nested certificates are issued for the classical certificates towards CAs/NCAs, since the aim of this method is to construct nested certificate paths towards only the end users. The details and an example of the application of this method were given in Section 4.6.1. Here, the ways to find the number of nested certificates necessary to be issued are discussed.

The number of nested certificates that must be issued for a single end user is related to the number of classical certificate paths towards this end user. This intuition is formalized by the following axiom.

Axiom 1: The total number of nested certificates that must be issued in order to form one nested certificate path for each classical certificate path towards an end user *e* in a PKI is equal to the total number of distinct non-singular classical certificate paths from all CAs/NCAs of PKI towards *e*. These non-singular classical certificate paths may overlap, that is they may have common certificates.

As an example consider the partial PKI in Figure 5.8. In this figure, a partial PKI, which shows the classical certification relationships for a target entity *T*, and the partial NPKI formed by the application of the nested certificate propagation towards end user method are shown together. There are seven non-singular classical certificate paths

towards *T*. By the application of the nested certificate propagation method, seven nested certificates are added to the PKI and seven corresponding nested certificate paths are created. Table 5.5 gives these paths.

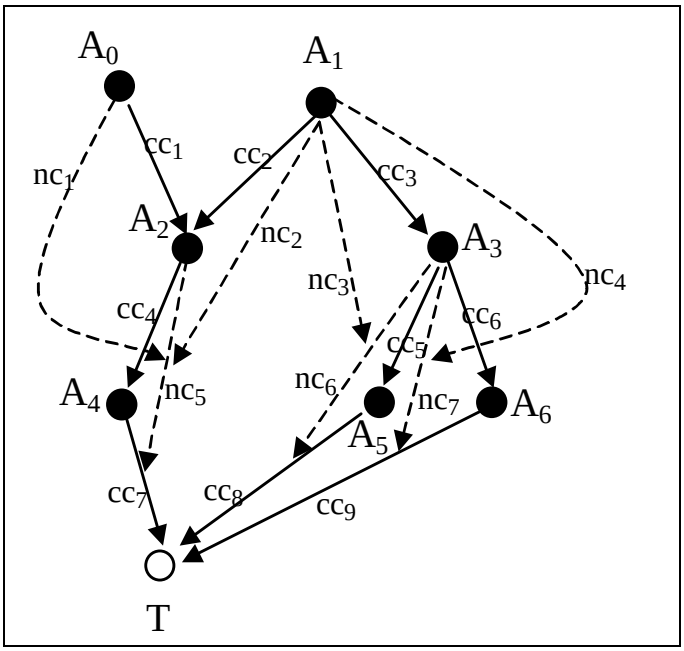

Figure 5.8. An example partial PKI and the application of the nested certificate propagation towards end user method for a single end user

Table 5.5. Paths in the example PKI and NPKI

Path #	**Non-singular classical certificate path**	**Corresponding nested certificate path**
1	A_2,cc_4,A_4,cc_7,T	A_2,nc_5,cc_7,T
2	$A_0,cc_1,A_2,cc_4,A_4,cc_7,T$	A_0,nc_1,nc_5,cc_7,T
3	$A_1,cc_2,A_2,cc_4,A_4,cc_7,T$	A_1,nc_2,nc_5,cc_7,T
4	A_3,cc_5,A_5,cc_8,T	A_3,nc_6,cc_8,T
5	$A_1,cc_3,A_3,cc_5,A_5,cc_8,T$	A_1,nc_3,nc_6,cc_8,T
6	A_3,cc_6,A_6,cc_9,T	A_3,nc_7,cc_9,T
7	$A_1,cc_3,A_3,cc_6,A_6,cc_9,T$	A_1,nc_4,nc_7,cc_9,T

The total number of nested certificates that must be issued by a CA/NCA is given by the next axiom.

Axiom 2: Suppose there is a PKI with the set of end user *E*. The total number of nested certificates that must be issued in order to form one nested certificate path for each classical certificate path between a CA/NCA *A* and the members of *E* equal to the total number of distinct non-singular classical certificate paths between *A* and the members of *E*. These non-singular classical certificate paths may overlap, that is they may have common certificates.

The total number of nested certificates that must be issued in the global network is related to the total number of paths towards the end users from all of the CAs/NCAs in the PKI, as formalized by the following axiom.

Axiom 3: Suppose there is a PKI with the set of end user *E*. A NPKI is required to be constructed from this PKI such that there will be one nested certificate path for each classical certificate path towards the members of *E*. The total number of nested certificates that must be issued to attain this goal is equal to the total number of distinct non-singular classical certificate paths from all of the CAs/NCAs towards the members of *E*. These non-singular classical certificate paths may overlap, that is they may have common certificates.

5.4.2. Formulation for a Tree Shaped Topology

In this section, the *Nested certificate propagation overhead* and the *average nested certificate path length* will be formulated for a specific tree shaped PKI/NPKI topology. Nested certificate propagation overhead is the factor of increase in the total number of certificates. This overhead value is always greater than or equal to 1. An overhead value 1 means that there is no overhead. An overhead value *x* means nested certificate propagation increases the number of total certificates *x* times. The average nested certificate path length

of the produced NPKI is also important in the analysis, since this value will give the idea about the efficiency gain for the PKI to NPKI transition.

Two nested certificate propagation overhead values will be formulated. One of them is the nested certificate propagation overhead to convert the whole PKI into NPKI, $NCPO_{PKI}$, which is the ratio of the total number certificates (nested + classical) in the NPKI over the number of classical certificates in the PKI. Other nested certificate propagation overhead is for a single authority *a*. This overhead value, $NCPO_a$, is the ratio of the total number certificates (nested + classical) issued by *a* over the number of classical certificates issued by *a*. In order to formulate these overhead values, the number of nested certificates must be formulated. According to the axioms given in Section 5.4.1, the number of nested certificates is related to the number of paths. The numbers of paths are specific to the topology of the PKI and it is mostly not possible to formulate them for irregular graph shaped PKIs. In the graph shaped PKIs, it is also impossible to formulate average nested certificate path length. Therefore, one should work on regular PKI topologies. That is why a balanced tree topology will be used in the analysis. The topology that will be analyzed in this subsection is a *k*-level *m*-ary balanced tree. Such a topology is chosen, since it is straightforward to enumerate the number of paths and the average nested certificate path length. Moreover, such tree topologies are also common in real life PKIs.

