DESIGN AND DEVELOPMENT OF

PRACTICAL AND SECURE E-MAIL SYSTEM

by

Mahmut Özcan

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

February 2003

DESIGN AND DEVELOPMENT OF

PRACTICAL AND SECURE E-MAIL SYSTEM

APPROVED BY:

Asst. Prof. Dr. Albert Levi .....
(Thesis Supervisor)

Asst. Prof. Dr. Erkay Savaş .....

Asst. Prof. Dr. Gürdal Ertek .....

DATE OF APPROVAL: .....

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to Dr. Albert Levi who supervised me during the thesis period. It was very important for me to get his helps at every phases of my thesis even in the completion of my thesis report. I can say that this thesis is mostly the product of his encouragement and motivations.

I am also grateful to Dr. Erkay Savaş and Dr. Gürdal Ertek for their participation in my thesis jury, careful reviews and comments. Besides, Dr. Erkay Savaş's encouragements and permission to make subpart of my thesis as a term project in "Introduction to Cryptography" course was a great support for my thesis.

Finally, I want to express my special thanks to my dear father for his motivation to my academic career. His un-limited advises lead me to academic life and surpassed all my other goals.

To my dear father…

ABSTRACT

Key distribution and management in applications that use public key cryptosystems generally rely on Public Key Infrastructures (PKI). In this thesis, the disadvantages of this approach are discussed and an e-mail system that performs public key distribution and management in a unique way is proposed. The name of this system is "Practical and Secure E-Mail System" ("PractiSES").

PractiSES does not use the certification mechanisms of PKIs. A central authority, which is trusted by all users, takes the responsibility of key distribution and management in PractiSES. PractiSES Client is an e-mail application that is designed for end users. On top of regular e-mail client features, PractiSES Client can also be used to exchange e-mails among users in encrypted and/or signed fashion.

PractiSES is designed according to the phases of "Object Oriented Analyses and Design (OOAD)". It is implemented using Java programming language. In PractiSES, there are several secure protocols developed for initializing users, removing and updating public keys of the users and obtaining the others' public keys. Key management and distribution features of PractiSES do not let the e-mail addresses move around in an uncontrolled fashion - this is one of the problems of PKI based systems. Moreover, certificate revocation problem does not exist in PractiSES. The trust mechanism of PractiSES is simple and straightforward so that an average user can easily use. Those characteristics of PractiSES make it "practical". On the other hand, PractiSES supports enough security features, such as authentic registration, encryption and digital signatures.

The first version of PractiSES will be for closed-group e-mail exchange. PractiSES will be a free application that can be used without any warranty by companies and universities.

ÖZET

Açık anahtar tabanlı kriptografi algoritmalarını kullanan sistemler anahtar dağıtım ve yönetim işlerini genellikle açık anahtar altyapıları (PKI) ile yaparlar. Bu tezde bu yaklaşımın sorunlarından bahsedilmiştir ve anahtar dağıtım ve yönetimini kendine özgü bir şekilde yapan bir e-posta sistemi önerilmiştir. Bu e-posta sistemi "Pratik ve Güvenli E-posta" ("PGE") olarak adlandırılmıştır.

PGE açık anahtar altyapılarından farklı olarak anahtar dağıtımında sertifikalandırma yöntemini kullanmaz. PGE sisteminde bütün kullanıcıların güvendiği merkezi bir otorite (sunucu) anahtar dağıtımını üstlenir. Anahtar yönetimi de bu otorite tarafından yerine getirilir. PGE sisteminin kullanıcılarına bakan kısmı kullanıcı e-posta programlarıdır. Bu programlar normal bir e-posta programının özelliklerine sahiptir ve kullanıcılarının açık anahtar tabanlı kriptografi algoritmalarını kullanarak, kendi aralarında şifreli ve/veya imzalı e-posta göndermelerini de sağlar.

PGE nesneye dayalı analiz ve tasarım (OOAD) aşamalarına uyularak gerçeklenmiştir. PGE'nin gerçeklenmesinde Java programlama dili kullanılmıştır. PGE sisteminde, son kullanıcıların kendi açık anahtarlarını açık anahtar deposuna koyabilmeleri, depodan silebilmeleri, yenileyebilmeleri ve başka kullanıcıların açık anahtarlarını depodan alabilmeleri için güvenli protokoller tasarlanmıştır. PGE sisteminin anahtar dağıtım ve yönetim mekanizması, PKI tabanlı sistemlerde olduğu gibi kullanıcıların e-posta adreslerinin kontrolsüz dolaşımına izin vermez. PGE'de sertifika iptali ve onun getirdiği problemlere rastlanmaz. PGE'nin güven mekanizması ortalama kullanıcıların kolayca kullanabilecekleri kadar basit ve düzgündür. Bütün bu özellikler PGE sistemini "pratik" yapmaktadır. PGE şifreleme, imzalama ve doğrulayarak kayıt yapma gibi özellikleri desteklediği için yeterince "güvenli" bir e-posta sistemidir.

PGE sisteminin ilk sürümü organizasyon içi e-posta değişimini sağlayacaktır. PGE uygulaması, şirket ve üniversite gibi kuruluşların hiç bir ücret ödemeden kullanmalarına izin vermektedir.

TABLE OF CONTENTS

Page

LIST OF FIGURES

DESIGN AND DEVELOPMENT OF

PRACTICAL AND SECURE E-MAIL SYSTEM

by

Mahmut Özcan

Submitted to the Graduate School of Engineering and Natural Sciences

in partial fulfillment of

the requirements for the degree of

Master of Science

Sabancı University

February 2003

DESIGN AND DEVELOPMENT OF

PRACTICAL AND SECURE E-MAIL SYSTEM

APPROVED BY:

Asst. Prof. Dr. Albert Levi .....
(Thesis Supervisor)

Asst. Prof. Dr. Erkay Savaş .....

Asst. Prof. Dr. Gürdal Ertek .....

DATE OF APPROVAL: .....

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to Dr. Albert Levi who supervised me during the thesis period. It was very important for me to get his helps at every phases of my thesis even in the completion of my thesis report. I can say that this thesis is mostly the product of his encouragement and motivations.

I am also grateful to Dr. Erkay Savaş and Dr. Gürdal Ertek for their participation in my thesis jury, careful reviews and comments. Besides, Dr. Erkay Savaş's encouragements and permission to make subpart of my thesis as a term project in "Introduction to Cryptography" course was a great support for my thesis.

Finally, I want to express my special thanks to my dear father for his motivation to my academic career. His un-limited advises lead me to academic life and surpassed all my other goals.

To my dear father…

ABSTRACT

Key distribution and management in applications that use public key cryptosystems generally rely on Public Key Infrastructures (PKI). In this thesis, the disadvantages of this approach are discussed and an e-mail system that performs public key distribution and management in a unique way is proposed. The name of this system is "Practical and Secure E-Mail System" ("PractiSES").

PractiSES does not use the certification mechanisms of PKIs. A central authority, which is trusted by all users, takes the responsibility of key distribution and management in PractiSES. PractiSES Client is an e-mail application that is designed for end users. On top of regular e-mail client features, PractiSES Client can also be used to exchange e-mails among users in encrypted and/or signed fashion.

PractiSES is designed according to the phases of "Object Oriented Analyses and Design (OOAD)". It is implemented using Java programming language. In PractiSES, there are several secure protocols developed for initializing users, removing and updating public keys of the users and obtaining the others' public keys. Key management and distribution features of PractiSES do not let the e-mail addresses move around in an uncontrolled fashion - this is one of the problems of PKI based systems. Moreover, certificate revocation problem does not exist in PractiSES. The trust mechanism of PractiSES is simple and straightforward so that an average user can easily use. Those characteristics of PractiSES make it "practical". On the other hand, PractiSES supports enough security features, such as authentic registration, encryption and digital signatures.

The first version of PractiSES will be for closed-group e-mail exchange. PractiSES will be a free application that can be used without any warranty by companies and universities.

ÖZET

Açık anahtar tabanlı kriptografi algoritmalarını kullanan sistemler anahtar dağıtım ve yönetim işlerini genellikle açık anahtar altyapıları (PKI) ile yaparlar. Bu tezde bu yaklaşımın sorunlarından bahsedilmiştir ve anahtar dağıtım ve yönetimini kendine özgü bir şekilde yapan bir e-posta sistemi önerilmiştir. Bu e-posta sistemi "Pratik ve Güvenli E-posta" ("PGE") olarak adlandırılmıştır.

PGE açık anahtar altyapılarından farklı olarak anahtar dağıtımında sertifikalandırma yöntemini kullanmaz. PGE sisteminde bütün kullanıcıların güvendiği merkezi bir otorite (sunucu) anahtar dağıtımını üstlenir. Anahtar yönetimi de bu otorite tarafından yerine getirilir. PGE sisteminin kullanıcılarına bakan kısmı kullanıcı e-posta programlarıdır. Bu programlar normal bir e-posta programının özelliklerine sahiptir ve kullanıcılarının açık anahtar tabanlı kriptografi algoritmalarını kullanarak, kendi aralarında şifreli ve/veya imzalı e-posta göndermelerini de sağlar.

PGE nesneye dayalı analiz ve tasarım (OOAD) aşamalarına uyularak gerçeklenmiştir. PGE'nin gerçeklenmesinde Java programlama dili kullanılmıştır. PGE sisteminde, son kullanıcıların kendi açık anahtarlarını açık anahtar deposuna koyabilmeleri, depodan silebilmeleri, yenileyebilmeleri ve başka kullanıcıların açık anahtarlarını depodan alabilmeleri için güvenli protokoller tasarlanmıştır. PGE sisteminin anahtar dağıtım ve yönetim mekanizması, PKI tabanlı sistemlerde olduğu gibi kullanıcıların e-posta adreslerinin kontrolsüz dolaşımına izin vermez. PGE'de sertifika iptali ve onun getirdiği problemlere rastlanmaz. PGE'nin güven mekanizması ortalama kullanıcıların kolayca kullanabilecekleri kadar basit ve düzgündür. Bütün bu özellikler PGE sistemini "pratik" yapmaktadır. PGE şifreleme, imzalama ve doğrulayarak kayıt yapma gibi özellikleri desteklediği için yeterince "güvenli" bir e-posta sistemidir.

PGE sisteminin ilk sürümü organizasyon içi e-posta değişimini sağlayacaktır. PGE uygulaması, şirket ve üniversite gibi kuruluşların hiç bir ücret ödemeden kullanmalarına izin vermektedir.

TABLE OF CONTENTS

Page

LIST OF FIGURES

1. INTRODUCTION

Public key cryptosystems (PKC systems)[1] are used extensively in network security and authentication applications. In PKC systems, every user has a key pair that is created by its owner. The private key is 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, everyone can know the public keys. PKC systems propose strong security.

In order to use public key cryptography in an application, its key distribution and key management problems should be solved first. Problems related with the life cycle of public keys are generally named as key management problems. Public key removal, update and recovery are the most common key management operations. Distribution of public key with legitimate bindings with the owner's identity is the key distribution problem. In order to solve this problem, a trusted third party is inevitably necessary. Trusted third parties are the organizations that people agree on their trustworthiness. If the trusted third party states that a public key belongs to a particular user, then people who trust the trusted third party make sure about the legitimacy of that public key. Public Key Infrastructure (PKI) [2] and Pretty Good Privacy (PGP) [3] propose solutions to key distribution and management problems.

PKI is an architecture of the Certification Authorities (CAs) which produce certificates. Certificates are the digital documents that are used as binding between user identity with user's public key. Secure Multi-purpose Internet Mail Extension (S/MIME) [4] applications use digital certificates in which there is a public key used to encrypt the messages and verify the signatures in secure Internet messaging. S/MIME is a protocol that provides secure Internet message exchange between parties. S/MIME needs certificates issued by CAs of PKI. There are several problems related with certificates. Certificates can tour in the Internet and some information (e.g. e-mail

address) in certificates may be distributed to un-intended people. Privacy-sensitive people criticize this situation. Certificates are generally not free and there are some shortcomings in free ones. There are some difficulties in certifying people such as, registration obligation to Registration Authority (RA), and constructing/validating the certification path from X.500 Directories [5, 6].

PGP is itself an e-mail application in which trusting to someone else is left to user's own criteria. Since there is no trusted third party in PGP, its key server that keeps the users' public keys does not guarantee the legitimacy of the keys in it.

PKC systems are mostly used in e-mail applications. PGP and S/MIME have several problems and difficulties which are mentioned above. In this thesis, we propose a new e-mail system that is secure and more practical than PGP and S/MIME. Name of our system is "Practical and Secure E-Mail System" ("PractiSES"). PractiSES provides encrypted and/or signed message interchange between parties by using PKC systems as a base. The objective of PractiSES is to eliminate the difficulties and problems of S/MIME and PGP.

PractiSES defines trusted third parties (a server) for closed-groups. The server distributes the public keys using the server's signature. It provides a practical way for key distribution. Registered users can facilitate the security features of PractiSES just after initialization protocol. Initialization protocol uploads user's public key to server's public key storage. Users can remove and update their current public keys and learn others' public keys with secure protocols. Trusting a key is straightforward since it escapes use of certificates and defines a centralized trusted third party. However, S/MIME applications face with problems of certificates and PGP suffers from absence of a trusted third party and pays its cost by complicated trust mechanism.

PractiSES has a practical client Graphical User Interface (GUI) that has English and Turkish language support.

The structure of thesis is mentioned in the following paragraphs.

In Section 2, background information including fundamentals of cryptography, security problems and solutions, problems of PKC systems and solutions to them are given. Contribution of thesis is also summarized there.

Section 3 deals with design and development of PractiSES. The architecture, properties, and components of PractiSES are explained in this section. The advantages of PractiSES over S/MIME and PGP are also given.

In Section 4, cryptographic structures and functions that are used in PractiSES, are detailed. The requirements and deployment of PractiSES system are also described here.

Conclusion and future works that can be carried out to improve the current status of PractiSES are given in Section 5.

Section 6, 7, 8, and 9 are appendices. In Section 6 the acronyms are listed. Section 7 shows the security dialogs that are used in the system. The snapshots of sending/receiving e-mails in PractiSES Client GUI are figured out in Section 8. Section 9 gives the details of the possible e-mail message structures in PractiSES.

2. BACKGROUND INFORMATION

In today's world, keeping information away from adversaries plays an important role in our lives. Most of the people, who are communicating, had developed different types of encoding/decoding techniques while transmitting sensitive information. The number of connected people over the world, and consequently the demand for information services in electronic platforms, are constantly increasing. Many types of critical operations are performed in the electronic systems. Giving credit card number while purchasing on the Internet, banking transactions over the Internet and exchanging military messages are just some of the samples of critical operations.

2.1. Security Requirements

The data sent over computer networks are sensitive to attacks by adversaries. Private/sensitive information must be protected from others since the malicious adversaries can read and/or alter the message content or masquerade himself/herself as someone else. In order to make sure that the information is secure, four main requirements are considered. These are authentication, integrity, non-repudiation and confidentiality.

2.1.1. Authentication

Authentication [7] is a process of proving and verifying certain information in a communication. Verifying the origin of a document, identifying the sender and/or the receiver, identifying a specific hardware device (a computer, printer, etc.), and verifying the time that a document is sent are some of the examples of authentication processes. This service can be realized by several cryptographic operations.

2.1.2. Integrity

Data may be altered, inserted, deleted or misordered by an unauthorized adversary during communication. Integrity is a process of ensuring that the message is received as it is sent. The receiver of the message wants to make sure that the received data has not been manipulated on the way. This service can be realized by Hash Functions and Message Authentication Codes (MACs). Hash Function [8] and MAC [8] will be discussed later in Section 2.2.4 and Section 2.2.5.

2.1.3. Non-Repudiation

Non-Repudiation is a process of preventing a sender from denying a transmitted data. By this way, the sender of the message cannot claim that he/she did not send the message. This service can be realized by Digital Signatures [9] as explained in Section 2.3.3.

2.1.4. Confidentiality

Confidentiality is a process of protecting the data from disclosure to unauthorized adversaries during a communication. Confidentiality means that the message can only be read by authorized people (sender and the receivers), since it prevents the eavesdroppers to observe the data during a communication. This service can be realized by encryption/decryption operations.

2.2. Overview of Cryptography

The word *cryptography* means hidden or secret writing. Cryptography is the study of secure communication over in-secure channels. Suppose that, Bob wants to declare his love to Alice in basic communication model shown in Figure 1. Bob wants to send a love message to Alice and he does not want the content of message to be read by anyone else. If a communication network is in-secure, he has to encrypt the message content. On the other hand, Alice can decrypt the encrypted message.

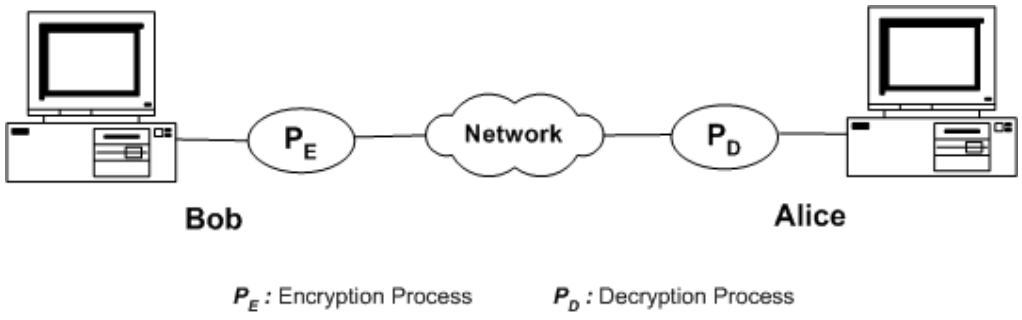

Figure 1. Basic Communication Model

Cryptologists have studied all kinds of problems related with security requirements that are discussed in Section 2.1. The common approaches can be categorized into two families of algorithms. One of them is symmetric cryptography and the other is the asymmetric cryptography. Asymmetric cryptography is sometimes called as public key cryptography. The systems based on public key cryptography are called as public key cryptosystems. Public key cryptosystems can be used for providing authentication, confidentiality, non-repudiation and integrity services.

2.2.1. Symmetric Cryptography

Symmetric cryptography is the traditional form of the cryptography in which the same key is used for both encryption and decryption. Therefore, the key should be kept as secret between the communicating parties. Symmetric cryptography may also provide authentication service by using MACs, which will be discussed in Section 2.2.5.

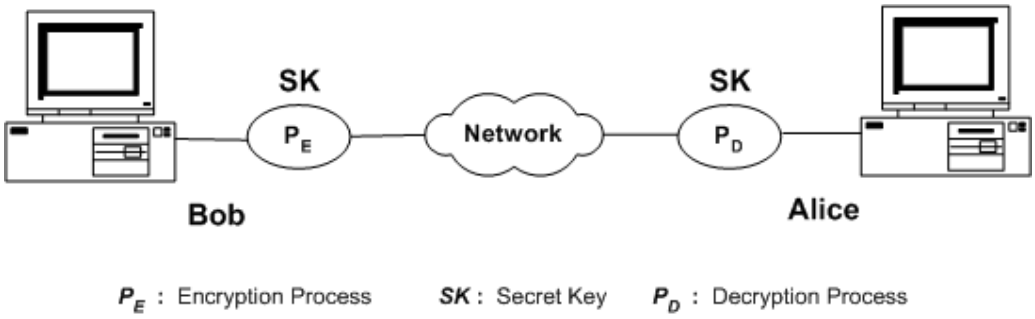

Figure 2. Communication Model with Symmetric Cryptography

Figure 2 shows the communication of Bob and Alice by using symmetric cryptography algorithms. Suppose Bob wants to send a message to Alice with symmetric cryptography. First, he decides a secret key together with Alice. Then, he uses that key while encrypting. On the other hand, Alice uses the same secret key while decrypting.