A generic *k*-level *m*-ary balanced tree is shown in Figure 5.9. In a *k*-level *m*-ary balanced tree shaped PKI, each non-leaf node issues *m* classical certificates to their children nodes and there are *k* non-leaf node levels.

Let V_i represent the set of nodes in the level *i*. In a PKI, there are two sets of users. These are the set of end users and the set of CAs. These two sets are disjoint, the means they have no common members. In the PKI to be analyzed, the end users are the leaf nodes of the tree. Therefore, the end user set is denoted as V_k. The set of CAs contains other nodes of the tree. The members of the set of CAs will also act as NCAs in the NPKI.

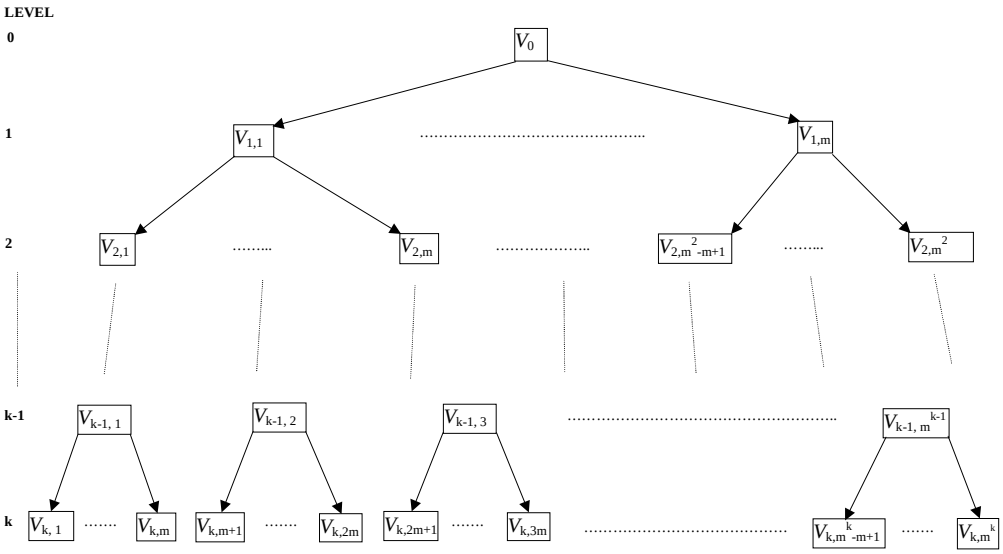

Figure 5.9. A generic *k*-level *m*-ary balanced tree

Let $v_i \in V_i$. As axiom 2 implies, the total number of nested certificates from v_i towards the end users, which is denoted as n_{v_i}, is equal to total number of non-singular paths from v_i towards the end users. Since there are no non-singular paths from the nodes in V_{k-1} and V_k towards the end users, $n_{v_{k-1}}$ and n_{v_k} values are zero. n_{v_i} is formulated as follows.

$$n_{v_i} = \begin{cases} \prod_{j=i+1}^{k} m = m^{k-i} & i < k-1 \\ 0 & i \geq k-1 \end{cases} \tag{5.30}$$

The number of nodes in the level *i*, which is denoted as $|V_i|$, is formulated as follows.

$$|V_i| = m^i \tag{5.31}$$

According to Axiom 3, the total number of nested certificates that must be issued to convert the whole PKI into NPKI is equal to the total number of non-singular paths towards the end users. Moreover, each node at the same level shows the same characteristics in terms of nested certificate issuance. Under these considerations, the total number of nested certificates, which is denoted as n_{PKI}, is formulated as the following.

$$n_{PKI} = \sum_{i=0}^{k} \sum_{v \in V_i} n_v = \sum_{i=0}^{k} |V_i| \cdot n_{v_i} = \sum_{i=0}^{k-2} |V_i| \cdot m^{k-i} = \sum_{i=0}^{k-2} m^i \cdot m^{k-i} = \sum_{i=0}^{k-2} m^k = (k-1) \cdot m^k \quad (5.32)$$

By definition, except the leaf nodes (a.k.a. the end users), each node in the PKI issues m classical certificates. Under this consideration, the total number of classical certificates in the PKI, which is denoted as c_{PKI}, is formulated as the following.

$$c_{PKI} = \sum_{i=0}^{k-1} m \cdot |V_i| = \sum_{i=0}^{k-1} m \cdot m^i = \sum_{i=0}^{k-1} m^{i+1} \quad (5.33)$$

Now the nested certificate propagation overhead values will be formulated. For a k-level m-ary balanced tree shaped PKI, $NCPO_{PKI}$ is formulated as follows.

$$NCPO_{PKI} = \frac{n_{PKI} + c_{PKI}}{c_{PKI}} = \frac{(k-1) \cdot m^k + \sum_{i=0}^{k-1} m^{i+1}}{\sum_{i=0}^{k-1} m^{i+1}} \quad (5.34)$$

Let v_i be a node in the i^{th} level of PKI/NPKI and c_{v_i} be the number of classical certificates issued by v_i, which is equal to m for $\forall i \leq k-1$, equal to 0 for $i = k$. $NCPO_{v_i}$ is the nested certificate propagation overhead for v_i. $NCPO_{v_i}$ is formulated as the following.

$$NCPO_{v_i} = \frac{n_{v_i} + c_{v_i}}{c_{v_i}} = \begin{cases} \dfrac{m^{k-i} + m}{m} = m^{k-i-1} + 1 & i < k-1 \\ 1 & i = k-1 \\ \text{n/a} & i = k \end{cases} \qquad (5.35)$$

As can be seen from this equation, for the end users, nested certificate propagation overhead is not applicable, since they do not issue certificates of any type. For the nodes in one upper level of the leaf nodes (level *k*-1), the nested certificate propagation overhead is 1. That is, there is no nested certificate propagation overhead, since they issue classical certificates, but do not issue nested certificates. For the upper levels, the nested certificate propagation overhead increases as the level number, *i*, decreases.