Well-known symmetric cryptography algorithms are the Rijndael that is accepted as the Advanced Encryption Standard (AES) [10], the Data Encryption Standard (DES) [18], International Data Encryption Algorithm (IDEA) [11], RC6 and RC5 [12].

2.2.2. Public Key Cryptography (PKC)

In symmetric cryptography, getting the sender and the receiver to agree on the same key is a challenging issue, especially if they are in different physical locations. In order to solve this problem, Whitfield Diffie and Martin Hellman invented a new concept called as public key cryptography in 1976 [1]. Public key cryptography is also called as asymmetric cryptography since -unlike to symmetric cryptography- different keys are used in encryption and decryption. In public key cryptosystems, every user has a key pair. One is public key that is freely available to everyone; the other is private key that is known only by the owner and should be kept secret. The most important and appealing property of a key pair is that obtaining a private key from a public key is practically impossible, although they are mathematically related to each other. Public keys are generally used for encryption of messages and verification of digital signatures. On the other hand, the private keys are used for decryption of encrypted messages and issuance of digital signatures.

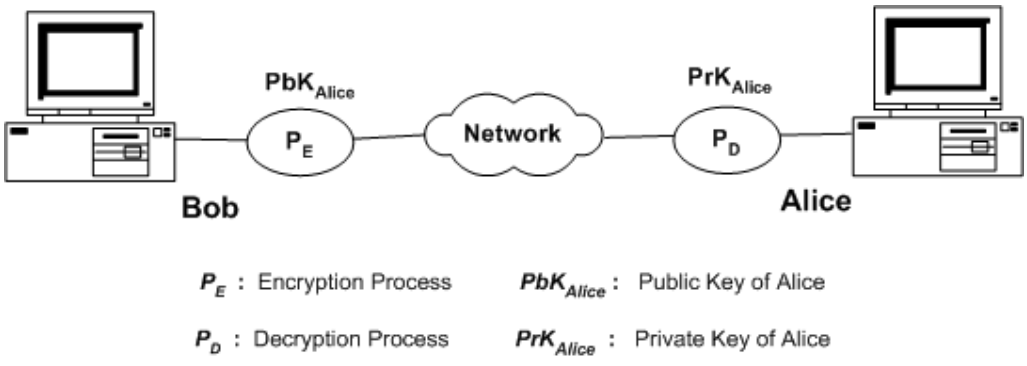

Figure 3. Communication Model with PKC for Encryption/Decryption

Figure 3 shows the communication of Bob and Alice by using public key cryptography algorithms. Suppose Bob wants to send a secret message to Alice using public key cryptography. In public key cryptography, encryption/decryption is very

simple. Bob gets Alice's public key from public key storage and uses it for encrypting the message. When Alice gets an encrypted message, she uses her own private key for decrypting the message. The encrypted message can only be decrypted by the private key of corresponding public key.

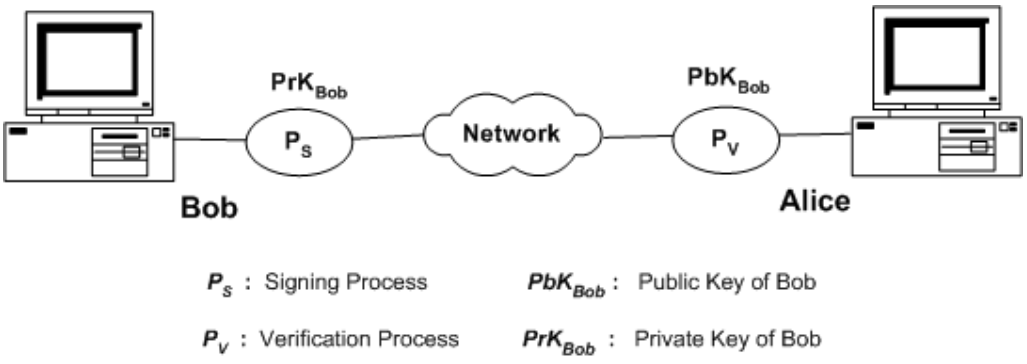

Figure 4. Communication Model with PKC for Signing/Verification

Figure 4 shows that Bob sends a digitally signed message to Alice by using public key cryptography algorithms. Bob uses his private key for digitally signing the message. When Alice gets the signed message from Bob, she gets Bob's public key from public key storage and uses it for verifying the signature of Bob on the message.

The most popular public key cryptography algorithm is RSA [13] which is invented by Rivest, Shamir and Adleman in 1977 at MIT [14]. There are other public key cryptography algorithms [48] in use such that Elliptic Curve Cryptosystem [15] and ElGamal Cryptosystem [16].

2.2.3. Comparison of Symmetric Cryptography and Public Key Cryptography

It is not necessary to transmit private keys in public key cryptography. This is a security-improving feature. In symmetric cryptography, secret keys must be transmitted over a communication channel or in an offline manner. Another major advantage of public key cryptography is that it provides non-forgeable digital signatures, which cannot be repudiated. In PKC systems, users cannot claim that their private key is compromised since the responsibility for protection of the private key is totally belongs to the owner, not to any central database. The main disadvantage of public key cryptography with respect to symmetric cryptography is speed. Symmetric cryptography algorithms operate much faster than public key cryptography algorithms. The best solution to provide both security and speed is employing Digital Enveloping mechanism that will be discussed in Section 2.3.5. That mechanism collates best parts of public key cryptography and symmetric cryptography. It should be clearly understood that the aim of public key cryptography is not to replace symmetric cryptography; they have to be used together.

2.2.4. Hash Function

Hash function is a function used for calculating the message digest that is sometimes called as message fingerprint. Hash function is a one-way function that produces fixed size output, given variable sized inputs. It is computationally infeasible to find out original message from the hash value. Besides, finding out two messages that produce the same hash result is also impractical. Hash functions are used in digital signatures and for integrity check. Some of the mostly used hash functions are Message Digest-5 (MD5) [17] and Secure Hash Algorithm-1 (SHA1) [8]. MD5 produces 128-bit digest, SHA-1 produces 160-bit digest.

2.2.5. Message Authentication Codes (MAC)

Message authentication code is an authentication value (a checksum) derived from the message. It is similar to digital signatures of the PKC systems. The only difference between MAC and digital signatures is that the same key is used for computing and verifying the MAC, whereas different keys are used in computing and verifying the digital signatures.

2.3. Cryptographic Solutions to Security Problems

In this section, we will discuss the cryptographic solutions and applications to security problems in a detailed way.

2.3.1. Confidentiality with Symmetric Encryption

Encrypting the data is a universal technique for providing confidentiality to data transmission. The original data is called as plaintext; the encrypted one is called as cipher text. Without knowing the secret key, the plaintext cannot be restored from the cipher text.

Suppose Bob wants to send an encrypted message for Alice. First, he should agree on a secret key with Alice in a secure way. He then, encrypts the message content with that key and sends over the insecure media (see Figure 2 in Section 2.2.1). If an adversary intercepts the encrypted message, he/she cannot restore it without knowing the symmetric key. Therefore, if the message has arrived to Alice, then it is obvious that this message is sent in a confidential way.

2.3.2. Message Integrity with Shared-Secret

Message integrity can be provided in several ways. Two of them will be discussed here. Presumably, it is supposed that Bob and Alice has the same shared-secret (actually it is a password string) and Bob wants to send a message to Alice by ensuring the integrity of the content.

i. Integrity with MAC

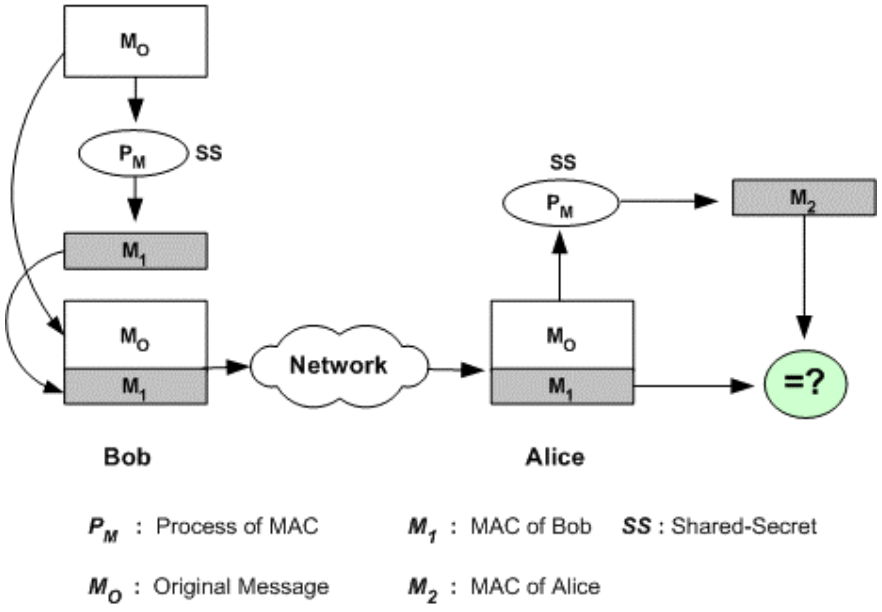

Figure 5. Message Integrity with MAC

Figure 5 shows the communication of Bob and Alice by using MAC. Bob gives both the message and shared-secret to the MAC function, then appends the output MAC value to the original message. He sends the message and the MAC value to Alice. Alice performs the same operations. First, she calculates the MAC value of the message that she has just received and then compares the received MAC value with calculated one. If

they are the same, then Alice makes sure that the message has not been altered on the way and originates from Bob.

ii. Integrity with Hash Function

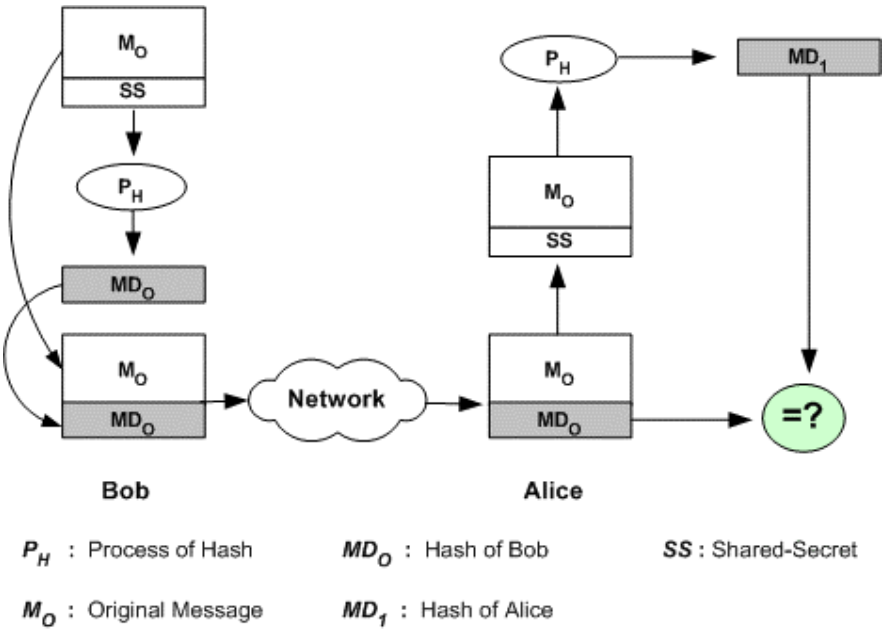

Figure 6. Message Integrity with Hash

Figure 6 shows the communication of Bob and Alice by using hash function. Bob appends the shared-secret value to the message and calculates the hash value of the message combined with shared-secret. He sends the original message and the hash value to Alice. She first takes the message and appends the shared-secret to it, then calculates the hash value with using the same hash function that Bob used. After that, she compares whether the received hash is equal to the one she calculated. If they are equal, then she makes sure that the message has been arrived to her unchanged.

2.3.3. Digital Signatures

Digital signature [19, 20] is an ultimate mechanism for providing non-repudiation. The main difference between digital signatures and hand-written signatures is that digital signatures are totally related with the content, while hand-written signatures are not. Because, a digital signature is a piece of data generated from the message content. They are generated by using private keys that are kept secret by the owner. It is more preferable to sign the hash (much shorter than the message) of the message instead of the message itself because of the performance reasons.

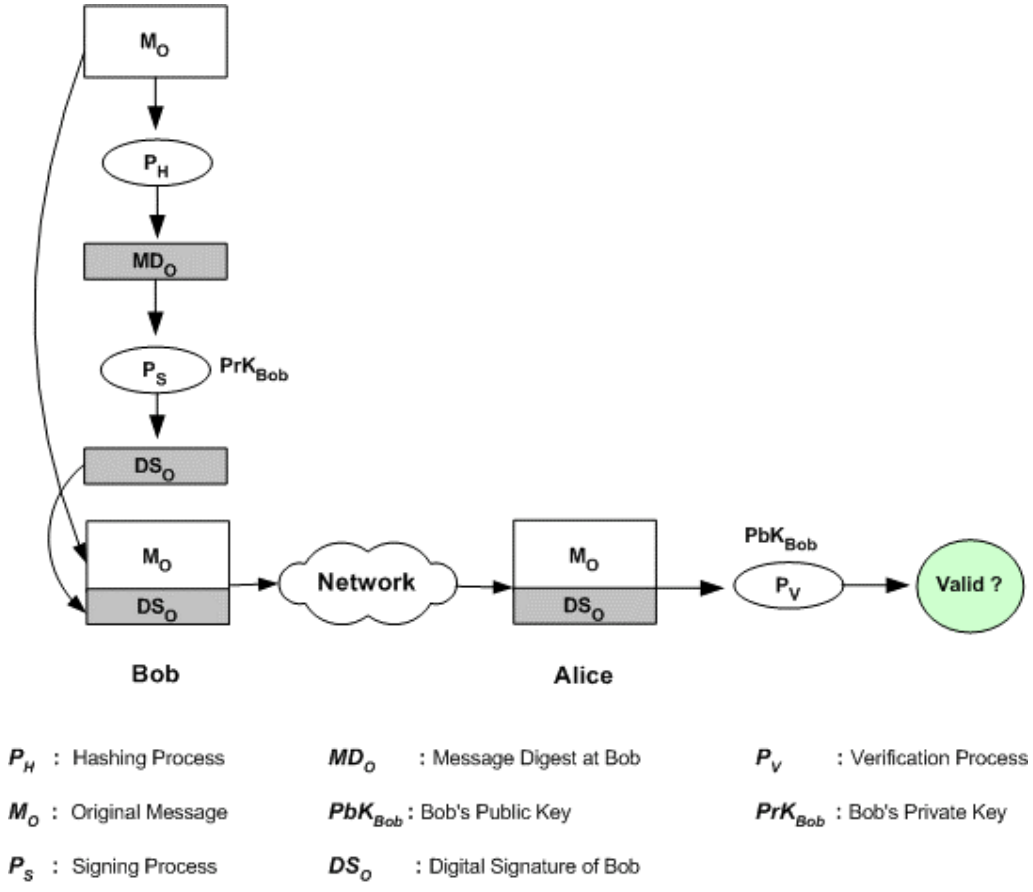

Figure 7. Digitally Signed Message

Figure 7 shows the communication of Bob and Alice by using digital signatures. Suppose Bob wants to send a digitally signed message to Alice. Initially he applies the hash function to the message and creates the hash. Then, he passes this hash value and his private key to the signing function to produce his signature on this message. He sends both the message and the signature to Alice. Alice obtains Bob's public key from public key storage and gives it to the verification function with the received message and the digital signature. If the function verifies the signature, then it means that the message has been definitely signed by the private key that corresponds to the public key used in verification. With this signature, Alice can make sure that message is from Bob and unchanged. In this way, authentication, integrity and non-repudiation services are satisfied.

2.3.4. Key Agreement

Key agreement is a protocol that provides two or more users to agree upon a key for using in symmetric cryptosystems. Key agreement protocols allow people to decide on the same key in a secure way over insecure channels without predetermined shared-secret. A characteristic example is Digital Envelopes, which will be discussed in the next section. Diffie-Hellman Key Exchange Protocol [18] is another well-known key agreement protocol.

2.3.5. Digital Envelopes

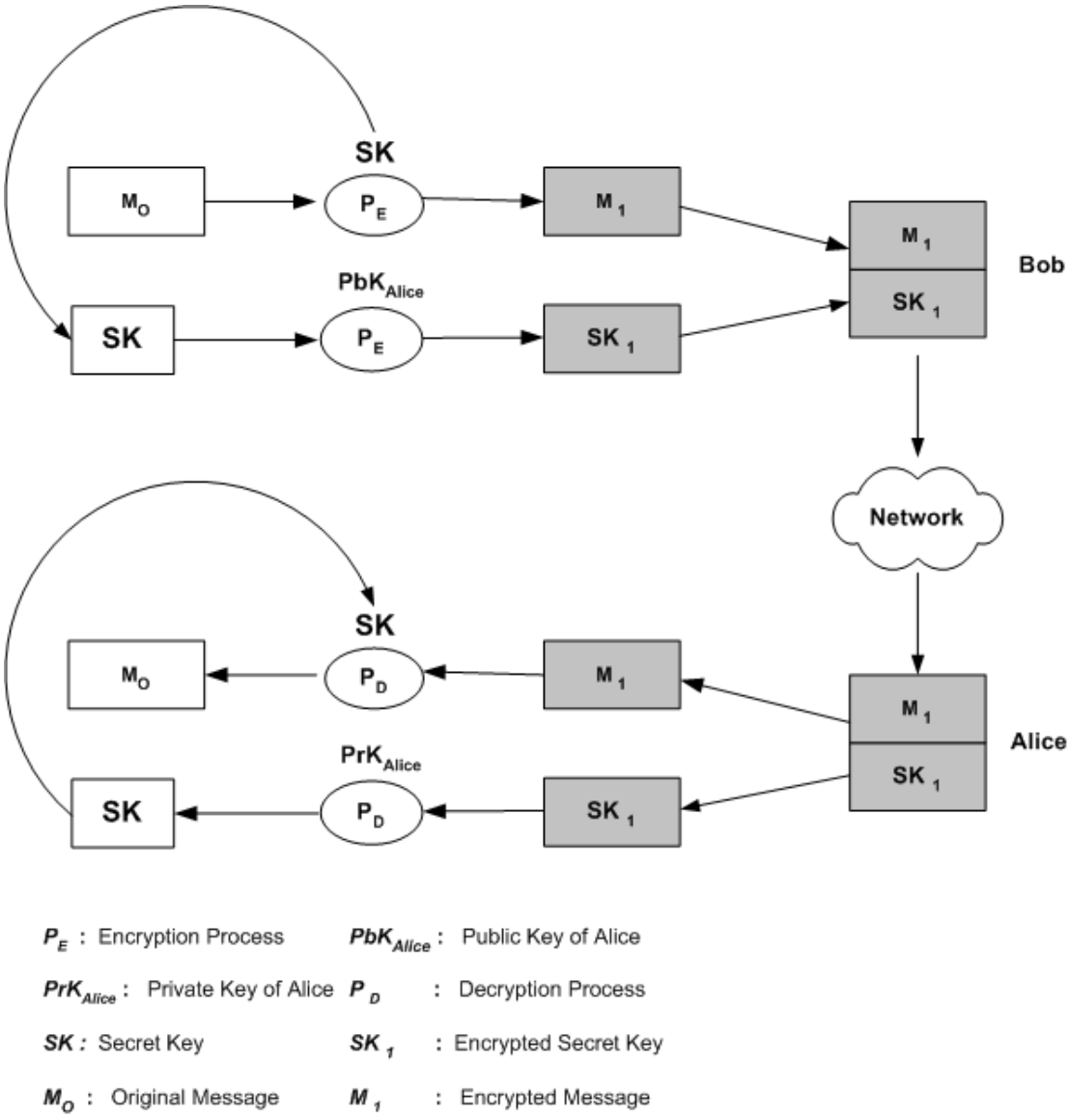

Figure 8. Digital Enveloping

Digital envelope is an attractive solution for fast message exchange that utilizes speed of symmetric cryptography and security of public key cryptography. A digital

envelope consists of two parts: One is the encrypted message using a symmetric key and the other is the symmetric key encrypted with the public key of the receiver using public key cryptography.