Nested certificate propagation overhead is actually the disadvantage of the system. In order to compare this disadvantage with the advantage of efficient nested certificate path verification, the average nested certificate path length and the average number of nested certificates on the nested certificate paths of NPKI must also be formulated. There is one path to each end user from each level. Therefore, there are *k* paths towards each end user. Moreover, the length of each path is only related to the level number. That is why the average nested certificate path length and the number of nested certificates values are solely dependant on *k*. The average nested certificate path length, which is denoted as pl_{NPKI}, is formulated as the following.

$$pl_{NPKI} = \frac{\sum_{i=0}^{k-1} k - i}{k} = \frac{\dfrac{k \cdot (k+1)}{2}}{k} = \frac{k+1}{2} \qquad (5.36)$$

One certificate of a nested certificate path is a classical certificate. Therefore, the number of nested certificates on the average length nested certificate path is one less than

pl_{NPKI} . The number of nested certificates on the average length nested certificate, which is denoted as npl_{NPKI} , is then formulated as follows.

$$npl_{NPKI} = pl_{NPKI} - 1 = \frac{k+1}{2} - 1 = \frac{k-1}{2} \tag{5.37}$$

In order to analyze the trade-off between the nested certificate propagation overhead and the nested certificate path verification improvement for a specific topology, one should calculate the overhead values, which are formulated by Equations 5.34 and 5.35, and calculate the npl_{NPKI} value, which is formulated by Equation 5.37. The overhead values are self-explanatory. However, in order to analyze the nested certificate path verification efficiency improvement, the analysis given in Section 5.2.3 must also be used together with the npl_{NPKI} value. An example of such an analysis is given in the next subsection.

5.4.3. A Numerical Example

In order to give an idea about the dimensions of the trade-off between the nested certificate path verification improvement and nested certificate propagation overhead, a numerical example will be given in this subsection. The example PKI topology will be a 4-level 10-ary balanced tree. The formulations given in Section 5.4.2 will be used here by substituting k=4 and m=10.

The $NCPO_{PKI}$ value for the example topology is calculated by using Equation 5.34 as the following.

$$NCPO_{PKI} = \frac{(k-1)\cdot m^{k} + \sum_{i=0}^{k-1} m^{i+1}}{\sum_{i=0}^{k-1} m^{i+1}} = \frac{(4-1)\cdot 10^{4} + \sum_{i=0}^{4-1} 10^{i+1}}{\sum_{i=0}^{4-1} 10^{i+1}} = \frac{41110}{11110} = 3.7 \qquad (5.38)$$

That means, the total number of nested certificates increases 3.7 times by the nested certificate propagation towards end user method. However, this overhead is not equally distributed among the nodes in the PKI/NPKI. The upper level nodes encounter more nested certification overhead. The overhead for each node in the same level is the same. Therefore, it is enough to analyze the level based overheads, in order to give the idea about the node-by-node nested certificate propagation overheads. The nested certificate propagation overheads, the number of issued classical certificates and the number of issued nested certificates for the nodes at each level are given in Table 5.6 together with the number of nodes at that level. Equations 5.30, 5.31 and 5.35 are used for these calculations.

Table 5.6. Level based nested certificate propagation values

level	**Number of nodes in the level**	**Number of classical certificates**	**Number of nested certificates**	**NCPO**
0	1	10	10000	1001
1	10	10	1000	101
2	100	10	100	11
3	1000	10	0	1
4	10000	0	0	n/a

The advantage of the nested certificate propagation method is the speedy nested certificate path verification. The measure of this improvement is the speed-up factor as discussed in Sections 5.1 and 5.2. The average speed-up factor in NPKI is related to the average number of nested certificates on the nested certificate paths. This value for the example tree is calculated by using Equation 5.37 as follows.

$$npl_{NPKI} = \frac{k-1}{2} = \frac{4-1}{2} = 1.5 \quad (5.39)$$

The improvement due to 1.5 average number of nested certificates can be looked up at Figure 5.7 of Section 5.2.3. According to this figure, the speed-up factor for the average 1.5 nested certificates is between 2.3 and 2.5 depending on the cryptosystems and the hash functions used. That is, the average path verification in NPKI is 2.3 to 2.5 times faster than the average path verification in pure classical PKI.

The trade-off in the nested certificate propagation method for a 4-level 10-ary balanced tree shaped PKI/NPKI topology is that by increasing the number of certificates 3.7 times, the average path verification becomes 2.3 to 2.5 times faster. Although the certification overhead is bigger than the efficiency improvement, this trade-off is still acceptable, since the certificate is issued only once, but the verification can be performed several times.

An important criticism of the nested certificate propagation method is the non-uniformity of the nested certificate issuance overhead. Indeed, there is a significant nested certificate issuance overhead on the upper level CAs of the NPKI, as can be seen at Table 5.6. In classical PKIs, the upper level CAs need not take any action, when a new end user is added to the PKI or the certificate for an end user is updated. However, in the nested certificate propagation method, when a new end user is added to the NPKI or the certificate for an end user is updated, one new nested certificate must be issued at each level. These characteristics can be considered as the disadvantages of the nested certificate propagation method. However, there are motivations to use this method that are listed below.

1. Assuming that each CA/NCA holds the certificates that it has issued, the worst case (for the top level CA/NCA) storage requirement for the nested certificates is in the order of 10 Mbytes, which is quite acceptable.

2. The nested certificates are issued once, but they are used to verify nested certificate paths several times. Therefore, the overhead is once, but the gain several times.

3. The NPKI is built over the PKI and the classical certificates of the PKI are still existing in the NPKI. Therefore, there are always the classical backups of the nested certificate paths in NPKI. Consequently, nested certificate propagation can be considered as an *off-line* process. The authorities may issue nested certificates at the idle times. During the initial set up time or when a new end user certificate is issued or updated, the verifiers may use the classical certificates until the nested certificate propagation is completed.

4. Actually, the classical CA servers are mostly idle and their utilization is small, since their classical certification loads are not so significant and they must be dedicated servers because of security reasons. By the nested certificate propagation method, their utilization increases.

5. Although the nested certificate issuance overhead seems to be significant, the example NPKI can be set up in 3-5 hours without interfering the normal PKI operations. Such a set up time is acceptable. This set up time tends to increase for bigger PKIs and the nested certificate propagation becomes useless. For example, the set up time for a 4-level 20-ary balanced tree shaped PKI is 2-3 days. For such large PKIs, the bottleneck is mostly due to the top level CA, since it must issue lots of nested certificates itself. For these cases, nested certificate propagation can be considered for sub hierarchies. That means, the top level CA does not issue nested certificates, but the CAs of its sub hierarchies do. This approach can be recursive such that if the sub hierarchies are also big, the nested certificate propagation is applied for its sub hierarchies.

6. CONCLUSIONS

In this thesis, a new certification scheme, *nested certification*, is proposed. Nested certification is an alternative certification scheme that allows flexibility in both certificate issuance and certificate verification. A *nested certificate* is a certificate to certify another certificate. The certificate that is certified via a nested certificate is called as *subject certificate*. Subject certificates can be either classical or other nested certificates. A nested certificate gives restricted assurance and it can only be issued to certify a single certificate. Moreover, the authorities need not trust the subject certificate issuers. Therefore, the certificate authorities may prefer to issue nested certificates instead of classical certificates, where there is limited trust information. In this way, some extra certification relationships can be constructed and these extra relationships allow additional certificate paths for the verifiers.

Verification of a certificate as the subject certificate of a nested certificate is named as *subject certificate verification*. The subject certificate verification method is also described in this thesis. Moreover, it has been proven that verification of a certificate as a subject certificate has the same confidence as the classical cryptographic verification of the same certificate.

The most important advantage of the nested certification scheme is the efficiency of the subject certificate verification method. Verification of a subject certificate via a nested certificate does not require public key cryptosystem operations. It only requires hash computation and comparisons. Therefore, compared to the cryptographic certificate verification method, subject certificate verification method is more efficient. This fact has been proven analytically. Moreover, the analytical formulation of the relative speed-up factor of the subject certificate verification time over the cryptographic certificate verification time has been derived and this speed-up factor is graphically analyzed. The factors that effect the speed-up factor are the certificate sizes, hash algorithms and the public key cryptosystems employed in the verifications. The analyses have been done for different combinations of RSA [12], DSA [32] cryptosystems and MD5 [15], SHA-1 [16]

hash algorithms. Moreover, different certificate sizes of up to 6000 bits were used. It has been observed that, for the cases considered, the speed-up factor range is between 8 and 3000. On the other hand, the typical certificate sizes are 3000 to 4500 bits. The speed-up factors for the typical certificate sizes are between 10 and 1220, depending on the algorithms used. The speed-up factor tends to decrease as the certificate size increases. Moreover, simulation studies have been performed for different cryptosystem and hash function combinations, but for 3000-bit certificate sizes. The simulation results have shown that, the subject certificate verification method is 13.3 to 1587.4 times faster than the cryptographic certificate verification method depending on the cryptosystems and the hash functions used. The simulation results are 5 – 20 per cent less than the corresponding analytical results, because of implementation overheads.

In this thesis, the design of a X.509 based PKI, which incorporates both classical and nested certificates, has also been presented. Such a PKI is called *Nested certificate based PKI* (*NPKI*). It is shown that the usage of nested certificates in NPKI does not cause significant incompatibilities with the standard X.509 certificate structure. Therefore, embedding the nested certificates into the X.509 certificate system would not be so difficult.

Two construction models in NPKI are proposed. The first one is called the *free certification* model. This model allows organic growth and it is the natural way to construct NPKI from zero. In this model, if a CA wants to give less assurance or if the trust information is not sufficient, then the CA may prefer to issue a nested certificate instead of a classical one. There is no enforcement here. The certification authorities of the NPKI are more flexible than the ones of other PKIs, because they have the alternative of issuing a nested certificate for the cases where they cannot issue a classical certificate. Such flexibility is the primary advantage of this model.

Another advantage of the free certification model is the efficiency of certificate path verification. The certificate paths obtained in this model are *NPKI certificate paths*. In NPKI certificate paths, the nested certificates are used together with classical certificates.

The total number of subject certificate verifications in a NPKI certificate path is equal to the number of nested certificates on the path. On the other hand, all of the certificates of a classical certificate path are verified cryptographically. Therefore, the usage of nested certificates in the NPKI certificate paths always improves the efficiency of the path verification as compared to the classical certificate paths of the same length. This fact has been proven analytically. Moreover, analytical formulation of the speed-up factor for the NPKI certificate path verification time over the classical certificate path verification time has been derived and this speed-up factor has been analyzed. This analysis has shown that the speed-up factor is directly related to the number of nested certificates on the NPKI certificate path and inversely related to the average certificate size of the path. Moreover, the speed-up factors for both subject certificate verification and NPKI certificate path verification methods are compared analytically. It has been concluded that the speed-up factor for the NPKI certificate path verification is always less than the speed-up factor for the corresponding subject certificate verification. The reason for this fact is the effect of the cryptographic verification(s) in the NPKI certificate path. Moreover, simulation studies were performed to analyze the behavior of the speed-up factor under different cryptosystems and hash functions. In the simulations, certificate paths with five certificates were used. The number of nested certificates among these five certificates is different in each run. Moreover, different cryptosystem – hash function pairs were used in the simulations. These analyses have shown that NPKI certificate path verification method is 1.22 to 4.93 times faster than the classical certificate path verification method as far as the example certificate paths and algorithms are considered. Moreover, it has been shown that the verification of a classical certificate via a NPKI certificate path has the same confidence as the cryptographic verification of it.