Suppose Bob wants to envelope his message for Alice as it is shown in Figure 8. First, he chooses a symmetric key at random and encrypts the message with that key. Then, he gets Alice's public key from public key storage and uses it to encrypt the symmetric key. He sends the encrypted message and the encrypted symmetric key to Alice. When Alice receives the digital envelope, she decrypts the encrypted symmetric key by using her private key and derives the symmetric key. Then, she decrypts the encrypted message by using the derived symmetric key. Finally she gets the original message, which is received as it is sent by Bob.

Digital envelope increases the performance of key exchange without sacrifying the security.

2.4. Key Distribution and Management in PKC Systems

Although PKC systems are more secure, they have some challenging problems. In PKC systems, everyone needs to make sure that the public key definitely belongs to the intended user. Otherwise, that public key should not be used, since it may belong to a wrong person. In this section, we will discuss the key distribution and management problems of PKC systems and solutions.

2.4.1. Key Management Problems

Problems related with the life-cycle of public keys are generally named as Key Management Problems [21] in PKC systems. Public key removal, update and recovery

are the most common key management operations. Every public key architecture (an architecture that uses PKC systems as their security base) should support fundamental key management operations.

2.4.2. Key Distribution Problems

In PKC systems, it is essential to bind the one's identity with his/her public key and consequently the private key. The problem of proof of possession of private key is called as Key Distribution Problem in PKC systems. In order to solve this problem, a trusted third party is inevitably necessary. Trusted third parties are the servers or associations that people agree on its trustworthiness. In this way, if trusted third party states that a public key belongs to a particular user, then people who trust trusted third party make sure about the legitimacy of that public key.

2.4.3. Digital Certificates

Digital certificates [22] are common solution for key distribution problem in PKC systems. It is a digitally signed document that provides a binding between user's identity and a public key. A digital certificate consists of an information part and a signature part. Signature is issued by a trusted third party. Information includes a serial number, issuer name, subject name, validity period and the public key, etc. Issuer name is the official name of the trusted third party. Subject name is the name of the user that will be certified. Validity period is the time period in which the certificate is valid. The issuers, i.e. trusted third parties, of certificates are called as Certification Authorities (CA). Every CA has to have a digital certificate that is issued by another CA or by itself (i.e. self-signed certificates). A CA that has self-signed certificate is the one at the top of the hierarchy. Such a CA is called as the Root-CA.

2.4.4. Public Key Infrastructure (PKI)

PKI [23] is an architecture that provides trustworthy certificate distribution and management. With the help of PKI services, confidentiality, integrity, authentication and non-repudiation services are provided. PKI performs key distribution by issuing certificate to its registered users and performs key management by supporting certificate revocation and certificate update. PKI must not be thought as a silver bullet. PKI is just an infrastructure. In order to utilize PKI, several applications that uses that infrastructure must be developed. SSL [24, 25] and S/MIME are successful applications that use digital certificates and PKI as an architecture.

2.4.5. Registration and Certification

Certification is a process of requesting and getting a certificate from a CA. Every user that requests a certificate from a CA has to be registered to a CA registry. Only the registered users may have certificates from a CA. Certification can be done both in online and offline manner. Both types of certification require identification and authentication of the requester. Standard certificate management protocols that contain "Initial Registration and Certification Request Protocol" [46], "Key Update Request Protocol", "Key Update Response Protocol", etc., are defined in RFC 2510 [26].

2.4.6. X.500 Directory and Lightweight Directory Access Protocol (LDAP)

X.500 [27, 28] Directory is described as "a distributed database, capable of storing information about people and objects in various nodes or servers distributed across a network". Clients can get certificates or other information from X.500 Directory by using the Directory Access Protocol (DAP). Because DAP is too cumbersome for many client applications, LDAP was developed by University of Michigan. It was developed further and standardized in the IETF as LDAPv3 [29]. All

the certificates that are issued by a CA should be stored in the corresponding X.500 Directory.

2.4.7. Certificate Revocation List (CRL)

Whenever a CA does not want to permit a specific certificate to be used anymore, it revokes that certificate by publishing its serial number in a blacklist called Certificate Revocation List (CRL) [22]. CRL includes certificates that have been revoked before their scheduled expiration date. Therefore, a CRL does not include expired certificates. CRLs are digitally signed by CAs and periodically updated. There are several reasons for a certificate to be revoked, such as compromise of private key, change in identity and promotion in a job or layoff.

2.4.8. Authority Revocation List (ARL)

Sometimes, use of a CA's certificate is not good anymore and it should be revoked. Certificate revocations of CAs are performed in ARL that is signed by upper level CA. Inclusion of the serial number of a specific CA's certificate in ARL means that it is not valid anymore. Besides, all of the user certificates of that CA are also functionally invalid even they are not revoked in CRL. Because a CA in ARL means that it is not a trusted third party anymore. It is sometimes catastrophic to revoke top level CAs.

2.4.9. Key Obtainment

Key obtainment is a process of getting a public key from the trusted third party. In PKI, key obtainment is an easy process that the requester just submits the distinguished name of the intended user to LDAP server and gets the certificate of that person. That

certificate is verified using the public key of the CA and, consequently, the public key of the user is validated.

2.4.10. Key Update

Key update is the process of renewing a public key. In PKC systems, periodic update of public keys is believed to increase the security of both the user and the system. Therefore, certificates are issued only for a limited time period. Suppose that a user wants to update his/her public key, first he/she should generate a new key pair. Next step is to sign it using old private key and sending this signed public key to the CA. CA checks the signature. Having verified it, CA issues a new certificate for the user and revokes the old one.

2.4.11. Path Construction and Certificate Chain

Path construction is a process of finding out a certification path, which reaches the verifier from a specific certificate to its trust anchor's certificate. That certification path is called as certificate chain. Root CA of a specific hierarchical PKI is called as the trust anchor of that PKI. Trust anchor's certificate is the point that trust begins. Path construction can be complicated when the PKI architecture is complex. There may be several certification paths between two specific certificates.

2.4.12. Path Validation

Applications should not use certificates, or public keys contained in them, without first constructing the certification path and validating it. Path validation is a process of identifying and verifying all of the certificates in the certificate chain. In order to verify all certificates in the chain, the verifier has to reach to its trust anchor's certificate at the

beginning. Valid certificates are the certificates that neither they are expired nor they are in the revocation lists (CRL or ARL). If any one of the certificates on the certificate chain is invalid for any reason, then whole chain becomes invalid too.

2.4.13. Multipurpose Internet Mail Extensions (MIME)

MIME [30] is the standard format for Internet e-mail. An Internet e-mail message comprises from a header and a body. Header is a structured information (defined in RFC 822) that is essential for the message transmission, while body is normally unstructured. MIME defines structure of body and permits on e-mail message to include graphics, sound, enhanced text, etc. MIME does not provide security services like encryption and digital signatures.

2.4.14. Secure/Multipurpose Internet Mail Extensions (S/MIME)

S/MIME [30] is a standard that adds digital signatures and encryption to MIME. S/MIME is supported as an add-on feature for security in e-mail client programs. S/MIME compliant applications use digital certificates for the public keys of the users.

2.4.15. Pretty Good Privacy (PGP) and Key Ring

PGP [31] is a widely used e-mail encryption/decryption application in the world since it is free, fast and secure. PGP [32, 33] utilizes the digital enveloping mechanism while sending an e-mail, therefore it is fast. It uses RSA (512-4096 bits) [13] as a public key cryptography algorithm, CAST (128-bits), IDEA (128-bits), 3-DES (168-bits) and Twofish (128-bits, 192-bits, or 256-bits) as a symmetric cryptography algorithm, and MD-5 (128-bits) and SHA-1 (160-bits) for message digesting. Every user in PGP has a key ring (“pubring.pgp” that is actually a signed file by user’s private key) that

comprises from well-known friends of the user and constructed by the user himself/herself. There is no precisely defined trusted third party in PGP, rather, every user may be a trusted third party and may certify another user. Therefore, a message from a person who is not in the receiver's key ring causes the receiver to hesitate. PGP [34] has a network of public key servers for storing all of the public keys. PGP key servers provide both key management and key distribution services, but not in authenticated manner.

2.5. Problems and Difficulties of PKI and PGP

PKI [34] and PGP are the most commonly used mechanisms to solve key distribution and management problems especially for e-mail applications. Although they provide useful services, they have some problems as will be described in this Section.

2.5.1. Unintended use of Certificates

Certificates are publicly available digital documents. A certificate may contain private information, such as e-mail address [51] that one may not want to disclose. Certificates are exchanged during SSL handshake. After that, certificates begin a tour on the Internet. If a malicious person catches a certificate, then he/she can easily use the e-mail address in it for sending junk mails. Privacy-sensitive people think that this is a problem.

2.5.2. Monetary Cost of Certificates

The certificates are not free. The corporations sell certificates for a limited period (generally for one year). There are some free certificates, which are given for trial

purposes. There are some shortcomings related with free certificates. Generally their validity periods are too short (a few weeks). The ones issued for longer periods are generally issued by unknown CAs, so those certificates are not verified worldwide. Free certification services generally offer Class-1 certificates where the identities of the certificate holders are not authenticated, but access to a specific e-mail address is validated. In this way, corporations aim to introduce and advertise their certification service and the corporation itself. In order to get certificates with identity control such as Class-2 or Class-3 certificates, customers have to pay money. For example, if a customer has a web server that contains peer-to-peer transfer of sensitive data then he/she has to buy an SSL web-server certificate about 70 – 80 $ per year.

2.5.3. Registration Authority in PKI

In order to obtain a Class-3 certificate one has to identify himself/herself to a CA in an offline manner, such that he/she submits his/her ID card, an official document that proves the affiliation. In this way, the user is registered to the Registration Authority (RA) of that CA. After registration, user obtains a shared-secret generally a MAC password, or sometimes physical card such as a smart-card or an e-token that contains shared-secret. These registration steps are believed as dissuasive and costly for most of the users of the system.

2.5.4. Self-Signed Certificates

Sometimes users find themselves in difficult conditions that they are forced to decide immediately about whether to trust a self-signed certificate or not. Two characteristic examples are the following e-mail exchange and the SSL handshake scenarios.

- If a user gets a signed e-mail from a user whose certificate belongs to another self-signed CA certificate then making a decision would be too difficult.

- A user may face with a self-signed SSL web server certificate and browser asks him/her to decide whether to trust or not. If a user selects not to trust this certificate, then he/she cannot see the content. Otherwise, if a user selects to trust a self-signed certificate, then there is risk of submitting sensitive information to malicious people such as mother's maiden surname or a secret password.

2.5.5. Certificate Chains

Using a certificate without constructing and validating its certification path, is an unusual operation. Constructing and validating a long certificate chain is a time-consuming operation. Having multiple certification paths for a single certificate, makes validation more time-consuming.

2.5.6. Checks from LDAP

In order to validate a specific certificate, at least three checks from LDAP have to be performed. One is the validity period check that is to control whether the current date and time is in between the validity interval specified in the certificate or not. The next is checking the CRL that aims to understand whether this certificate is revoked or not. The final one is the ARL check whose purpose is to decide whether the issuer certificate (CA certificate) is in ARL or not. If all checks succeed, then the certificate is accepted as a valid certificate. In order to accept an e-mail message as valid, performing so many checks for every certificate in the chain is too time-consuming.

2.5.7. Customization of Key Ring in PGP

In PGP, key ring is a local file (“pubring.pgp”) in the user’s file system. Key ring includes the list of trusted people. Therefore, trusting someone else is left to user only. As an example, if a PGP user receives a message from a sender who is not in the receiver’s key ring, then decision of trust belongs to user himself/herself. A receiver may trust a sender directly or indirectly in PGP. If a receiver trusts a sender directly, then receiver adds the sender’s public key to his/her key ring and accepts that sender as trusted anymore. In indirect trusting, a receiver may trust a sender who is trusted by another PGP user who is in the receiver’s key ring, and then receiver adds the public key of sender to his/her key ring. Many people are criticizing this mechanism since they believe the users may not have adequate information about public key cryptography. Those people accept PGP as a program for PKC experts.

2.5.8. Absence of Trusted Third Party in PGP

In PGP, there is a network of public key servers, which are responsible for keeping and distributing the users’ public keys. No trusted third party gives guarantee to the legitimacy of the public keys in the key servers. Besides, some of the users may forget to update their public keys in the corresponding key servers when they update their key pairs. Because of the lack of guarantee to the legitimacy of the public keys in the key servers, e-mail receivers are forced to decide on trusting the e-mail sender themselves alone. Therefore, there is no absolute trust mechanism in PGP.

2.6. Alternative Key Management and Distribution Solutions

Studies on finding out solutions to key management and distribution problems are not limited with PKI and PGP. There are several approaches that have been proposed. We will discuss two of them briefly in this section.

2.6.1. Public File Model of Diffie and Hellman

When Diffie and Hellman had introduced the PKC concept in 1977, they proposed a new model for key distribution and management. This is called as Public File model. In this model, there is a public file, which is to store the public keys only. Writing to this file is restricted, while reading is free.

2.6.2. Account Authority Model of Wheeler

Wheeler [36] proposed Account Authority Model for signature verification in 1990's. Account authority is a kind of trusted third party. It takes the responsibility of keeping all the public keys in the system. If anybody wishes to verify a specific signature, he/she sends it to the account authority and waits for response of account authority. Account authority checks the signature and returns the verification result to requester as valid or invalid signature, but does not disclose public keys.

2.7. Contribution of Thesis

As discussed in previous sections, PKI and PGP are well-known systems that promise solutions to key distribution and management problems of public key cryptography. PKC systems are mostly used in e-mail applications that has several

security issues [47, 49]. PGP is itself an e-mail application, while PKI is an infrastructure that produces certificates used in Secure Multi-purpose Internet Mail Extension (S/MIME), and other applications. Both PGP and PKI have several problems and difficulties (discussed in sub-sections of Section 2.5). Moreover, they are impractical. In this thesis, we propose a new e-mail system that will be as secure as them and more practical. In our system, we design and develop an application that aims to provide encrypted and/or signed message exchange between parties by using public key cryptography algorithms as a base. Name of our system is "Practical and Secure E-Mail System" that is called as "PractiSES". The objectives of PractiSES are below.

- System will solve both key distribution and management problems.
- Users of the system do not need to have depth information about PKC.
- System should not propose certificates to solve key distribution problem.
- Everything should be performed in an online manner even initialization.
- System should be easy to use and has to have user friendly GUIs.
- System should provide multi-language support (English and Turkish for the first version).
- Decryption, signature verification and key obtainment services should be performed transparent to the users.
- System should be free.
- System should sign the header of a message and present only the signed information.
- System should also check the correctness of sender's name, surname and e-mail address from the database of system's trusted third party while verifying the signatures.
- System should provide authentication of origin, integrity and confidentiality of a message and non-repudiation of sender.
- System should provide sending secure (signed and/or encrypted) e-mails to multiple recipients.

- System will be designed and implemented for close communities such as companies, universities or corporations in version 1.0.
- System should deserve “PractiSES” name by being practical and secure.

3. DESIGN OF PRACTICAL AND SECURE E-MAIL SYSTEM (PRACTISES)

PractiSES has been designed according to the well-defined objectives, which are explained in Section 2.7. First version of PractiSES will be a corporate version. It will serve to close groups. With its "Cross-Trust Module", PractiSES will serve to users of different groups in the next version. While designing PractiSES, every step of "Object Oriented Analyses and Design (OOAD) [37, 38]" has been performed systematically. First, the requirements of the system were clearly specified and documented in the requirement analyses phase. After that, system and application classes were determined in the design phase. Finally, the implementation and testing step is performed. Implementation details will be discussed in Section 4.

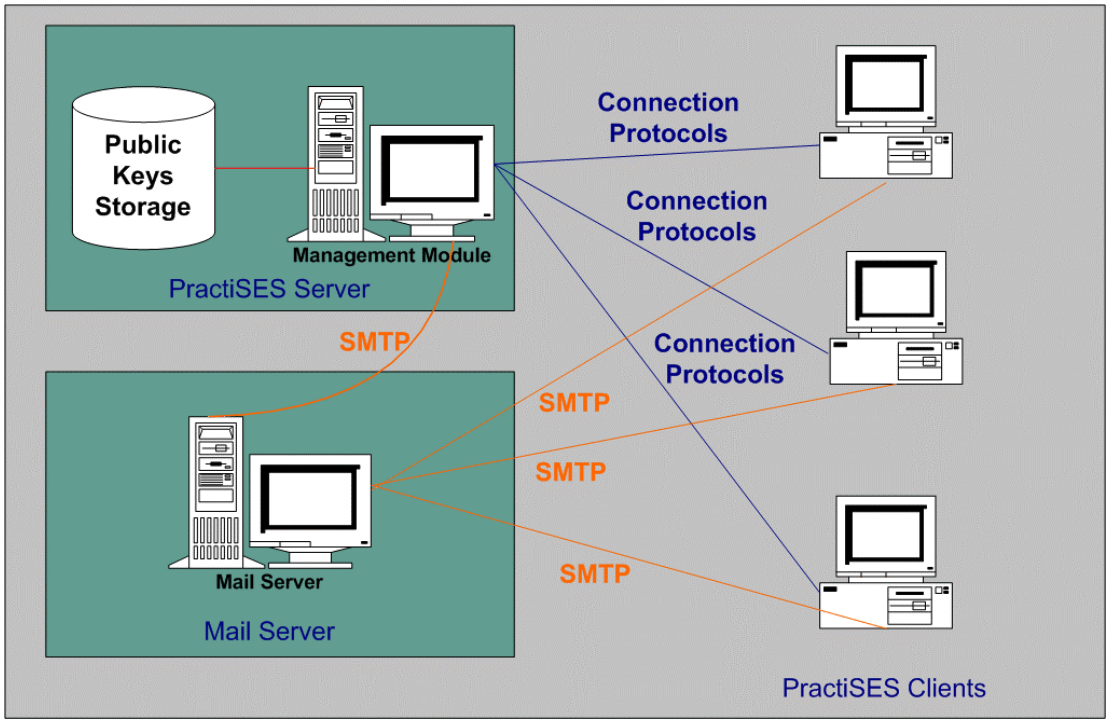

Figure 9. **Practi**cal and **S**ecure **E**mail **S**ystem (PractiSES)

3.1. PractiSES Architecture and Properties

As it is shown in Figure 9, PractiSES comprises from three parts, which are "PractiSES Server", "PractiSES Client" and "Connection Protocols". All parts are implemented in Java [39]. Server and clients need a "Mail Server" to exchange e-mails (using Simple Mail Transport Protocol (SMTP) [40]).

3.2. PractiSES Server

It is the server side of the system. It consists of a public key storage and a management module. Public key storage is an information database of users. Management module co-works with and operates on the public key storage. PractiSES Server is managed by an administrator. PractiSES Server has a key-pair, which comprises from public and private key. Public key is available to everyone from a public web or ftp site. Private key is stored in the server's file system as encrypted. Only the administrator may know the password of that file. PractiSES Server is set up on a special server machine that operates fast and has high storage capacity. It should be located in a secure room, into which no unauthorized person can enter. Both the management module and public key storage run on the same machine.