The second NPKI construction model is called *transition from existing PKI*. This model forces the CAs to issue nested certificates to the certificates that are issued by their neighbor CAs in the PKI. The outcome of this model is having a *nested certificate path* for each classical certificate path towards the end users in the PKI. Nested certificate path is a special case of NPKI certificate path, on which all of the certificates are nested certificates except the last one. The last certificate of a nested certificate path is a classical certificate, since it is used to certify the target entity of the path. The speed-up factor for the nested

certificate path verification is formulated and examined analytically. This analysis shows that the speed-up factor asymptotically increases as the path length increases. The speed-up factor approaches to the corresponding subject certificate verification speed-up factor, as the number of certificates approaches to infinity. Although nested certificate path verification performs better for long paths, it is not so practical to obtain such long certificate paths from NPKI. Practical results are obtained from the analyses of the nested certificate paths with 1 to 8 nested certificates and by using different cryptosystems and hash functions. These analyses are done both analytically and by simulation. Simulation results have shown that, for the cases considered, nested certificate path verification performs 1.87 to 8.83 times faster than the classical certificate path verification. Analytical results show similar characteristics.

Another important advantage of the *transition from an existing PKI* model is that the trust structure of PKI is not spoiled by nested certification. On the other hand, the average verification time is improved significantly as discussed above. However, to attain such improvement, numerous nested certificates must be issued. That means, there is a trade-off between the verification improvement and the nested certificate issuance overhead. This trade-off is also analyzed. A generic balanced tree PKI model is used for this analysis. The number of nested certificates is formulated for the nested certification overhead. Average nested certificate path length is formulated in order to look up the corresponding speed-up factor. It has been observed that, for a 4-level, 10-ary balanced tree shaped PKI, the average path verification speed-up factor is between 2.3 and 2.5, depending on the cryptosystems and the hash algorithms used. However, the number of total certificates increases by 3.7 times. The most important overhead here is the fact that this certification overhead is not distributed uniformly among the authorities. Upper level authorities encounter more nested certification overhead than the lower level ones. However, these overheads are tolerable in order to improve the path verification time.

In all of the speed-up factor analyses, it has been observed that the speed-up factors are larger for slower public key cryptosystems. The reason for this behavior is the fact that the public key cryptosystem operations are not used in the subject certificate verification

method. In this way, the numerators of the speed-up factors become larger for slower cryptosystems. The hash algorithms also effect the speed-up factor, but they are not as effective as the public key cryptosystems. Contrary to public key cryptosystems, the speed-up factors and the speed of the hash functions are directly related, since the hash operations are performed in both subject and cryptographic certificate verification methods.

To summarize, in this thesis a new nested signature based certification scheme, *nested certification*, is proposed and its applications in PKIs are examined. Moreover, analytical and simulation based performance analyses are performed, in order to show their efficiency improvement in the verification processes.

6.1. Future Work

The nested certificates can be adopted into X.509 standard as discussed in Section 4.8. However, a more detailed and formal design must be done in order to carry out this adoption. This design can be realized as a possible future work.

As another direction for research, the design of NPKI can be extended to detail nested certificate issuance policies. These policies can be developed for both *free certification* and *transition from existing PKI* models. Moreover, NPKI is general. As another further research area, the concepts of NPKI can be adopted into specific applications (like E-mail, electronic commerce, electronic payment, etc.) to use nested certificates in their PKIs.

Nested certificate revocation has not been detailed in this thesis, since it is strongly believed that this is wide area of research and does not fit into this thesis. However, the intuition is that nested certificates need not be revoked, since they do not directly certify a public key. When a classical certificate, which is certified via a chain of nested certificates, is revoked, these nested certificates automatically become useless. However, these issues

must be analyzed in more detail. Moreover, the use of nested certificates for classical certificate revocation list signing can also be a future research area.

APPENDIX A: LIST OF ACRONYMS

AH	Authentication Header
CA	Certification Authority
CDC	Certificate Distribution Center
CMS	Certificate Management System
CRL	Certificate Revocation List
DAP	Directory Access Protocol
DASS	Distributed Authentication Security Service
DES	Data Encryption Standard
DNS	Domain Name System
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ESP	Encapsulated Security Payload
HTTP	HyperText Transfer Protocol
IBIP	Information-Based Indicia Program
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
ISAKMP	Internet Security Association and Key Management Protocol
ISO	International Standards Organization
ITU	International Telecommunications Union
ITU	International Telecommunications Union
LDAP	Lightweight Directory Access Protocol
MCRL	Minimal CRL

MD4	Message Digest 4
MD5	Message Digest 5
NCA	Nested Certification Authority
NIST	National Information Standards Institute
NNTP	Network News Transfer Protocol
NPKI	Nested certificate based PKI
PEM	Privacy Enhanced Mail
PGP	Pretty Good Privacy
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructures
PKIX	Public Key Infrastructure for X.509 certificates
RFC	Request for Comments
RSA	Rivest, Shamir, Adleman,
S/MIME	Secure/Multipurpose Internet Mail Extensions
SAProxies	Self Authentication Proxies
SDSI	Simple Distributed Security Infrastructure
SET	Secure Electronic Transaction
SHA-1	Secure Hash Algorithm 1
S-HTTP	Secure HyperText Transfer Protocol
SPKI	Simple Public Key Infrastructure
SSL	Secure Socket Layer
UCA	User Certification Authorities
URL	Uniform Resource Locator
USPS	United States Postal Service
W3C	World Wide Web Consortium
WWW	World Wide Web

REFERENCES

1. Seberry, J. and J. Pieprzyk, *Cryptography: An Introduction to Computer Security*, Prentice-Hall, Sydney, 1989.

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

3. ITU-T Recommendation X.500, ISO/IEC 9594-1, Information Technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Services, 1997 Edition.

4. Zimmermann, P., *PGP User's Guide Volume I: Essential Topics*, available with free PGP software from http://www.pgpi.com/download, 1994.

5. Zimmermann, P., *PGP User's Guide Volume II: Special Topics*, available with free PGP software from http://www.pgpi.com/download, 1994.

6. National Institute of Standards and Technology (NIST), Data Encryption Standard (DES), Federal Information Processing Standard (FIPS) PUB 46, Washington D.C., 1977.

7. Lai, X., and J. Massey, “A Proposal for a New Block Encryption Standard,” *Proceedings of EuroCrypt 90*, May 1990, Aarhus, Denmark, Lecture Notes in Computer Science, vol. 473, pp. 389-404, Springer-Verlag, Berlin, 1991.

8. Lai, X., J. Massey and S. Murphy, “Markov Cycles and Differential Cryptanalysis,” *Proceedings of EuroCrypt 91*, April 1991, Brighton, UK, Lecture Notes in Computer Science, vol. 547, pp. 17-38, Springer-Verlag, Berlin, 1991.

9. Baldwin, R., and R. Rivest, The RC5, RC5-CBC, RC5-CBC-Pad and RC5-CTS Algorithms, RFC 2040, October 1996.

10. Rivest, R., A Description of the RC2 Encryption Algorithm, RFC 2268, March 1998.

11. Diffie, W., and M. E. Hellman, “New Directions in Cryptography,” *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644-654, November 1976.

12. Rivest, R., A. Shamir and L. Adleman, “A Method for Obtaining Digital Signatures and Public Key Cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, February 1978.

13. Kaufman, C., R. Perlman and M. Speciner, *Network Security – PRIVATE Communication in a PUBLIC World*, Chapter 4, pp. 101-127, Prentice Hall, New Jersey, 1995.

14. Rivest, R., The MD4 Message Digest Algorithm, RFC 1320, April 1992.

15. Rivest, R., The MD5 Message Digest Algorithm, RFC 1321, April 1992.

16. National Institute of Standards and Technology (NIST), Secure Hash Standard (SHS), Federal Information Processing Standard (FIPS) PUB 180 –1, U.S. Department of Commerce, Washington, 17 April 1995.

17. Gong, L., “Variations on Themes of Message Freshness and Replay,” *Proceedings of the IEEE Computer Security Foundations Workshop VI*, Franconia, New Hampshire, June 1993, pp 131-136, IEEE Computer Society Press.

18. Needham, R. M. and M. D. Schroeder, “Using Encryption for Authentication in Large Networks of Computers,” *Communications of the ACM*, vol. 21, no. 12, pp. 993-999, December 1978.

19. Denning, D. E. and G. M. Sacco, “Timestamps in Key Distribution Protocols,” *Communications of the ACM*, vol. 24, no. 8, pp. 533-536, August 1981.

20. Steiner, J. G., C. Neuman and J. I. Schiller, “Kerberos: An Authentication Service for Open Network Systems,” *Proceedings of Winter USENIX Technical Conference*, Dallas, February 1988, pp. 191-202.

21. Neuman, C. and T. Ts’o, “Kerberos: An Authentication Service for Computer Networks,” *IEEE Communications Magazine*, vol. 32, no. 9, pp 33-38, September 1994.

22. Woo, T. Y. C. and S. S. Lam, “Authentication for Distributed Systems,” *IEEE Computer*, vol. 25, no. 1, pp. 39-51, January 1992.

23. Open Software Foundation, *Introduction to OSF DCE*, Prentice Hall, New Jersey, 1992.

24. Bird, R., I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva and M. Yung, “The KryptoKnight Family of Light-Weight Protocols for Authentication and Key

Distribution," *IEEE/ACM Transactions on Networking*, vol. 3, no. 1, pp. 31-41, February 1995.

25. Needham, R. M. and M. D. Schroeder, "Authentication Revisited," *ACM Operating System Review*, vol. 21, no. 1, pp. 7, January 1987.

26. Popek, G. and C. Kline, "Encryption and Secure Computer Networks," *ACM Computing Surveys*, vol. 11, no. 4, pp. 331-356, December 1979.

27. Denning, D. E., "Protecting Public Keys and Signature Keys," *IEEE Computer*, vol. 16, no. 2, pp. 27-35, February 1983.

28. Kaufman, C., DASS - Distributed Authentication Security Service, RFC 1507, September 1993.

29. Tardo, J. J. and K. Alagappan, "SPX: Global Authentication using Public Key Certificates," *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, May 1991, pp. 232-244.

30. Fumy, W. and P. Landrock, "Principles of Key Management," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 785-793, June 1993.

31. Akl, S. G., "Digital Signatures: A Tutorial Survey," *IEEE Computer*, vol. 16, no. 2, pp. 15-24, February 1983.

32. National Institute of Standards and Technology (NIST), Federal Information Processing Standard (FIPS) PUB 186, Digital Signature Standard (DSS), U.S. Department of Commerce, 19 May 1994.

33. El Gamal, T., "A Public Key Cryptosystems and a Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, no. 4, pp. 469-472, July 1985.

34. Chadwick, D. W., 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.

35. Stallings, W., *Network and Internetwork Security*, Prentice Hall, New Jersey, 1995.

36. Ellison, C. M., "What do you need to know about the person with whom you are doing business", 1997, http://www.clark.net/pub/cme/html/congress1.html

37. ITU-T Recommendation X.509, ISO/IEC 9594-8, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, 1988 Edition.

38. I'anson, C. and C. Mitchell, "Security Defects in CCITT Recommendation X.5O9 - The Directory Authentication Framework," *ACM SIGCOMM Computer Communication Review*, vol. 20, no. 2, April 1990.