3.2.1. Public Key Storage

Public key storage is a database that keeps all the public keys and authentication information of registered users in the system. Only an administrator via the management module may access to the information in the public key storage.

Public key storage contains values of ID, name, surname, e-mail address and shared-secret information (e.g. mother's maiden surname) of the potential users. It is

assumed that such information exists in organization's records and they are conveyed to the public key storage. This potential user information is used for partial user authentication during the initialization protocols.

During the connection protocols, storage may be updated to store the MAC passwords and public keys of users.

3.2.2. Management Module (MM)

Management module is a software that operates on the public key storage. Only an administrator can start the server by submitting a password as shown in Figure 26.

MM has a GUI, by which an administrator may have a chance to perform service operations, registration operations and security operations. With service operations, administrator can start/stop PractiSES Server and monitor the important events from logger screen. With registration operations, administrator can register a new user to the system, update his/her information, or remove him/her. With security operations, administrator can generate/update key pair of the server, inform users about key update, and generate MAC passwords of users.

Beside the GUI operations, MM is an entity of the connection protocols that are responsible for authenticating a user, storing the public key of a user, and responding to public key obtainment/removal/update requests etc. MM listens to the clients' connection requests on a specific port. It processes the clients' requests in a concurrent manner. It has a connection manager (master server) to assign different threads (slave servers) according to clients' connections.

3.2.3. Security Aspects of PractiSES Server

There are three security aspects of PractiSES Server:

(1) Physical security of the server machine and the public key storage: No one can stop servicing the PractiSES Server by turning the machine off, since both the management module and the public key storage run on the same machine and machine is in a physically secure room.

(2) Security of information in the public key storage: Access to the public key storage is restricted only to the administrator and the management module. Because MM guarantees the operations' security by secure connection protocols, security and ACID (Atomicity, Consistency, Isolation and Durability) properties of public key storage are satisfied after every transaction.

(3) Security of server's private key: The private key of server is kept as encrypted in the server's file system. In case server's private key is lost, MM has an option to update the key pair of the server and informing users about that.

3.3. PractiSES Client

It is the client side of the system. It comprises from only an e-mail client module with additional security options. Client module may run on any PC. As all other e-mail client systems, PractiSES Client needs an SMTP server (not necessarily the same one that MM is connected).

3.3.1. Client Module (CM)

CM is the software behind the PractiSES Client. It has user friendly GUI. Moreover, it has security tools that provide secure data exchange between clients and the server. Security tools are actually the connection protocols that can be triggered from the menu bar of the GUI. Initializator, key updater, key remover, and key pair generator are the security tools of CM. As an example, initializator starts initialization protocol, provides key pair generation, and uploads the public key to the server's public key storage.

After a user, who has an e-mail account on the mail server, authenticates himself/herself to the mail server can use the CM to send/receive emails. Several users can use a single CM on the same machine, since CM supports changing the current user by a new one with the account's password of new user. Every account owner may create his/her own "account.ini" file on the local file system that keeps account settings. The users who have not created the "account.ini" file cannot use the CM.

In order to run a CM, an account holder should select his/her account as shown in Figure 27 and submit a correct password for it as shown in Figure 28. He/she has at most three chances to submit a correct password, otherwise program exits. If he/she submits the correct password, then CM asks him/her for whether to start the "initialization and key settlement protocol" immediately or not. If he/she selects starting the protocol immediately, then his/her key pair is generated and the public key is uploaded to the public key storage of PractiSES Server. Consequently, the corresponding private key is stored on clients' local disk in encrypted manner. Before signing a message or decrypting an encrypted message for a user, the password dialog will ask the private key password of that user in CM as shown in Figure 29. User has a chance to start the "initialization and key settlement protocol" later, since PractiSES Client GUI has a menu to trigger that protocol.

Although a CM can be used as a classical e-mail client, its superiority lies in its security tools. PractiSES users, can send/receive signed and/or encrypted e-mails as shown in Figures 30, 31, 32, 33, 34, 35, 36 and 37 of Appendix C. Another plus of PractiSES Client is that signature verification process of CM also compares the name, surname and e-mail address of sender in the message with the values in public key storage, and informs the result of any mismatch to receivers.

PractiSES Client supports sending secure (signed and/or encrypted) e-mails to multiple recipients. There is no limitation to number of recipients in sending secure e-mails.

3.4. Mail Server and SMTP

Mail Server is the one of the important parts of PractiSES. Both the server and the clients need it. Server needs a mail server to distribute the MAC passwords to the clients. Mail server may also be used by server to inform users about server's key update. Clients need it each time they send either a secure or a non-secure e-mail. Mail server relays all messages using Simple Mail Transport Protocol (SMTP).

3.5. Connection Protocols

Connection protocols are implemented in both Management Module (MM) and Client Module (CM) by using Java. The most appealing characteristic of the connection protocols is their security features. The protocols are leveraged on TCP/IP protocol stack. MM and CM are connected to each other over TCP sockets.

The secure connection protocols are:

(i) Initialization and Public Key Settlement Protocol (InitKeySet)

(ii) Public Key Obtainment Protocol (KeyObt)

(iii) Public Key Removal Protocol (KeyRem)

(iv) Public Key Updating Protocol (KeyUpdate)

(v) Un-signed Public Key Removal Protocol (USKeyRem)

(vi) Un-signed Public Key Updating Protocol (USKeyUpdate)

3.5.1. Initialization and Public Key Settlement Protocol (InitKeySet)

Initial interaction between CM and MM is the initialization phase. First, an end user introduces and authenticates himself/herself to the MM. Then, user's public key is uploaded to the public key storage of PractiSES Server. In order to authenticate the clients, MM uses clients' private information that already exist in the public key storage, such as ID, shared-secret, birthday and some other identity information. The sequence diagram of *InitKeySet* protocol is presented in Figure 10.

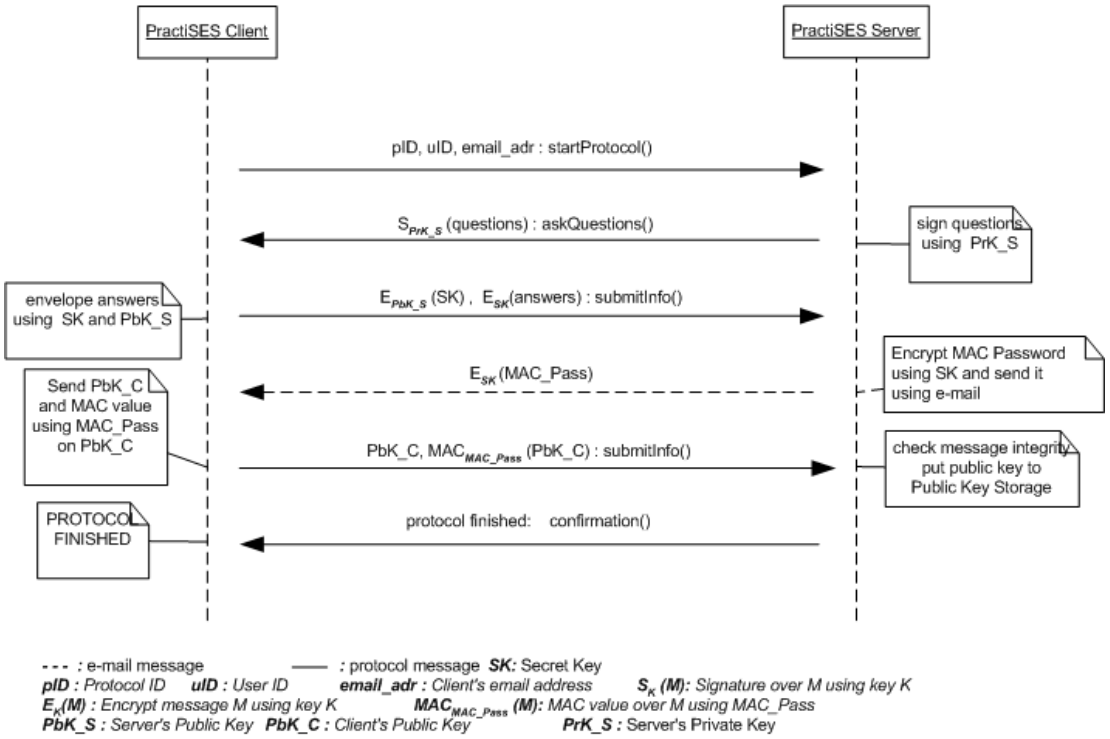

Figure 10. Sequence Diagram of InitKeySet Protocol

Details of the protocol:

1. PractiSES Client connects to the PractiSES Server. An end user introduces himself/herself to the server by providing his/her ID and email address with a protocol-ID as "0". (Protocol-ID = 0 means that protocol is InitKeySet). First, server detects the protocol-ID as "0" and executes the *InitKeySet* protocol in the server side.
2. The server checks and validates user ID and email address from the public key storage. If the submitted information is correct, then server asks user for private information. Server signs the questions before they are sent.
3. Client sends the answer encrypted using a secret key and the secret key encrypted using server's public key.
4. Server gets the secret key that is encrypted with server's public key and decrypts the user's answer. If answer is correct, then server sends an e-mail that contains

server-given MAC password encrypted by using the secret key. Otherwise protocol stops and socket is closed.

5. Client decrypts the MAC password from that e-mail. It uses this password to provide integrity for the message that contains his/her public key.
6. Server checks the MAC within the message that contains the user's public key. If verified, then public key is stored in the public key storage. Server sends a confirmation message about successful/unsuccessful key upload. Protocol finishes.

This protocol can be figured out in a state diagram as in Figure 11.

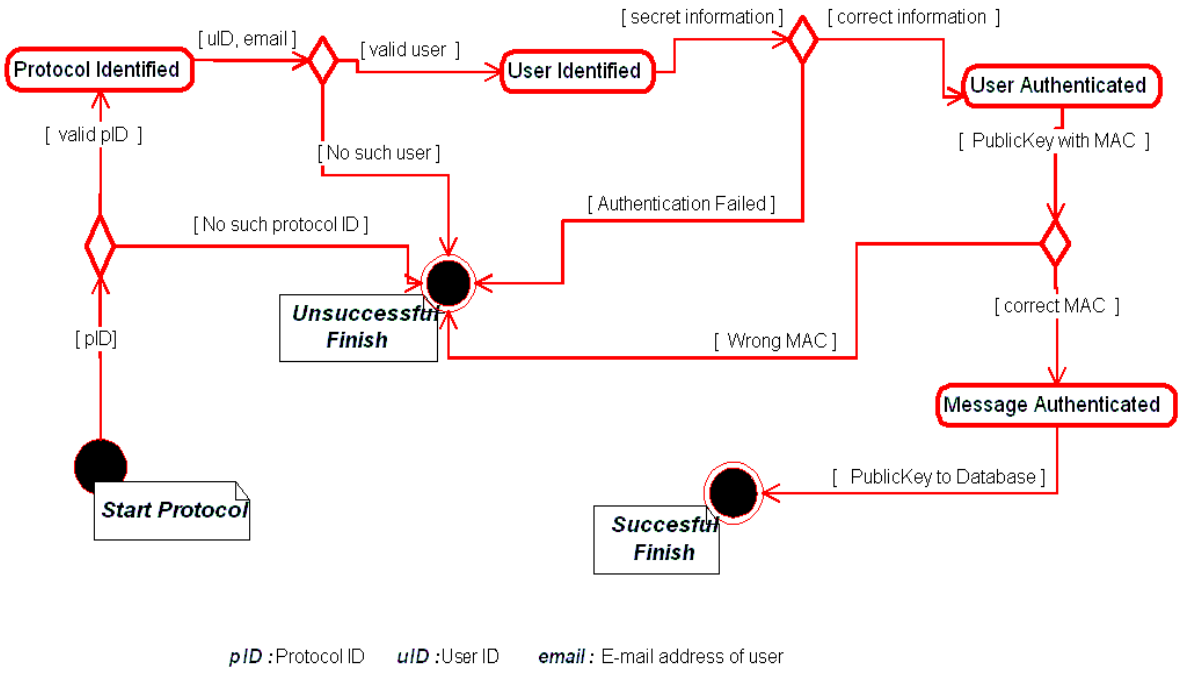

Figure 11. State Diagram of InitKeySet Protocol

3.5.2. Public Key Obtainment Protocol (KeyObt)

The easiest and mostly used protocol of PractiSES is *KeyObt* protocol. A user may need another user's public key to send an encrypted e-mail and verify that user's signatures. To get another user's public key from the public key storage, *KeyObt* protocol is designed. The protocol does not need to identify the requester's credentials since everybody can learn each other's public key. The sequence diagram of *KeyObt* protocol is presented in Figure 12.

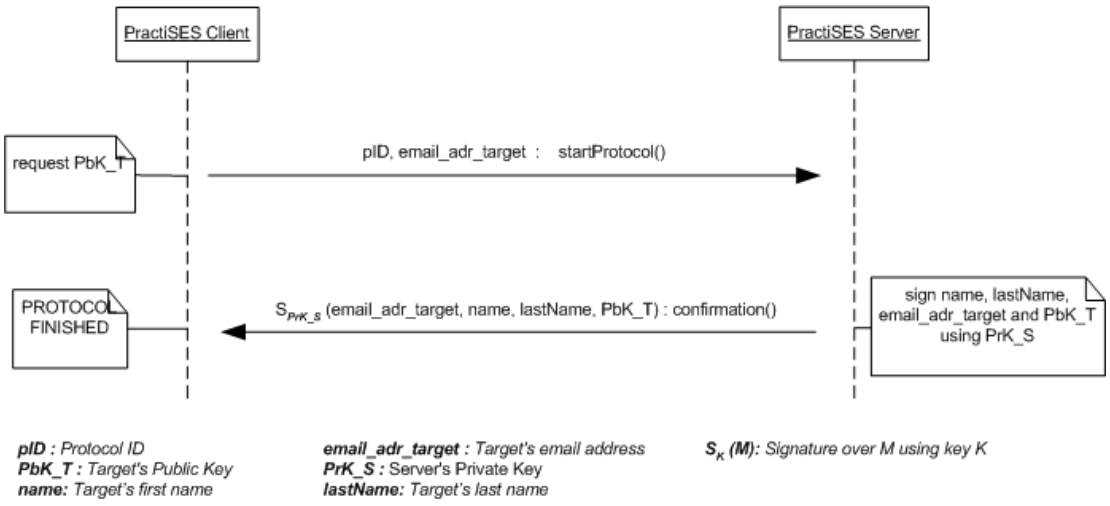

Figure 12. Sequence Diagram of KeyObt Protocol

Details of the protocol:

1. PractiSES Client connects to PractiSES Server. An end user sends the protocol ID as "2" and the e-mail address of the intended person. First, server identifies the protocol ID and gets the intended person's public key from public key storage. If no such user or public key belonging to that user in the public key storage, then protocol returns an error code.
2. Server delivers the name, last name, e-mail address and public key of target user to the requester. Before sending this information, they are signed by server's

private key. In this way, requester can make sure about the legitimacy of the public key and the other information it received. Protocol finishes.

This protocol can be figured out in a state diagram as in Figure 13.

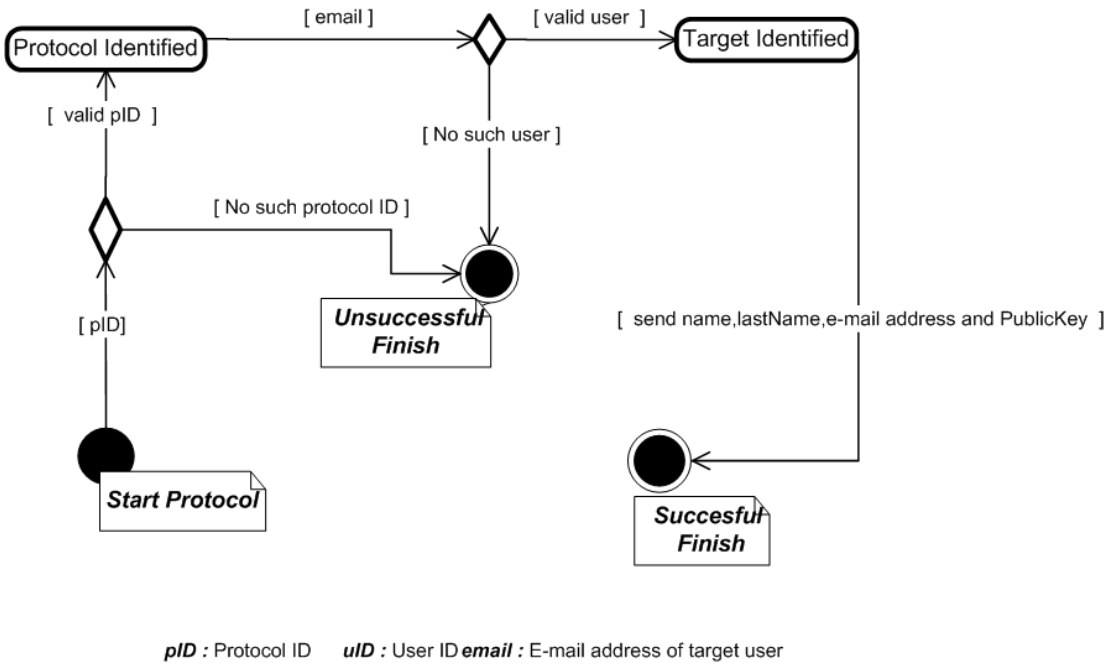

Figure 13. State Diagram of KeyObt Protocol

3.5.3. Public Key Removal Protocol (KeyRem)

In the case of compromise of a user's private key, that user must remove his/her public key from the public key storage. This can be accomplished using the *KeyRem* protocol. The most important property of *KeyRem* is that the requester should sign the request by current private key while removing corresponding public key. The sequence diagram of *KeyRem* protocol is presented in Figure 14.

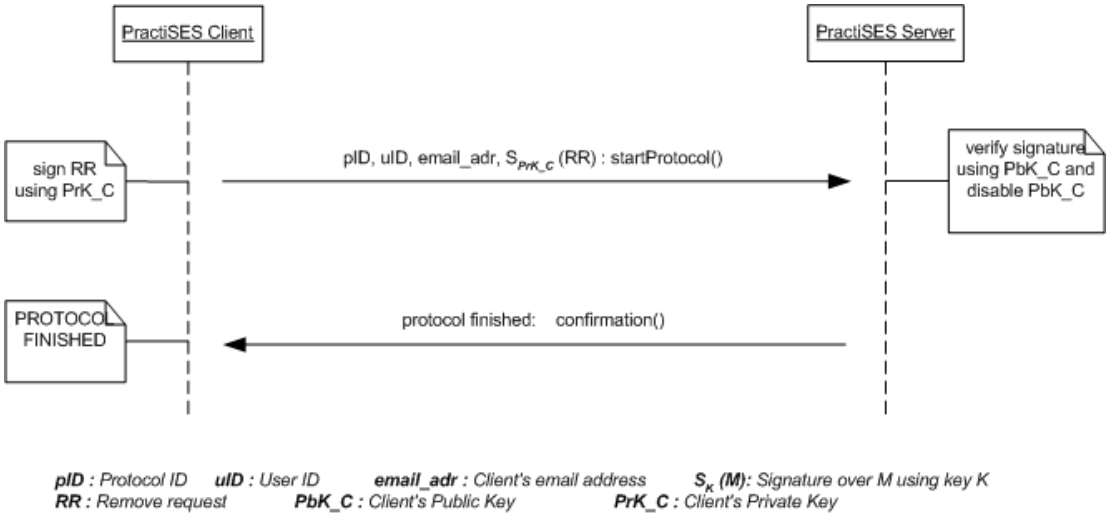

Figure 14. Sequence Diagram of KeyRem Protocol

Details of the protocol:

1. PractiSES Client connects to the PractiSES Server. First, an end user introduces himself/herself to the server by providing his/her user ID and e-mail address with protocol ID "4". Then, he/she signs the message "I want to remove my public key from storage" using his/her private key.
2. The server identifies the protocol ID and verifies the user's signature. If signature is valid then protocol continues, otherwise server closes the connection and the protocol finishes unsuccessfully. The server does not literally remove the public key, but marks it as deleted and puts a date and a time of deletion. That public key should not be used after marked deletion time. Protocol finishes.