39. Burrows, M., M. Abadi and R. M. Needham, "A Logic for Authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18-36, February 1990.

40. ITU-T Recommendation X.509, ISO/IEC 9594-8, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, 1993 Edition.

41. RSA Laboratories, PKCS #6: Extended Certificate Syntax Standard, version 1.5, RSA Laboratories Technical Note, available from http://www.rsa.com/rsalabs/pub/PKCS, November 1, 1993.

42. RSA Laboratories, PKCS #7: Cryptographic Message Syntax Standard, version 1.5, RSA Laboratories Technical Note, available from http://www.rsa.com/rsalabs/pub/PKCS, November 1, 1993.

43. Kaliski, B. S. and K. W. Kingdon, Extensions and Revisions to PKCS #7, RSA Laboratories Technical Note, available from http://www.rsa.com/rsalabs/pub/PKCS, May 13, 1997.

44. RSA Laboratories, PKCS #10: Certification Request Syntax Standard, version 1.0, RSA Laboratories Technical Note, available from http://www.rsa.com/rsalabs/pub/PKCS, November 1, 1993.

45. Wahl, M., T. Howes and S. Kille, Lightweight Directory Access Protocol (v3), RFC 2251, December 1997.

46. Kent, S. T., “Internet Privacy Enhanced Mail,” *Communications of the ACM*, vol. 36, no. 8, pp. 48-60, August 1993.

47. MasterCard Inc., SET Secure Electronic Transaction Specification Book 1: Business Description, MasterCard Inc., May 1997.

48. United States Postal Service, Performance Criteria for Information-based Indicia and Security Architecture for IBI Postage Metering Systems, August 1998, available from http://www.usps.gov/ibip/documents/specs/pc0819.pdf

49. Housley, R., W. Ford, W. Polk and D. Solo, Internet Public Key Infrastructure: X.509 Certificate and CRL Profile, RFC 2459, March 1999.

50. Adams, C. and S. Farrell, Internet X.509 Public Key Infrastructure Certificate Management Protocols, RFC 2510, March 1999.

51. Ramsdell, B., S/MIME Version 3 Certificate Handling, work in progress, Internet Draft <draft-ietf-smime-cert-08.txt>, April 1999.

52. Chokhani, S., "Towards a National Public Key Infrastructure," *IEEE Communications Magazine,* vol. 32, no. 9, pp 70-74, September 1994.

53. Levi, A. and M. U. Çağlayan, "Türkiye için bir Açık Anahtar Altyapısı Modeli," *Bilişim 98 - TBD 15. Bilişim Kurultayı Bildiriler Kitabı*, Istanbul, pp. 354-361, September 1998.

54. Levi, A. and M. U. Çağlayan, "Elektronik Posta Güvenliği ve Açık Anahtar Sunucuları," *Bilişim 97 - TBD 14. Bilişim Kurultayı*, Istanbul, pp. 114-117, September 1997.

55. Reiter, M. K. and S. G. Stubblebine, "Path Independence for Authentication in Large-Scale Systems," *Proceedings of the Fourth ACM Conference on Computer and Communications Security*, pp. 57-66, April 1997.

56. Kapidzic, N., “Creating Security Applications Based on the Global Certificate Management System,” *Computers and Security*, vol. 17, no. 6, pp. 507-515, 1998.

57. Crispo, B. and M. Lomas, “A Certification Scheme for Electronic Commerce,” *Proceedings of Security Protocols International Workshop*, Cambridge UK, April 1996, Lecture Notes in Computer Science, vol. 1189, pp. 19-32, Springer-Verlag, Berlin, 1997.

58. Ellison, C. M., “Uses of Certificates”, 1996, http://www.clark.net/pub/cme/html/cert-use.html

59. Blaze, M., J. Feigenbaum and J. Lacy, “Decentralized Trust Management,” *Proceedings of IEEE Conference on Security and Privacy*, May 1996.

60. Ellison, C. M., et. al., SPKI Certificate Theory, work in progress, Internet Draft < draft-ietf-spki-cert-theory-04.txt>, 17 November 1998.

61. Ellison, C. M., et. al., Simple Public Key Certificate, work in progress, Internet Draft <draft-ietf-spki-cert-structure-05.txt>, 13 March 1998.

62. Rivest, R. and B. Lampson, “SDSI – A Simple Distributed Security Infrastructure”, 1996, http://theory.lcs.mit.edu/~cis/sdsi.html

63. Mockapetris, P., Domain Names - Concepts and Facilities, RFC 1034, November 1987.

64. Eastlake, D. and C. Kaufman, Domain Name System Security Extensions, RFC 2065, January 1997.

65. Freier, A. O., P. Karlton and P. C. Kocher, The SSL Protocol Version 3, Netscape Communications Corp., 1996, available from http://home.netscape.com/eng/ssl3.

66. Rescorla, E. and A. Schiffman, The Secure HyperText Transfer Protocol, work in progress, Internet Draft <draft-ietf-wts-shttp-06.txt>, June 1998.

67. Kent, S. and R. Atkinson, IP Authentication Header, RFC 2402, November 1998.

68. Kent, S. and R. Atkinson, IP Encapsulated Security Payload (ESP), RFC 2406, November 1998.

69. Orman, H., The Oakley Key Determination Protocol, RFC 2412, November 1998.

70. Maughan, D., M. Schertler, M. Schneider and J. Turner, Internet Security Association and Key Management Protocol (ISAKMP), RFC 2408, November 1998.

71. Harkins, D. and D. Carrel, The Internet Key Exchange (IKE), RFC 2409, November 1998.

72. Simpson, W. A., Photuris: Secret Exchange, work in progress, Internet Draft <draft-simpson-photuris-secret-01.txt>, March 1999.

73. Neumann, B. C., “Security, Payment, and Privacy for Network Commerce,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1523 – 1531, October 1995.

74. Schöter, A. and R. Willmer, Digital Money Online: A Review of Existing Technologies, Intertrader Ltd., February 1997, available from http://www.intertrader.com/library/DigitalMoneyOnline/dmo/dmo.htm

75. Micali, S., Certificate Revocation System, United States Patents 5,793,868, 1998.

76. Low, M. R. and B. Christianson, "Self Authenticating Proxies," *The Computer Journal*, vol. 37, no. 5, pp. 422-428, 1994.

77. Low, M. R. and B. Christianson, "Technique for Authentication, Access Control and Resource Management in Open Distributed Systems," *IEE Electronics Letters*, vol. 30, no. 2, pp. 124-125, January 1994.

78. Chu, Y., P. DesAutels, B. LaMacchia and P. Lipp, "PICS Signed Labels (DSig) Specification", May 1998, http://www.w3.org/TR/REC-Dsig-label

79. Hoffman, P., Enhanced Security Services for S/MIME, work in progress, Internet Draft <draft-ietf-smime-ess-12.txt>, March 1999.

80. Bosselaers, A., R. Govaerts and J. Vandewalle, "Fast Hashing on the Pentium," *Advances in Cryptology, Proceedings of Crypto'96*, August 1996, Santa Barbara, California, Lecture Notes in Computer Science, vol. 1109, pp. 298-312, Springer-Verlag, 1996.

81. Bosselaers, A., "Even Faster Hashing on the Pentium," *Rump Session of Eurocrypt '97*, Germany, May 1997, ftp://ftp.esat.kuleuven.ac.be/COSIC/bosselae/pentiumplus.ps.gz

82. RSA Data Security Inc. Engineering Report, “JSAFE 1.1 Benchmarks”, 1998, http://www.rsa.com/rsa/products/jsafe/html/benchmarks.html

83. Touch, J. D., “Performance Analysis of MD5,” *Proceedings of the ACM SIGCOMM’95, Computer Communication Review*, vol. 25, no. 4, pp. 77-86, October 1995.

84. Wiener, M. J., “Performance Comparison of Public-Key Cryptosystems,” *RSA Laboratories’ Cryptobytes*, vol. 4, no. 1, pp. 1 – 5, 1998.

85. Reiter, M. K. and S. G. Stubblebine, “Toward Acceptable Metrics of Authentication,” *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pp 10 –20, May 1997.

86. GMD Security Technology, “SECUDE – A General Purpose Security Toolkit”, 1998, http://www.darmstadt.gmd.de/secude

REFERENCES NOT CITED

Anderson, R. and R. Needham, "Robustness Principles for Public Key Protocols," *Proceedings of Crypto 95*, Springer-Verlag, pp. 236 – 247.

Atkinson, R. J., "Toward a More Secure Internet," *IEEE Computer*, vol. 30, no. 1, pp. 57-61, January 1997.

Balenson, D. M., "Automated Distribution of Cryptographic Keys Using the Financial Institution Key Management Standard," *IEEE Communications Magazine*, vol. 23, no. 9, pp. 41-46, September 1985.

Bhimani, A., "Securing the Commercial Internet," *Communications of the ACM*, vol. 39, no. 6, pp. 29-35, June 1996.

Borcherding, B. and M. Borcherding, "Efficient and Trustworthy Key Distribution in Webs of Trust," *Computers and Security*, vol. 17, no. 5, pp. 447-454, 1998.

Çağlayan, M. U., F. Kadifeli and A. C. C. Say, *Standard C Programming*, Boğaziçi University Press, Istanbul, 1989.

Gong, L., M. A. Lomas, R. M. Needham and J. H. Saltzer, "Protecting Poorly Chosen Secrets from Guessing Attacks," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 648-656, June 1993.

Gong, L., R. M. Needham and R. Yahalom, "Reasoning about Belief in Cryptographic Protocols," *Proceedings of the IEEE 1990 Symposium on Security and Privacy*, Oakland, California, May 1990, pp. 234-248.

Harn, L. and S. Yang, "ID-Based Cryptographic Schemes for User Identification, Digital Signatures, and Key Distribution," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 757-760, June 1993.

Lampson, B., M. Abadi, M. Burrows and E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *ACM Transactions on Computer Systems*, vol. 10, no. 4, pp. 265 – 310, Nov. 1992.

Levi, A. and M. U. Çağlayan, "A Multiple Signature Based Certificate Verification Scheme," *Proceedings of BAS'98, The Third Symposium on Computer Networks*, 25-26 June 1998, İzmir, Turkey, pp. 1 –10.

Levi, A. and M. U. Çağlayan, "Analytical Performance Evaluation of Nested Certificates," accepted to *IFIP WG 7.3 Performance '99 Conference*, will also appear in *Performance Evaluation Journal*, 1999.

Levi, A. and M. U. Çağlayan, "Integrity Control in Nested Certificates," *Proceedings of BAS'99, The Fourth Symposium on Computer Networks*, 20-21 May 1999, Istanbul, Turkey, pp. 149-157.

Levi, A. and M. U. Çağlayan, Nested Certificates and Their Applications in Public Key Infrastructures, Technical Report FBE/CmpE-02/98-13, Boğaziçi University, 1998.

Levi, A. and M. U. Çağlayan, "NPKI: Nested Certificate Based Public Key Infrastructure," *Advances in Computer and Information Sciences '98 - Proceedings of the Thirteenth International Symposium on Computer and Information Sciences - ISCIS XIII*, IOS Press, Concurrent Systems Engineering Series, vol. 53, pp. 397 – 404, October 1998, Turkey.

Lin, P. and L. Lin, "Security in Enterprise Networking: A Quick Tour," *IEEE Communications Magazine*, vol. 34, no. 1, pp. 56-61, January 1996.

MathWorks Inc., *Matlab User's Guide*, 1992.