This protocol can be figured out in a state diagram as in Figure 15.

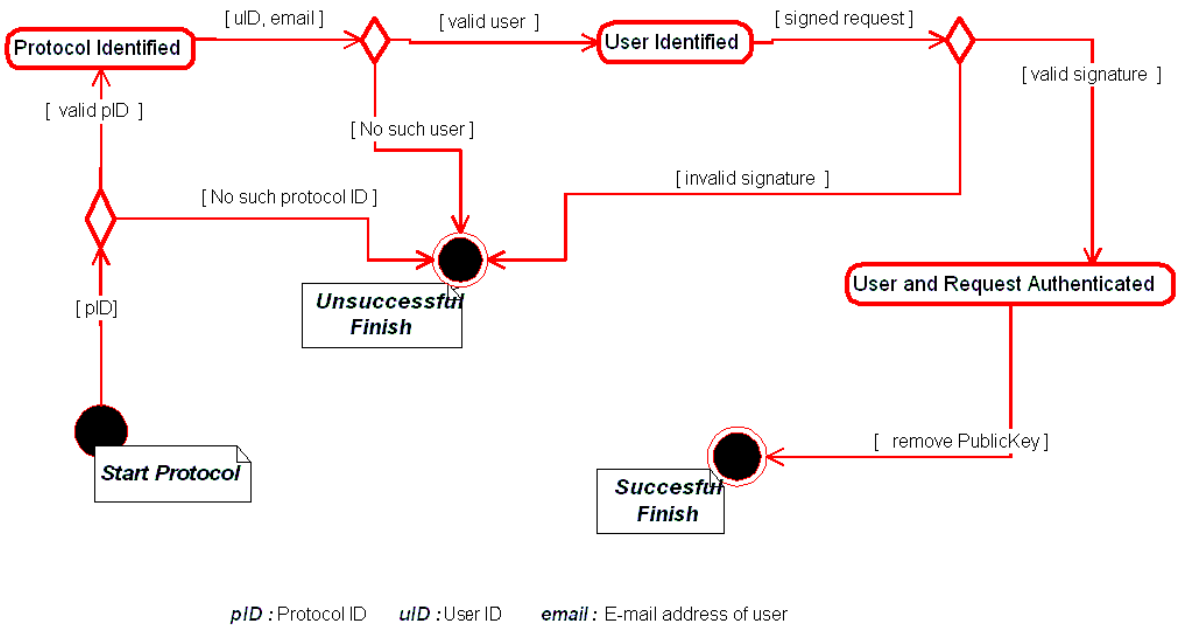

Figure 15. State Diagram of KeyRem Protocol

3.5.4. Public Key Update Protocol (KeyUpdate)

A user may want to update current public key in the public key storage. One reason may be a key compromise. It is also recommended to update the keys periodically for the sake of improved security. Key update can be accomplished using the *KeyUpdate* protocol.

The sequence diagram of *KeyUpdate* protocol is presented in Figure 16.

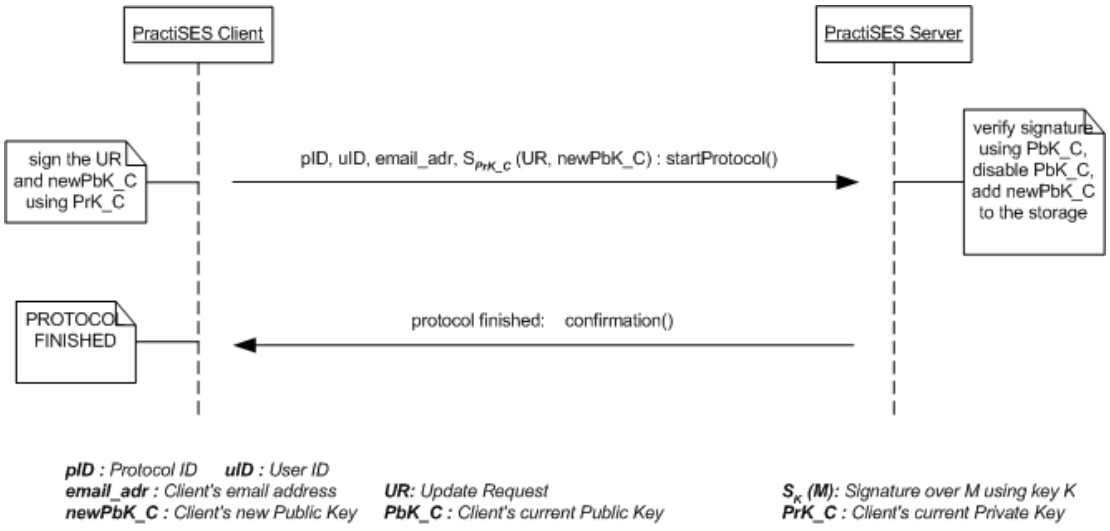

Figure 16. Sequence Diagram of KeyUpdate Protocol

Details of the protocol:

1. PractiSES Client connects to the PractiSES Server. An end user introduces himself/herself to the server by providing the his/her ID and an email address, with protocol ID "6". Then, he/she signs the message "I want to update my public key from storage" and the new public key using his/her private key. First, server identifies the protocol ID and verifies the user's signature. If signature is valid and user identified, then protocol continues otherwise server closes the connection and the protocol finishes unsuccessfully.
2. Server first disables the current public key by putting a "deleted" flag. Finally, it adds the new public key into the public key storage. Protocol finishes.

This protocol can be figured out in a state diagram as in Figure 17.

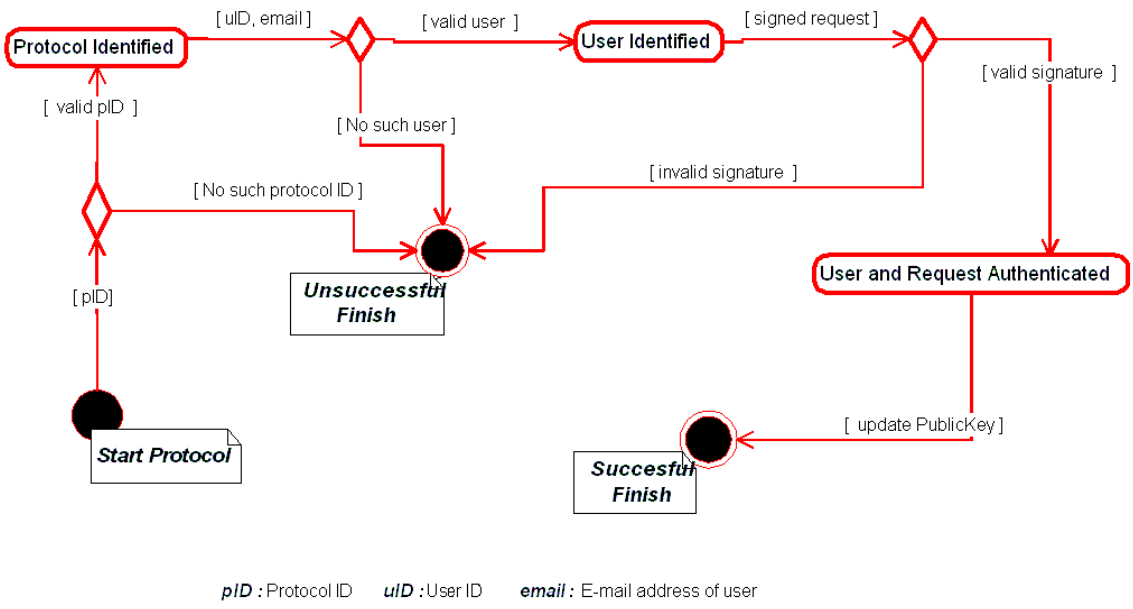

Figure 17. State Diagram of KeyUpdate Protocol

3.5.5. Un-Signed Public Key Removal Protocol (USKeyRem)

A user may want to remove current public key from the public key storage, even if the corresponding private key is lost. In this way, the user suspends using PractiSES till setting new public key to the storage. Since the private key is lost, the user cannot run the *KeyRem* protocol. *USKeyRem* protocol is designed to remove keys in an authentic manner, even if the corresponding private key is not available. The difference of this protocol is that *USKeyRem* uses MAC instead of digital signatures for integrity and authentication purposes. The sequence diagram of *USKeyRem* protocol is presented in Figure 18.

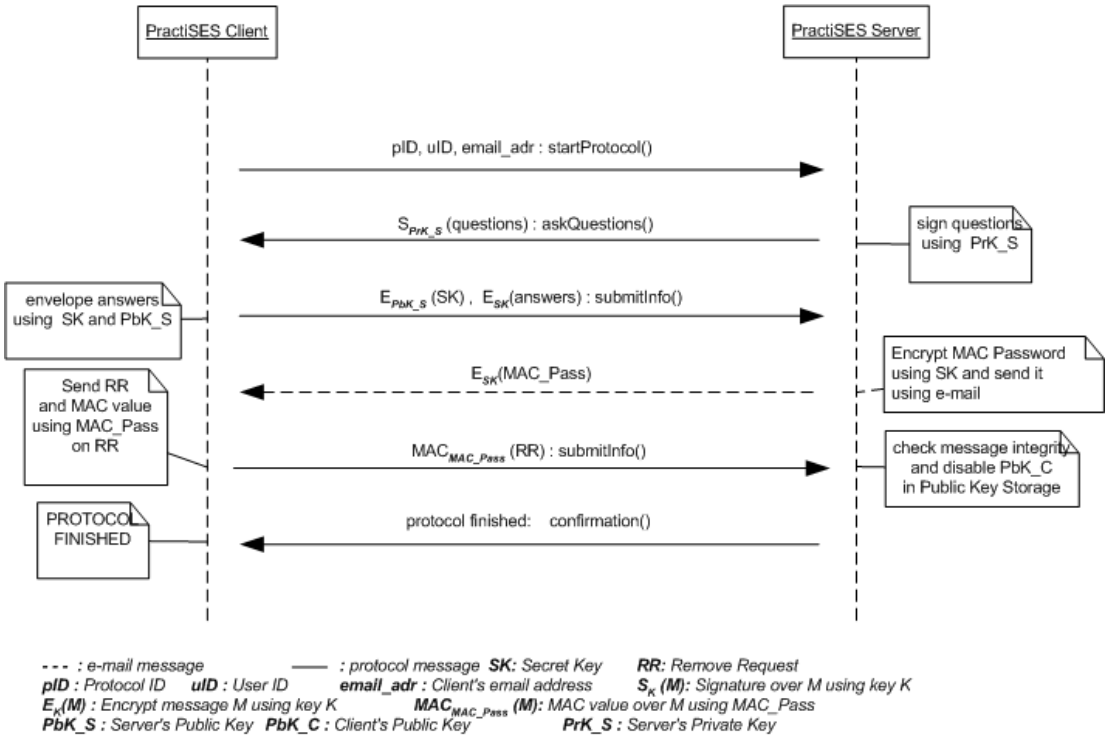

Figure 18. Sequence Diagram of USKeyRem Protocol

Details of the protocol:

1. PractiSES Client connects to the PractiSES Server. An end user introduces himself/herself to the server by providing the user ID and an e-mail address, with protocol ID "8". First, the server identifies the protocol-ID as "8" and executes the *USKeyRem* protocol in the server side.
2. Server checks and validates user ID and e-mail address from the public key storage. If submitted information is correct, then the server asks the user for private information. Server signs the questions before they are sent.
3. User sends the answer encrypted using a secret key and the secret key encrypted using server's public key.
4. Server gets the secret key that is encrypted with server's public key and decrypts the user's answer. If answer is correct, then server sends an e-mail that contains server-given MAC password encrypted by using the secret key. Otherwise protocol stops and socket is closed.

5. User decrypts the MAC password from that e-mail. He/she uses this password to provide integrity for the message "I want to remove my current public key".
6. Server checks the MAC within the message that contains key remove request. If verified, then it disables the current public key from the public key storage by setting a "deleted" flag. Server sends a confirmation message about successful/unsuccessful key removal. Protocol finishes.

This protocol can be figured out in a state diagram as in Figure 19.

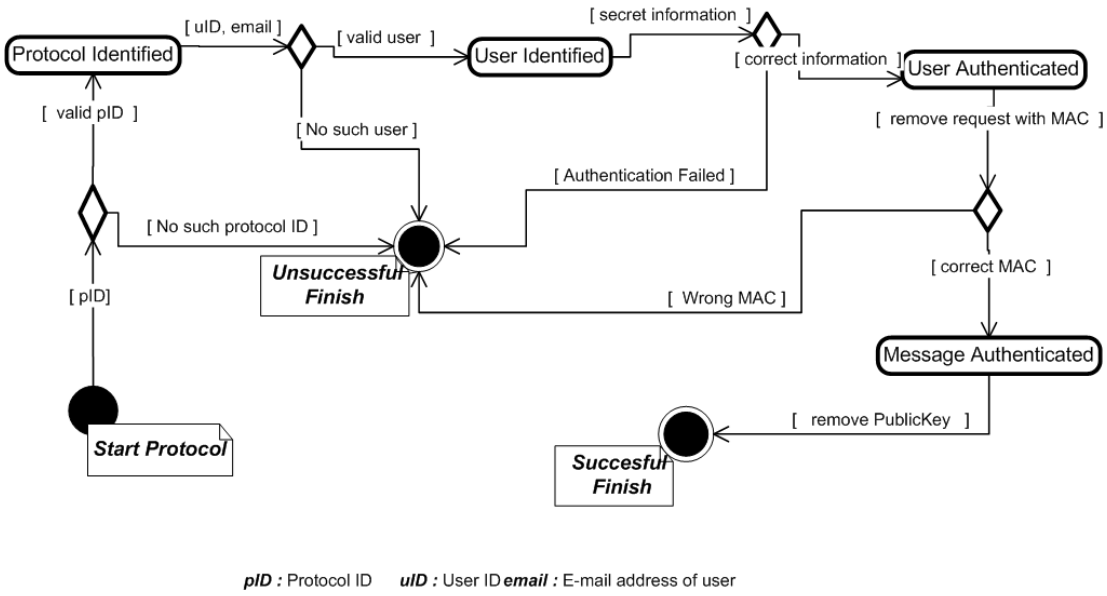

Figure 19. State Diagram of USKeyRem Protocol

3.5.6. Un-Signed Public Key Update Protocol (USKeyUpdate)

A user may want to update current public key in the public key storage, even if the corresponding private key is lost. Since the private key is lost, the user cannot run the *KeyUpdate* protocol. *USKeyUpdate* protocol is designed to update keys in an authentic manner, even if the corresponding private key is not available. The difference

is that *USKeyUpdate* uses MAC instead of digital signatures for integrity and authentication purposes. The sequence diagram of *USKeyUpdate* protocol is presented in Figure 20.

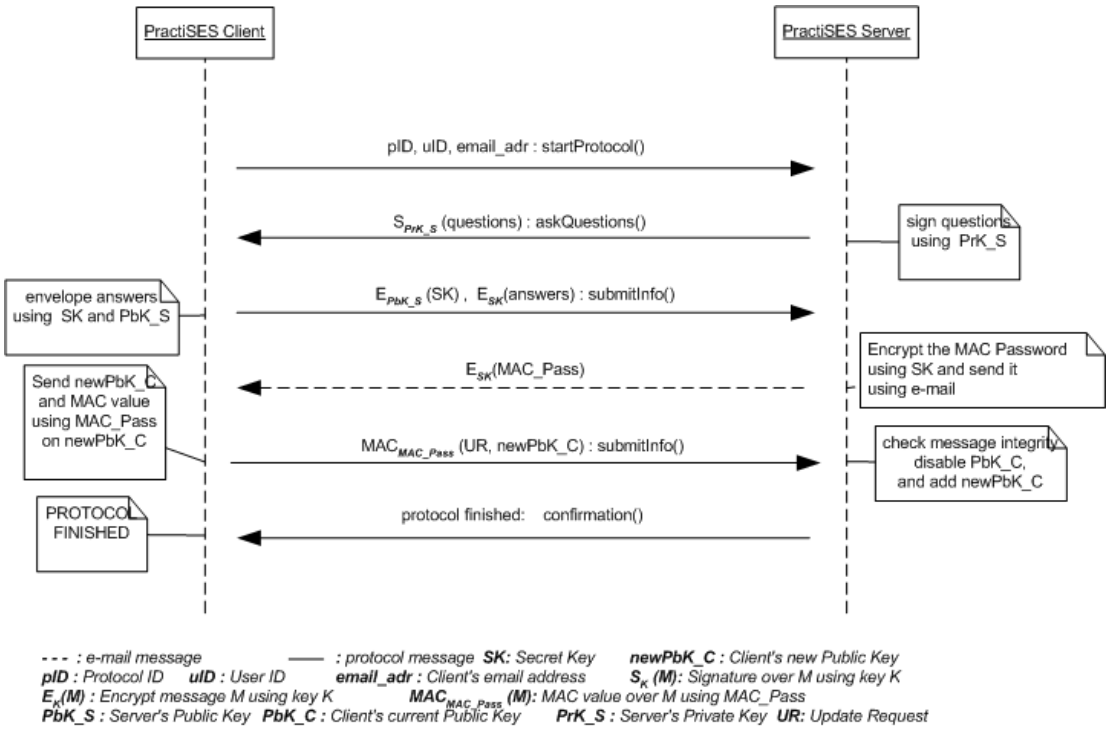

Figure 20. Sequence Diagram of USKeyUpdate Protocol

Details of the protocol:

1. PractiSES Client connects to the PractiSES Server. An end user introduces himself/herself to a server by providing the user ID and an e-mail address, with protocol ID "10". First, the server detects the protocol-ID as "10" and executes the *USKeyUpdate* protocol in the server side.
2. The server checks and validates user ID and e-mail address from the public key storage. If the submitted information is correct, then the server asks user for private information from the storage. Server signs the questions before they are sent.

3. User sends the answer encrypted using a secret key and the secret key encrypted using server's public key.
4. Server gets the secret key that is encrypted with server's public key and decrypts the user's answer. If answer is correct, then server sends an e-mail that contains server-given MAC password encrypted by using the secret key. Otherwise protocol stops and socket is closed.
5. User decrypts the MAC password from that e-mail, and then he/she uses this password to provide integrity for the message that contains his/her update request and public key to the server.
6. Server checks the MAC within the message that contains update request and new public key. If verified, then it disables the current public key from the public key storage by setting a "deleted" flag. And then it puts new public key to the public key storage. Finally, server sends a confirmation message about successful/unsuccessful key update. Protocol finishes.

This protocol can be figured out in a state diagram as in Figure 21.

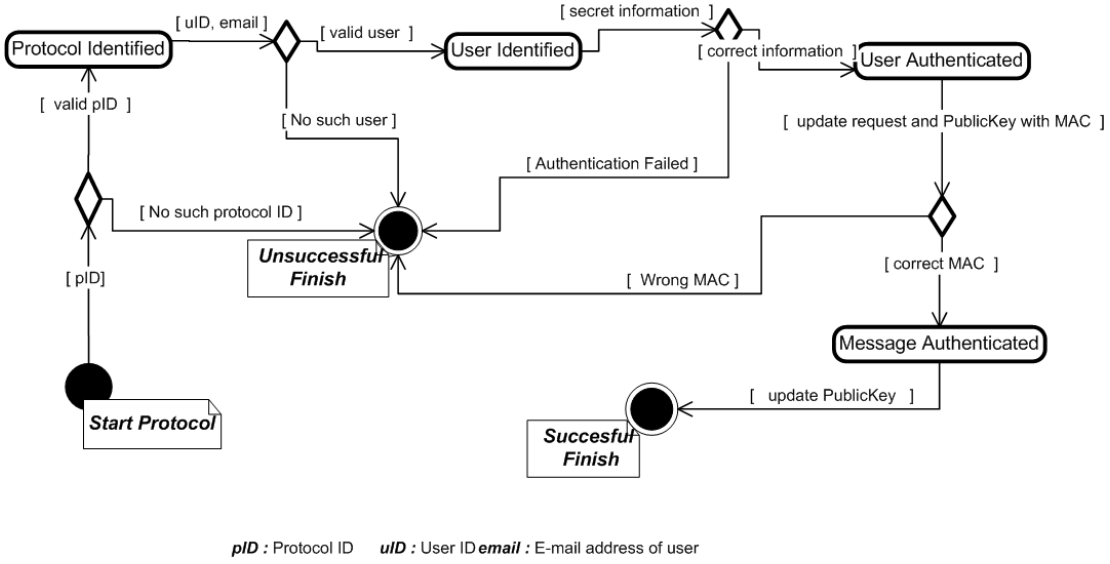

Figure 21. State Diagram of USKeyUpdate Protocol

3.6. Advantages of PractiSES over PKI and PGP

PractiSES is not only an architecture, which provides key distribution and management, but also an e-mail client that provides both insecure (normal e-mail) and secure (encrypted and/or signed) e-mail exchange. Moreover, it has several advantages over S/MIME (uses digital certificates of PKIs) and PGP. They are presented below:

- Users in PractiSES must not need to have depth information about public key cryptography which is the case in PGP.
- PractiSES delivers the public keys and other information of user only to group members. Therefore, the problem discussed in Section 2.5.1 does not exist in PractiSES.
- In PractiSES, there is no monetary cost of delivering public keys that is discussed in Section 2.5.2.
- Every key operation, even initialization, can be performed in an online manner in PractiSES.
- Since the information about potential PractiSES users have already been entered to the database, they do not have to perform an offline registration. This is a solution to problem discussed in Section 2.5.3.
- Since there is a single trusted third party (server) for users of a specific group, the problems discussed in Section 2.5.4, 2.5.5, and 2.5.6 do not exist in PractiSES.
- PractiSES is easy to use and has a user friendly GUI.
- PractiSES provides multi-language support (English and Turkish for v1.0).
- Decryption, signature verification and key obtainment services are performed transparent to the users.
- PractiSES is freeware software, and available from the project's web site [50].
- In PractiSES, the message headers are signed by the sender, and only the signed information is presented to the receiver. It does not exist in S/MIME.
- Verification process of signed messages in PractiSES includes also controlling the name, surname and e-mail address of sender. It does not exist in S/MIME.

- PractiSES supports sending secure (signed and/or encrypted) e-mails to multiple recipients.
- PractiSES is deserving its name, since it is more practical than both S/MIME and PGP and it is secure enough.

4. IMPLEMENTATION ISSUES

While implementing PractiSES, some cryptographic structures and functions are used. Details of these structures and functions, and then deployment of the system will be covered in this section.

4.1. Cryptographic Functions of PractiSES

There several cryptographic functions that are implemented and used in PractiSES. All the security functions are gathered within "PractiSES.crypto" package. Some of the critical cryptographic functions of PractiSES are signature/verification, encryption/decryption, HMAC, key-pair generation, etc.

4.1.1. Secret Key Generator

To generate random numbers, PractiSES uses pseudo random number generator function of "SecureRandom" object of "javax.crypto" library. Generated random number is used as an input (salt) to the "SecretKeyFactory" object. "SecretKeyFactory" generates a secret key for 3-DES (Triple DES).

4.1.2. Key Pair Generator

PractiSES has a "PractiSES.crypto.SESKeyPairGenerator" object to generate 2048-bit key pair (i.e. public key and private key) for RSA cryptosystem.

4.1.3. Signature and Verification

PractiSES has a “PractiSES.crypto.SESSignature” object to sign a message and verify a signature. “SESSignature” uses SHA-1 algorithm to get 160-bit digest from a message and uses RSA algorithm to sign that digest. While signing, “SESSignature” takes message and private key as an input and finds out the signature as an output. Reversely in verification, it takes signature, message and public key as inputs and returns valid/invalid decision as an output.

4.1.4. Symmetric Encryption/Decryption

PractiSES uses its “PractiSES.crypto.TripleDESSymmetricEncryption” object to encrypt/decrypt cleartext/ciphertext. While encrypting, it uses cleartext and secret key and finds out ciphertext. Reversely, it uses ciphertext and secret key and finds out the cleartext while decrypting. Algorithm used in both encryption and decryption is 3-DES (Triple-DES).

4.1.5. Public Key Encryption/Decryption

PractiSES uses “PractiSES.crypto.RSAEncryption” object to encrypt/decrypt cleartext/ciphertext. It uses cleartext and recipients public key to encrypt the message that yields ciphertext. While decrypting, it uses ciphertext and corresponding private key to find out the original cleartext. Algorithm used in both public key encryption and decryption is RSA.

4.1.6. Hash-based MAC

To generate an integrity check value with a one-way hash function, HMAC is the most common method of using a shared secret. PractiSES has a "PractiSES.crypto.SESEncDecMac" object to calculate HMAC [41] of a message. HMAC function produces fixed length message authentication code. SHA-1 algorithm is used as a one-way hash function

4.2. Cryptographic Structures of PractiSES

Cryptographic structures that are used for message transmission in PractiSES are discussed in this section.

4.2.1. Signed Message

A Signed message consists of two parts: one is the message itself and the other is the digital signature over the message. Before signing a message, message digest (a.k.a. message hash) is produced from message using SHA-1 algorithm. SHA-1 always generates 160-bit digest from the variable length message. The digest of message is used in signing process. RSA algorithm (2048-bits) is used while signing. Message and the signature of message constitute signed message structure of PractiSES as shown in Figure 22.

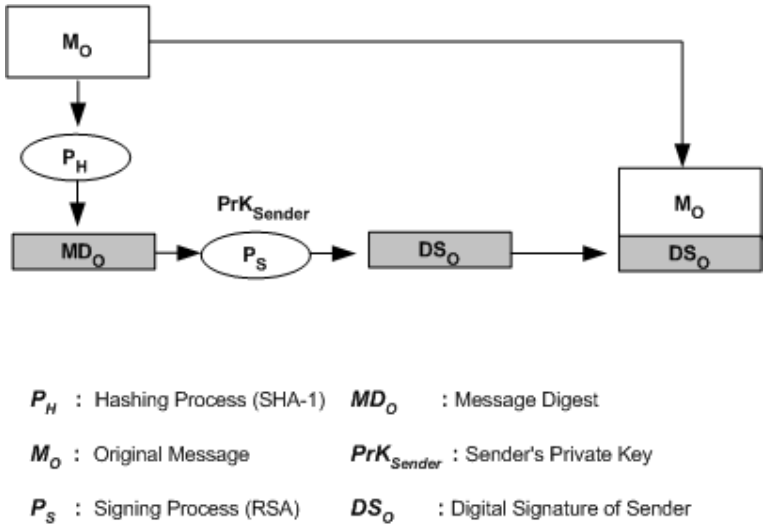

Figure 22. Signed Message

Note that, whole message is signed in PractiSES even e-mail header. The content of the signed message is shown in Figure 39 of Appendix D.

4.2.2. Encrypted Message

PractiSES utilizes digital enveloping mechanism while encrypting a message. An encrypted message consists of two parts: one is the message that is encrypted with a random secret key and the other is the secret key that is encrypted with the public key of receiver. Before encrypting a message, a secret key is randomly selected and it is used for encrypting the message. Besides, secret key is encrypted with a public key of receiver. 3-DES algorithm is used for encrypting the message and the RSA algorithm is used for encrypting the secret key. The encrypted key and the encrypted message constitute the encrypted message structure of PractiSES as shown in Figure 23.

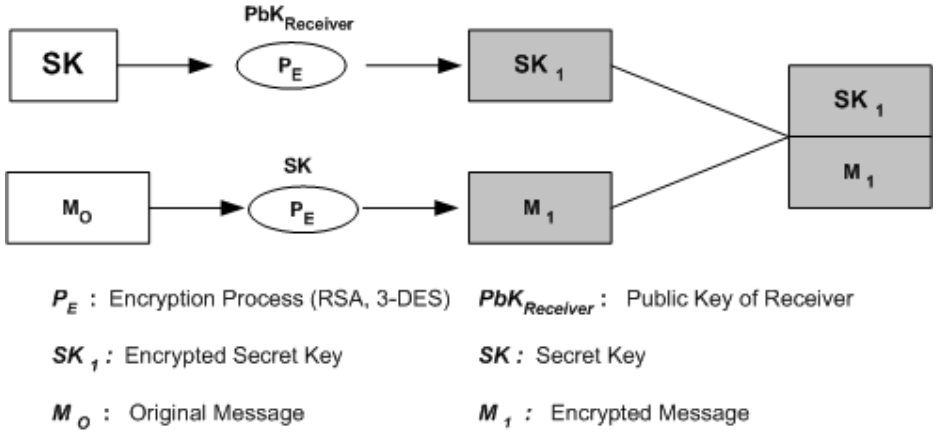

Figure 23. Encrypted Message

The content of the encrypted message is shown in Figure 40 of Appendix D.

4.2.3. Signed and Encrypted Message

Signed and encrypted message of PractiSES is the combination of signature and encryption procedures. Signed and encrypted message consists of three parts: (i) message that is encrypted with a random secret key; (ii) secret key that is encrypted with public key of the receiver, and (iii) the digital signature of message. 3-DES algorithm is used for encrypting the message and the RSA algorithm is used for encrypting the secret key and signing the message. Before encrypting a message, a secret key is randomly selected and it is used for encrypting the message. Besides, the secret key is encrypted with the public key of receiver. At the end, sender digitally signs hash of the message. SHA-1 algorithm is used to get hash of the message. The encrypted key, an encrypted message and digital signature of the original message constitute the signed and encrypted message structure of PractiSES as shown in Figure 24.

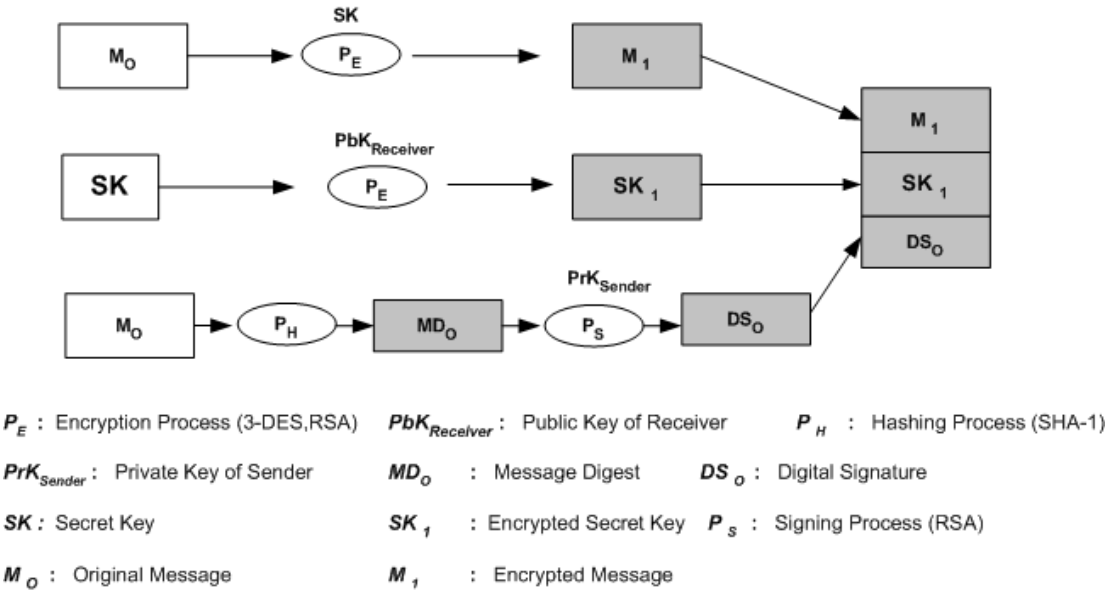

Figure 24. Signed and Encrypted Message

The content of the signed and encrypted message is shown in Figure 41 of Appendix D. Note that, all information is either encrypted or signed.

4.2.4. Message Authentication Coded (MACed) Message

PractiSES uses MAC to provide integrity in transmitted messages in the absence of digital signatures. A MACed message consists of two parts: one is the message itself and the other is MAC of the message. A shared-secret is used while producing MAC of the message. Shared-secret is appended to the end of the message and then this combination is given to the MAC function. Message and MAC result constitute the MACed message structure in PractiSES as shown in Figure 25.

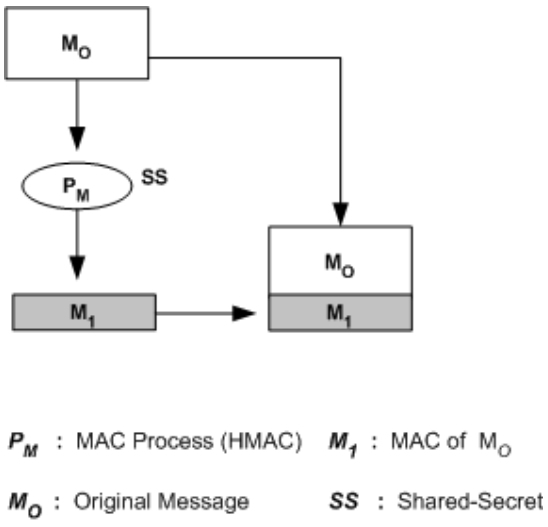

Figure 25. MACed Message

4.3. Requirements

In this section, necessary libraries, database and system properties are explained. Basically, PractiSES needs MySQL as a database tool, Java Cryptographic Environment (JCE1.2.2) as the cryptographic base and Java Mail as an e-mail client.

4.3.1. System Requirements

In order to run PractiSES Client, it is enough for any personal computer to have java runtime environment (over jre1.3). However, PractiSES Server should be relatively faster computer, because the server will be able to serve multiple users at the same time and needs to have java database connectivity (jdbc) with MySQL Server.

4.3.2. MySQL

MySQL is a free database management tool. PractiSES uses it as the public key storage. MySQL server should run on the same machine with PractiSES Server.

4.3.3. Java Cryptographic Environment (JCE)

JCE provides many cryptographic functions that are necessary in PractiSES. 3-DES, SHA-1 and RSA algorithms [42, 43, 44, 45] are used from JCE.

4.3.4. Java Mail

Java Mail is a free e-mail tool. PractiSES Clients use its e-mail sending/receiving methods to send/receive e-mails.

4.4. Deployment of System

PractiSES Server and Client softwares will be available in the project's web page [50]. In order to use PractiSES, MySQL Server, PractiSES Server and PractiSES Client applications should be downloaded. Initially, both the PractiSES Server and MySQL Server are set up. While server is being set up, key pair of PractiSES Server is generated and the public key storage tables are created on MySQL. The password for private key should be known by the "administrator". Administrator should make the server's public key reachable to all group members from a web or an ftp site. Lastly, from PractiSES Server's GUI, administrator should make necessary settings such as mail server IP, database user name, password etc.

In the client side, client first downloads the client application and sets it up to his/her PC. Consequently, client downloads the server's public key from a web site that server informed. Secondly, he/she performs security settings such as defining a profile, mail server IP etc. Lastly, he/she triggers the "InitKeySet" protocol from his/her client GUI and uploads his/her public key to public key storage. Note that, PractiSES Server and public key storage must be located in a physically secure room. The system is now ready for all registered and initialized users to exchange secure emails. Other users (uninitialized users) can use the client software for normal e-mail exchange.

5. CONCLUSIONS AND FUTURE WORK

PractiSES proposes a new key distribution and management mechanism for e-mail applications. It takes advantage over S/MIME by escaping use of certificates and over PGP by less complicated trust mechanism based on a trusted third party. It provides key distribution in real-time by sending requested public keys with trusted third party's signature on it. It is a practical key distribution mechanism that does not require any kind of revocation control.

Current version of PractiSES is designed for closed-group communication. In PractiSES, it is presumed that the members of closed-group is already registered to group registry. Therefore, the users just need to download the client application software and upload newly generated public keys to the public key storage.

PractiSES proposes signing e-mail headers before transmission. That provides integrity for the information in the header. Such an integrity is not provided in S/MIME. Besides, PractiSES proposes comparing the name, surname and e-mail address of sender in the signed header with the values in the corresponding public key storage. Such control is not provided in S/MIME either.

PractiSES has a practical and easy-to-use GUI. User-transparent nature of encrypting/decrypting and signing/verifying a message in PractiSES is its another plus. Besides, PractiSES supports both English and Turkish. All the GUI components and application messages are displayed in selected language.

The support for sending secure (signed and/or encrypted) e-mails to multiple recipients also makes PractiSES as an attractive system.

PractiSES is a freeware system. Moreover, there are no export restrictions in PractiSES.

We plan some future works to make PractiSES more successful. These are explained below.

In future, PractiSES servers will provide cross-trust between each other. The first version of PractiSES is for internal use in a group. Different users that belong to different PractiSES Servers (different groups) will be able to exchange signed and/or encrypted messages in the future.

In this version, signed messages with obsolete private keys cannot be verified since public key storage distributes only current public key to the requester. In the next version, a key history module will be added. This module will be used to distribute old public keys as well.

We will develop plug-ins for popular e-mail client programs to allow PractiSES users continue to use their popular e-mail client programs.

In order to increase the security of the system, the private key of PractiSES Server can be kept in a physical device such as smart-card or e-token. Since this is only for the server, it will not change the monetary cost for the clients.

Security model of PractiSES can be standardized or documented as an IETF Internet Draft. Especially after Cross-Trust module, it will be necessary to standardize the structures and the model.

6. APPENDIX A: LIST OF ACRONYMS

CA	Certification Authority
CRL	Certificate Revocation List
ARL	Authority Revocation List
DAP	Directory Access Protocol
LDAP	Lightweight Directory Access Protocol
DES	Data Encryption Standard
HTTP	HyperText Transfer Protocol
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Standards Organization
ITU	International Telecommunications Union
MD4	Message Digest 4
MD5	Message Digest 5
NIST	National Institute of Standards and Technology
PGP	Pretty Good Privacy
PKC	Public Key Crypto(graphy)
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
PKIX	Internet Public Key Infrastructure using X.509 certificates
RFC	Request for Comments
RSA	Rivest, Shamir, Adleman

MIME	Multipurpose Internet Mail Extensions
S/MIME	Secure/Multipurpose Internet Mail Extensions
SHA-1	Secure Hash Algorithm 1
S-HTTP	Secure HyperText Transfer Protocol
SSL	Secure Socket Layer
TLS	Transport Layer Security

7. APPENDIX B: SECURITY DIALOGS

Figure 26. Starting PractiSES Server Dialog

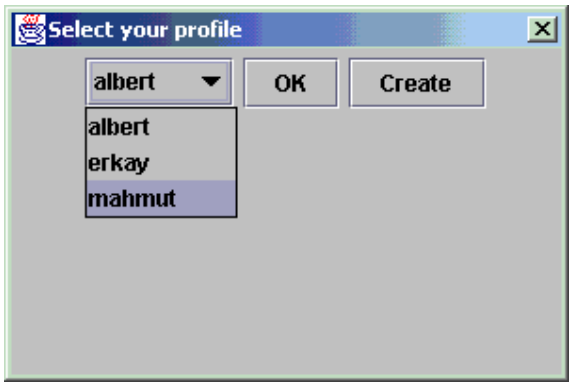

Figure 27. Starting PractiSES Client Dialog

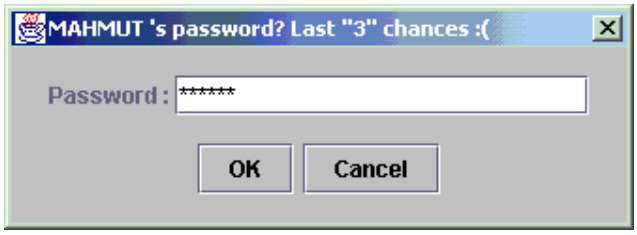

Figure 28. User Login Dialog

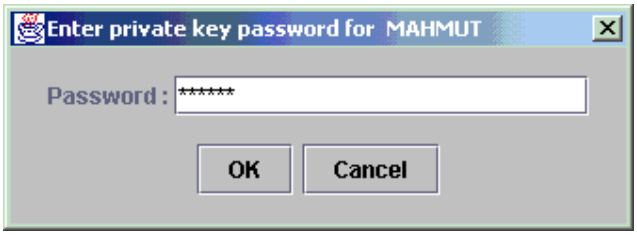

Figure 29. Message Signing/Decrypting Dialog

8. APPENDIX C: MESSAGE SENDING/RECEIVING

8.1. Sending

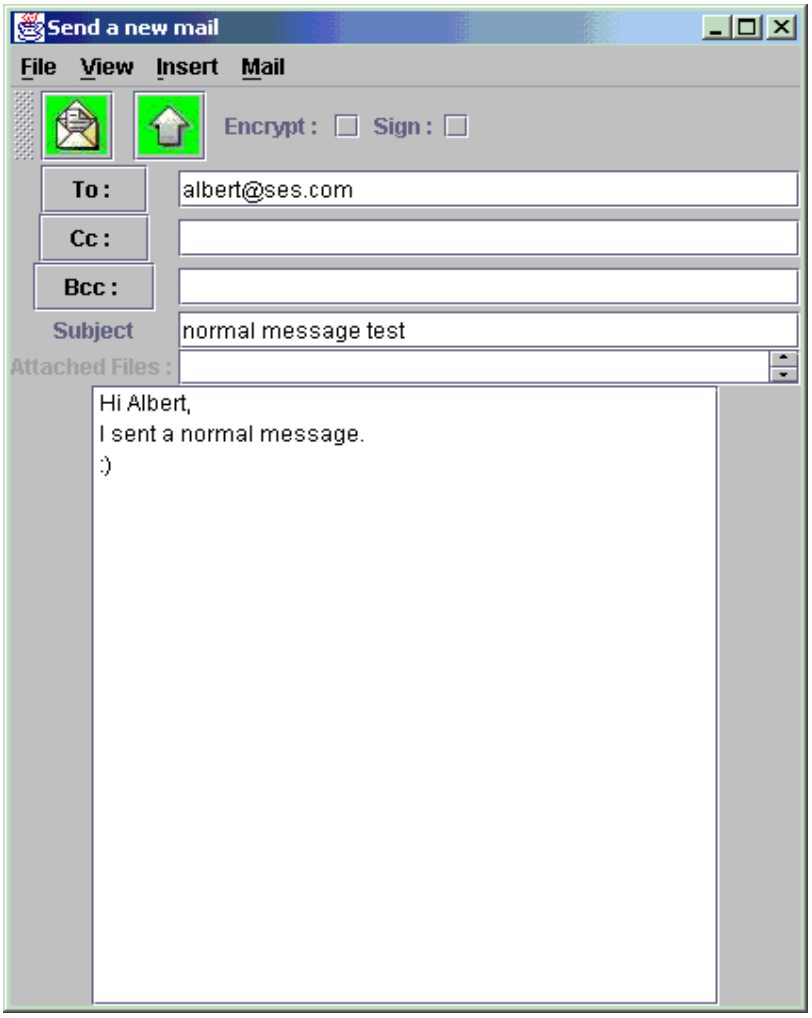

Figure 30. Normal Message

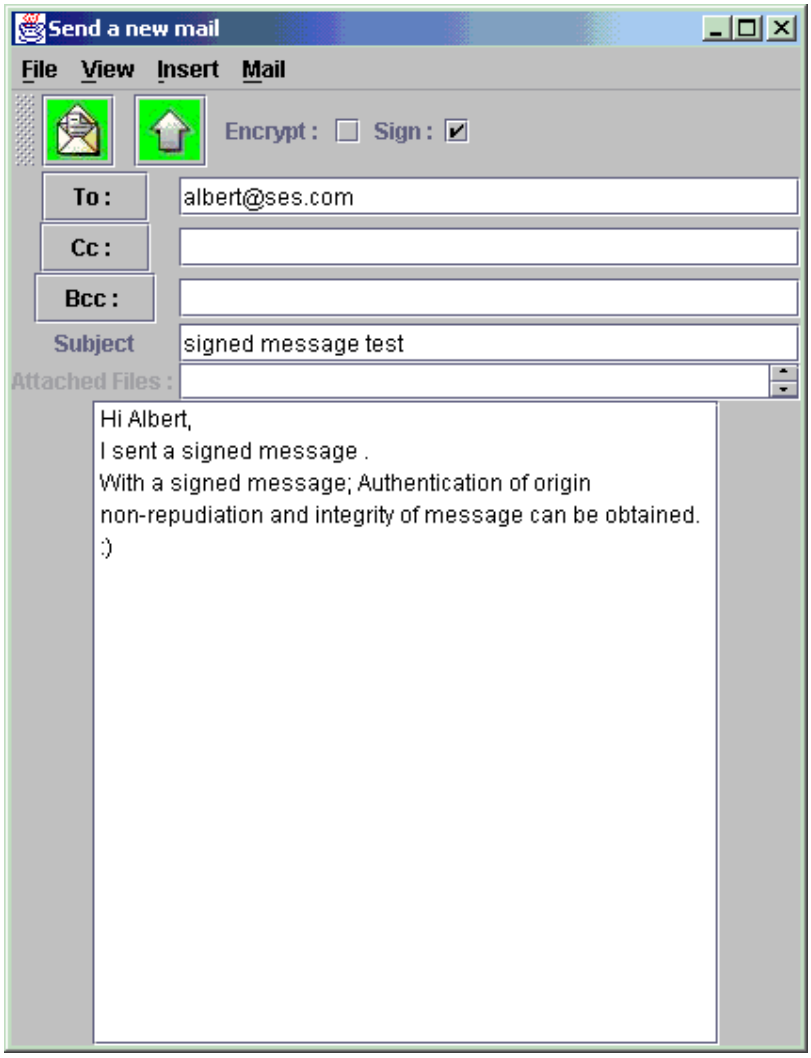

Figure 31. Signed Message

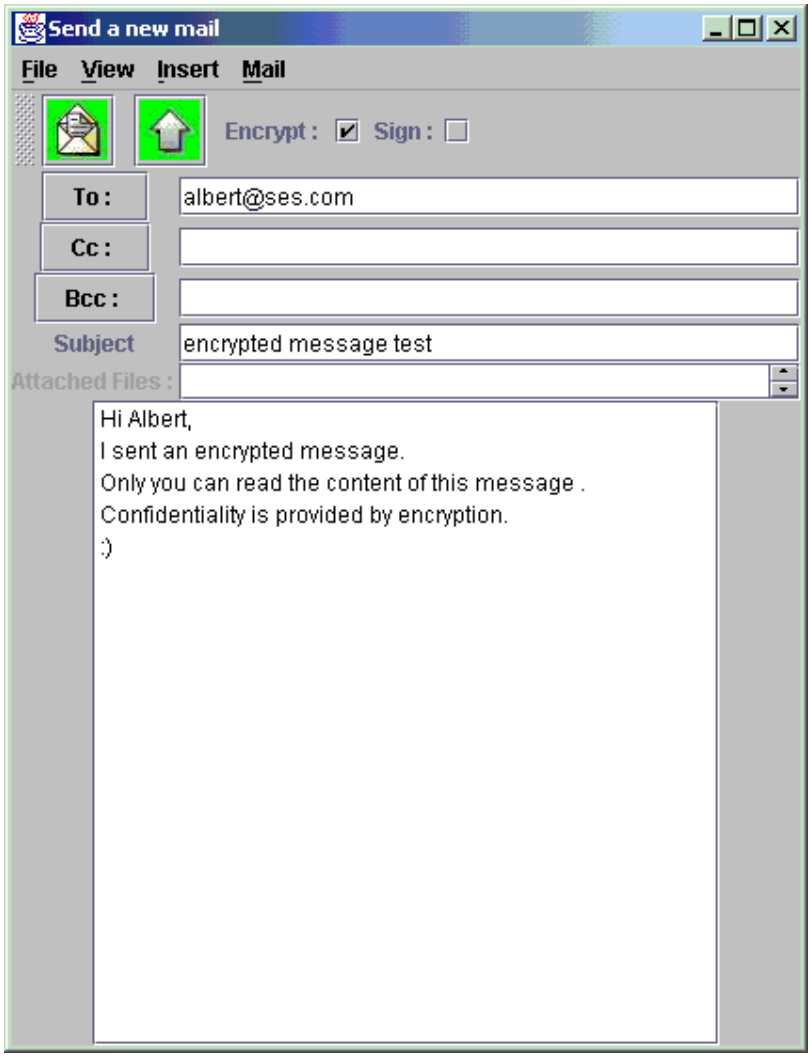

Figure 32. Encrypted Message

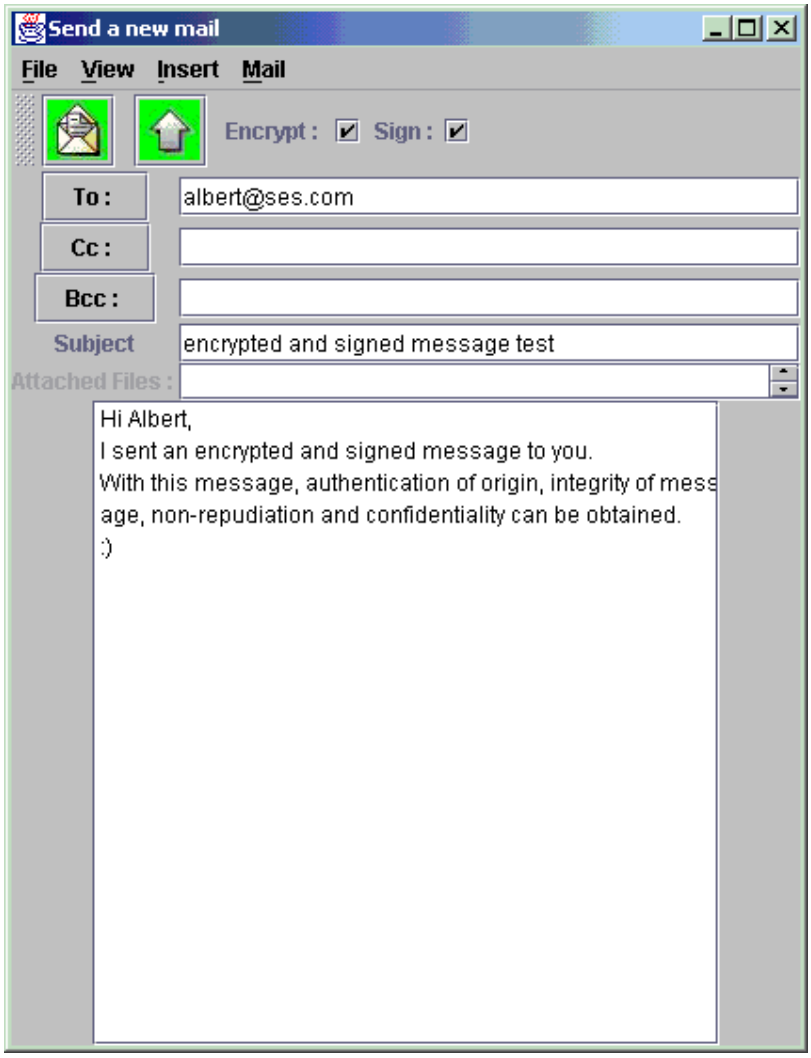

Figure 33. Encrypted and Signed Message

8.2. Receiving

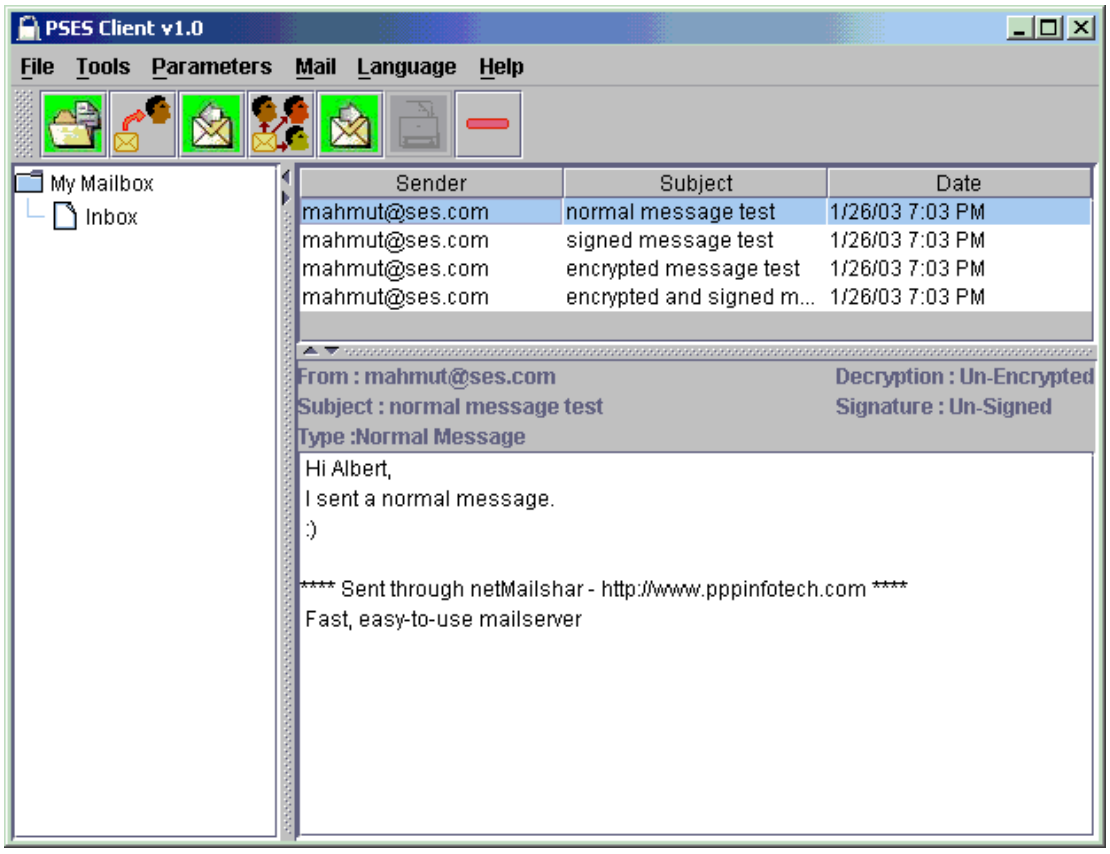

Figure 34. Normal Message

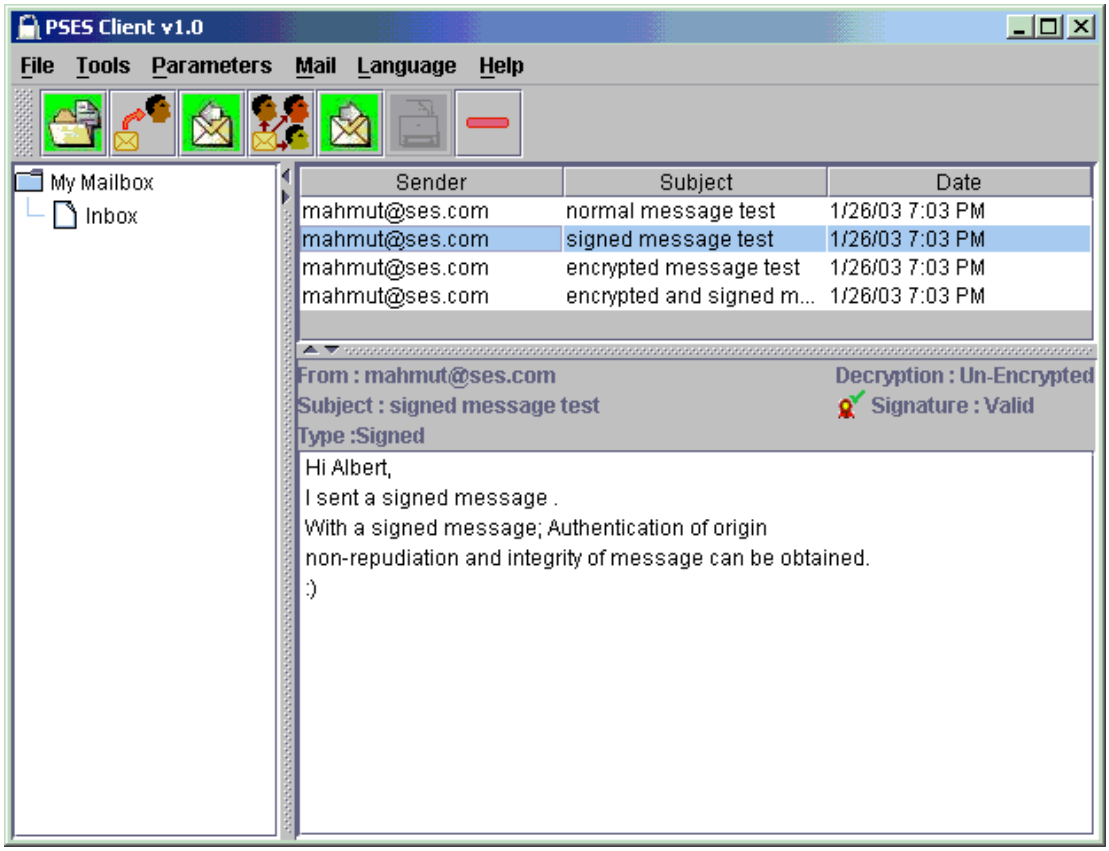

Figure 35. Signed Message

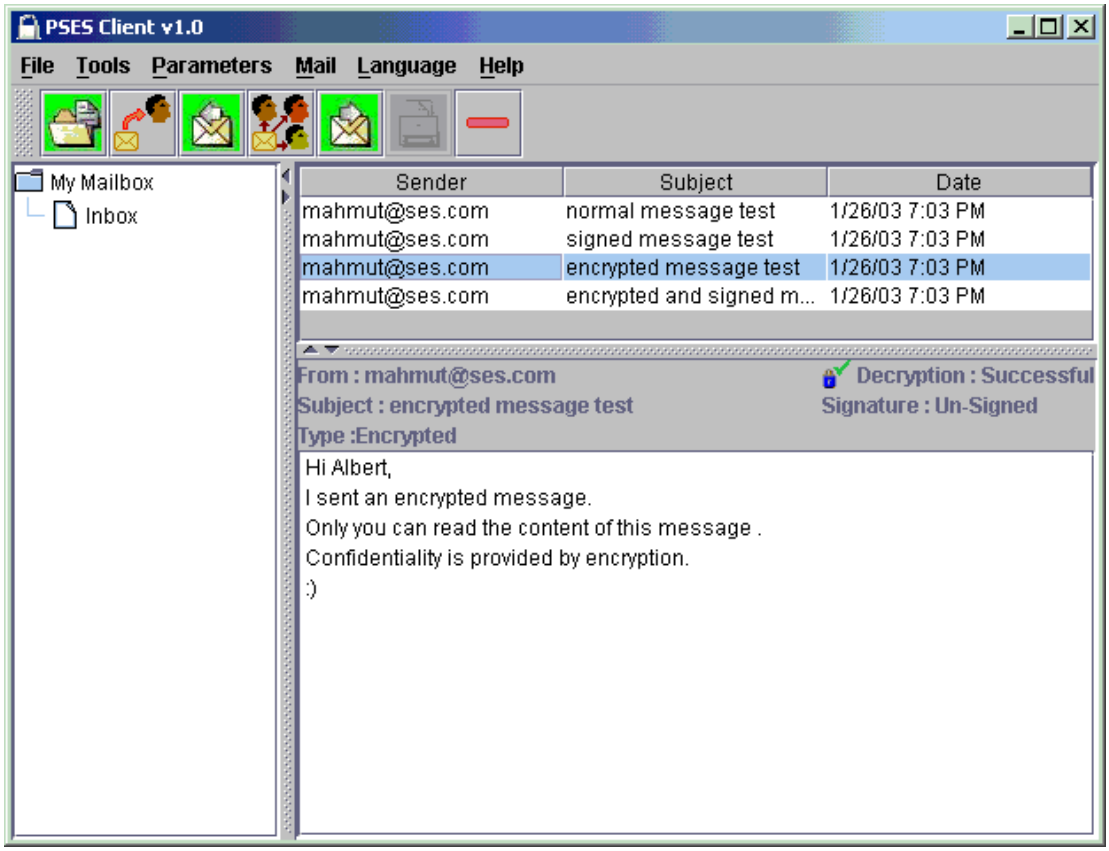

Figure 36. Encrypted Message

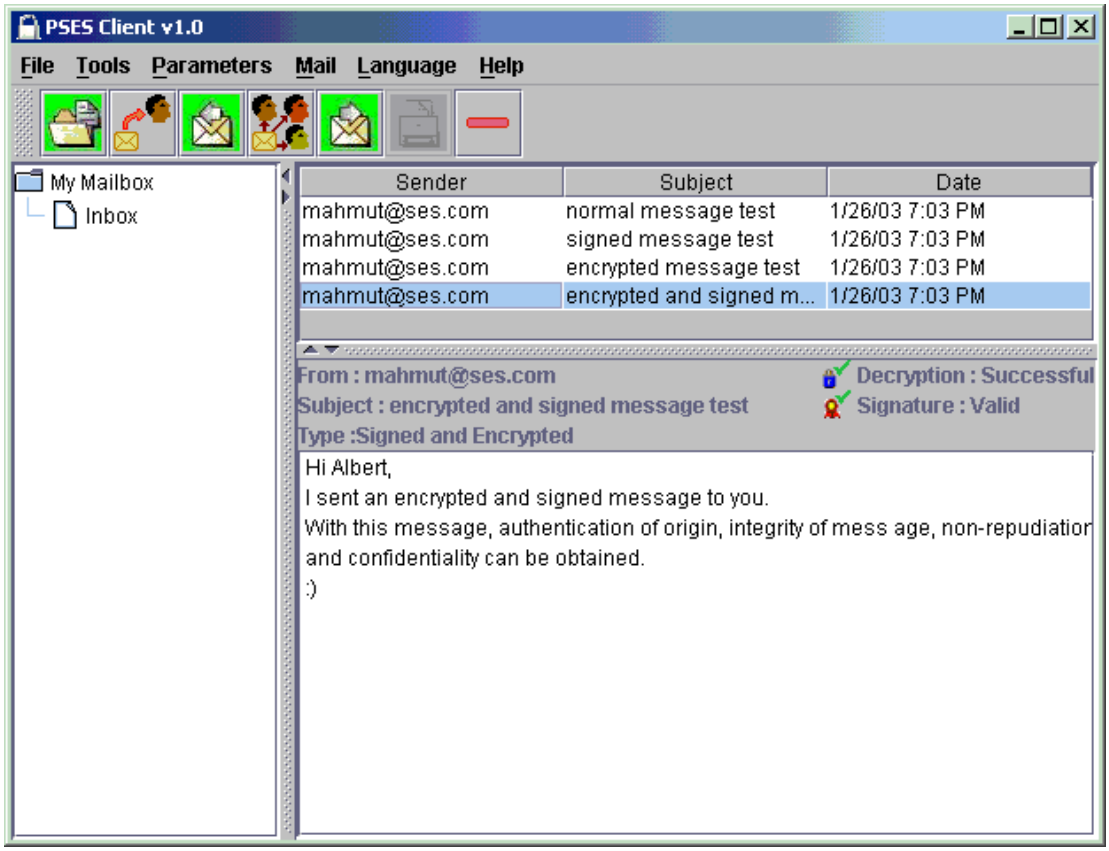

Figure 37. Encrypted and Signed Message

9. APPENDIX D: MESSAGE STRUCTURES

Figure 38. Normal Message

Figure 39. Signed Message

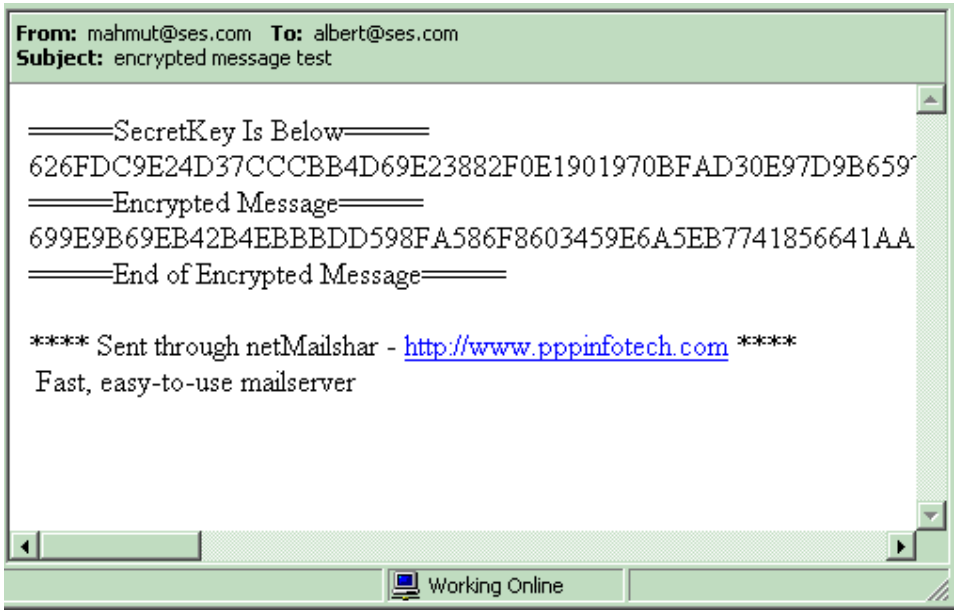

Figure 40. Encrypted Message

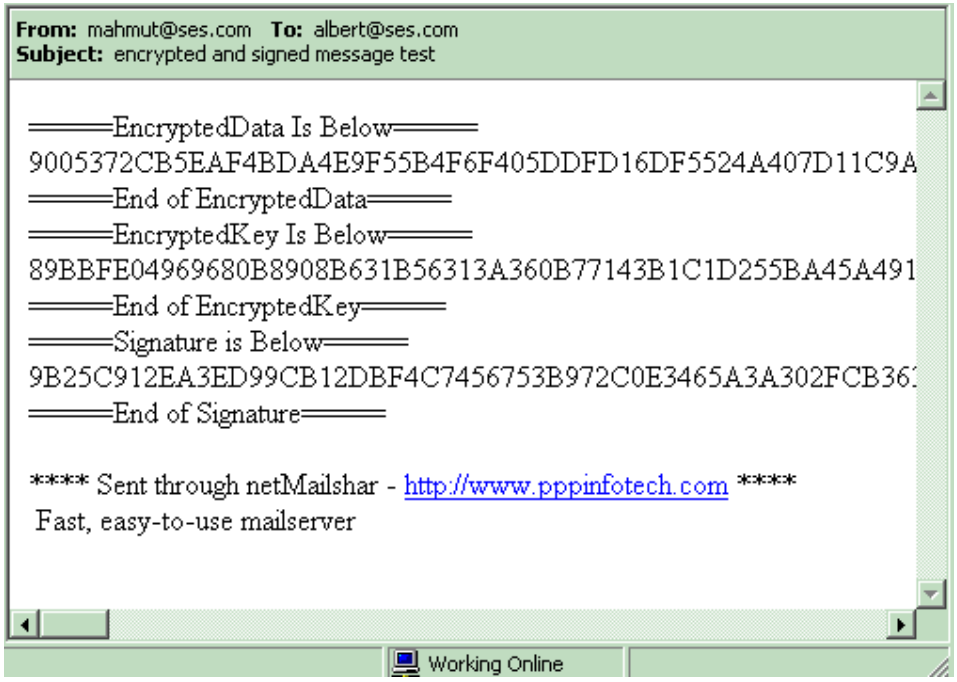

Figure 41. Encrypted and Signed Message

10.REFERENCES

1. 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.

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

3. Network Associates, *"PGP Freeware for Windows 95, Windows 98,Windows NT, Windows 2000 & Windows Millennium User's Guide Version 7.0",* available from http://www.pgpi.org/doc/guide/7.0/en/win/, 2001.

4. S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka, *"S/MIME Version 2 Message Specification",* RFC 2311, March 1998.

5. ITU-T, *"The Directory: Overview of Concepts, Models and Services",* Recommendation X.500, 2001.

6. ITU-T, *"Information technology - Open Systems Interconnection -The Directory: Authentication framework",* Recommendation X.509, 1997.

7. U.S. Department of Commerce, *"Computer Data Authentication",* Federal Information Processing Standards Publication 180-1,1995.

8. Stallings, W., *"Cryptography and Network Security",* Prentice Hall, New Jersey, 2003.

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

10. P. Chown, *"Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)",* RFC 3268, June 2002.

11. S. Teiwes, P. Hartmann, D. Kuenzi, *"Use of the IDEA Encryption Algorithm in CMS",* RFC 3058, February 2001.

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

13. 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.

14. Massachusetts Institute of Technology, http://web.mit.edu/

15. V.S. Miller, *"Use of elliptic curves in cryptography",* Advances in Cryptology - Crypto '85*,* Springer-Verlag, 417-426, 1986.

16. T. ElGamal, *"A public-key cryptosystem and a signature scheme based on discrete logarithms",* IEEE Transactions on Information Theory **31**, 469-472, 1985.

17. Rivest, R., *"The MD5 Message Digest Algorithm",* RFC 1321, April 1992

18. Trappe W., L.C. Washington, *"Introduction to Cryptography with Coding Theory",* 1997

19. U.S. Department of Commerce, *"Digital Signature Standard (DSS)",* Federal Information Processing Standards Publication 186, 1994.

20. NIST, *"Digital Signature Standard (DSS)",* FIPS PUB 186-2, 2000 January 27.

21. 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.

22. Housley, R., W. Ford,W.Polk, and D. Solo, *"Internet X.509 Public Key Infrastructure Certificate and CRL profile",* RFC 2459, 1999.

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

24. Dierks, T., and C. Allen, *"The TLS Protocol Version 1.0",* RFC 2246, 1999.

25. 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.

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

27. ITU-T, *"The Directory: Public Key and Attribute Certificate Frameworks",* Recommendation X.509, 2001.

28. Tebbut, J., *"Guideleness for the evaluation of X.500 Directory Products",* NIST Special Publication 500-228, May 1995

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

30. RSA Security, http://www.rsasecurity.com/standards/

31. Elkins, M., *"MIME Security with Pretty Good Privacy (PGP)",* RFC 2015, 1996.

32. Atkins, D., W. Stallings, and P. Zimmermann, *"PGP Message Exchange Formats",* RFC 1991, 1996.

33. J. Callas, L. Donnerhacke, H. Finney, R. Thayer, *"OpenPGP Message Format",* RFC 2440, November 1998.

34. Network Associates, ftp://ftp.pgpi.org/pub/pgp/7.0/docs/english/IntroToCrypto.pdf

35. Ellison C., and B. Schneier, *"Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure",* Computer Security Journal, vol. 16, no. 1, pp. 1-7, 2000.

36. Wheeler A., and L. Wheeler, *"Payment, Security & Internet References",* http://www.garlic.com/~lynn/

37. McConnel S., *"Software Project Survival Guide",* 1998.

38. Wirfs-Brock, Rebecca, B. Wilkerson, and L. Wiener, *"Designing Object Oriented Software",* 1990.

39. Budd T., *"Understanding Object-Oriented Programming with Java",* 1998.

40. Stallings W., *"Network Security Essentials",* 1999.

41. Krawczyk, H., M. Bellare, and R. Canetti, *"HMAC: Keyed-Hashing for Message Authentication",* RFC2104, 1997.

42. RSA Laboratories. *"PKCS #1: RSA Encryption Standard. Version 2.0",* October 1998.

43. J. Jonsson, B. Kaliski, *"Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1",* RFC 3447, February 2003.

44. RSA Laboratories, *"PKCS #7: Cryptographic Message Syntax Standard, version 1.5",* RSA Laboratories Technical Note, available from ftp://ftp.rsasecurity.com/pub/pkcs/doc/pkcs-7.doc, November 1, 1993.

45. Burton S. Kaliski Jr., Ph.D. and Kevin W. Kingdon, *"Extensions and Revisions to PKCS #7",* An RSA Laboratories Technical Note, available from ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-7/pkcs-7v16.pdf, May 13, 1997.

46. RSA Laboratories, *"PKCS #10 v1.7: Certification Request Syntax Standard",* available from ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf, May 26, 2000.

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

48. Wiener, M. J., *"Performance Comparison of Public-Key Cryptosystems",* RSA Laboratories' Cryptobytes, vol. 4, no. 1, pp. 1 – 5, 1998.

49. Levi A., and C. K. Koc, *"Risks in email security",* Communications of the ACM, vol. 44 no. 8, pp.112, August 2001.

50. PractiSES Project of Mahmut Özcan and Dr. Albert Levi, http://students.sabanciuniv.edu/~mahmut/SESDocuments/PractiSES.html

51. Levi A., and M.Özcan, *"Açık Anahtar Altyapısı Neden Zordur?"* Bilişim 2002 – TBD 19. Bilişim Kurultayı, İstanbul, September 2002.

10.REFERENCES

1. 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.

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

3. Network Associates, *"PGP Freeware for Windows 95, Windows 98,Windows NT, Windows 2000 & Windows Millennium User's Guide Version 7.0",* available from http://www.pgpi.org/doc/guide/7.0/en/win/, 2001.

4. S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka, *"S/MIME Version 2 Message Specification",* RFC 2311, March 1998.

5. ITU-T, *"The Directory: Overview of Concepts, Models and Services",* Recommendation X.500, 2001.

6. ITU-T, *"Information technology - Open Systems Interconnection -The Directory: Authentication framework",* Recommendation X.509, 1997.

7. U.S. Department of Commerce, *"Computer Data Authentication",* Federal Information Processing Standards Publication 180-1,1995.

8. Stallings, W., *"Cryptography and Network Security",* Prentice Hall, New Jersey, 2003.

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

10. P. Chown, *"Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)",* RFC 3268, June 2002.

11. S. Teiwes, P. Hartmann, D. Kuenzi, *"Use of the IDEA Encryption Algorithm in CMS",* RFC 3058, February 2001.

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

13. 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.

14. Massachusetts Institute of Technology, http://web.mit.edu/

15. V.S. Miller, *"Use of elliptic curves in cryptography",* Advances in Cryptology - Crypto '85*,* Springer-Verlag, 417-426, 1986.

16. T. ElGamal, *"A public-key cryptosystem and a signature scheme based on discrete logarithms",* IEEE Transactions on Information Theory **31**, 469-472, 1985.

17. Rivest, R., *"The MD5 Message Digest Algorithm",* RFC 1321, April 1992

18. Trappe W., L.C. Washington, *"Introduction to Cryptography with Coding Theory",* 1997

19. U.S. Department of Commerce, *"Digital Signature Standard (DSS)",* Federal Information Processing Standards Publication 186, 1994.

20. NIST, *"Digital Signature Standard (DSS)",* FIPS PUB 186-2, 2000 January 27.

21. 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.

22. Housley, R., W. Ford,W.Polk, and D. Solo, *"Internet X.509 Public Key Infrastructure Certificate and CRL profile",* RFC 2459, 1999.

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

24. Dierks, T., and C. Allen, *"The TLS Protocol Version 1.0",* RFC 2246, 1999.

25. 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.

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

27. ITU-T, *"The Directory: Public Key and Attribute Certificate Frameworks",* Recommendation X.509, 2001.

28. Tebbut, J., *"Guideleness for the evaluation of X.500 Directory Products",* NIST Special Publication 500-228, May 1995

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

30. RSA Security, http://www.rsasecurity.com/standards/

31. Elkins, M., *"MIME Security with Pretty Good Privacy (PGP)",* RFC 2015, 1996.

32. Atkins, D., W. Stallings, and P. Zimmermann, *"PGP Message Exchange Formats",* RFC 1991, 1996.

33. J. Callas, L. Donnerhacke, H. Finney, R. Thayer, *"OpenPGP Message Format",* RFC 2440, November 1998.

34. Network Associates, ftp://ftp.pgpi.org/pub/pgp/7.0/docs/english/IntroToCrypto.pdf

35. Ellison C., and B. Schneier, *"Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure",* Computer Security Journal, vol. 16, no. 1, pp. 1-7, 2000.

36. Wheeler A., and L. Wheeler, *"Payment, Security & Internet References",* http://www.garlic.com/~lynn/

37. McConnel S., *"Software Project Survival Guide",* 1998.

38. Wirfs-Brock, Rebecca, B. Wilkerson, and L. Wiener, *"Designing Object Oriented Software",* 1990.

39. Budd T., *"Understanding Object-Oriented Programming with Java",* 1998.

40. Stallings W., *"Network Security Essentials",* 1999.

41. Krawczyk, H., M. Bellare, and R. Canetti, *"HMAC: Keyed-Hashing for Message Authentication",* RFC2104, 1997.

42. RSA Laboratories. *"PKCS #1: RSA Encryption Standard. Version 2.0",* October 1998.

43. J. Jonsson, B. Kaliski, *"Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1",* RFC 3447, February 2003.

44. RSA Laboratories, *"PKCS #7: Cryptographic Message Syntax Standard, version 1.5",* RSA Laboratories Technical Note, available from ftp://ftp.rsasecurity.com/pub/pkcs/doc/pkcs-7.doc, November 1, 1993.

45. Burton S. Kaliski Jr., Ph.D. and Kevin W. Kingdon, *"Extensions and Revisions to PKCS #7",* An RSA Laboratories Technical Note, available from ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-7/pkcs-7v16.pdf, May 13, 1997.

46. RSA Laboratories, *"PKCS #10 v1.7: Certification Request Syntax Standard",* available from ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf, May 26, 2000.

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

48. Wiener, M. J., *"Performance Comparison of Public-Key Cryptosystems",* RSA Laboratories' Cryptobytes, vol. 4, no. 1, pp. 1 – 5, 1998.

49. Levi A., and C. K. Koc, *"Risks in email security",* Communications of the ACM, vol. 44 no. 8, pp.112, August 2001.

50. PractiSES Project of Mahmut Özcan and Dr. Albert Levi, http://students.sabanciuniv.edu/~mahmut/SESDocuments/PractiSES.html

51. Levi A., and M.Özcan, *"Açık Anahtar Altyapısı Neden Zordur?"* Bilişim 2002 – TBD 19. Bilişim Kurultayı, İstanbul, September 2002.