# TRUSTED MAIL GATEWAY

**A THESIS SUBMITTED TO**
**THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**
**OF**
**MIDDLE EAST TECHNICAL UNIVERSITY**

**BY**

**ERKUT SİNAN AYLA**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**
**FOR**
**THE DEGREE OF MASTER OF SCIENCE**
**IN**
**COMPUTER ENGINEERING**

**DECEMBER 2004**

Approval of the Graduate School of Natural and Applied Sciences.

_____

Prof. Dr. Canan Özgen

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Ayşe Kiper

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Dr. Attila Özgit

Supervisor

Examining Committee Members

| | | |
|---|---|---|
| Prof. Dr. Semih Bilgen | (METU,EEE) | _____ |
| Dr. Attila Özgit | (METU,CENG) | _____ |
| Prof. Dr. Adnan Yazıcı | (METU,CENG) | _____ |
| Assoc. Prof. Dr. Halit Oğuztüzün | (METU,CENG) | _____ |
| Dr. Onur Tolga Şehitoğlu | (METU,CENG) | _____ |

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

**Erkut Sinan AYLA**

# ABSTRACT

## TRUSTED MAIL GATEWAY

**AYLA, Erkut Sinan**

**M.S., Department of Computer Engineering**

**Supervisor: Dr. Attila Özgit**

**December 2004, 135 pages**

The study aims to make contribution to solve mail security problems. The Trusted Mail Gateway provides a domain with the basic security services that are message integrity, confidentiality, non-repudiation, origin authentication and availability while message (e-mail) being delivered through the Internet. It generates S/MIME digital signatures and performs S/MIME encryption on behalf of the domain using secret key cryptography and public-key techniques and generating Cryptographic Message Syntax (CMS) data to provide origin authenticity, integrity and

confidentiality. It applies anti-virus control and protection, spam filtering and content check to both incoming mails to the domain and outgoing mails from the domain to prevent attacks against availability. Also, the Trusted Mail Gateway provides intra-domain security. It keeps e-mail messages in corresponding mailboxes as encrypted messages. Trusted Mail Gateway processes all the mails passing through and records processing results in database as notary information. Moreover, it establishes trust relations with other trusted domains and exchanges notary information with them via a secure channel.

Keywords: S/MIME, security services, mail protocols, domain, notary

# ÖZ

## GÜVENİLİR E-POSTA GEÇİDİ

**AYLA, Erkut Sinan**

**Yüksek Lisans, Bilgisayar Mühendisliği Bölümü**

**Tez Yöneticisi: Dr. Attila Özgit**

**Aralık 2004, 135 sayfa**

Güvenli E-posta Geçidi çalışmasının amacı e-posta sistemlerindeki güvenlik sorunlarına çözüm bulunmasına katkıda bulunmaktır. Güvenli E-posta Geçidi, içinde bulunduğu alana e-posta İnternet yoluyla gönderilirken, e-posta bütünlüğü, gizlilik, inkar edilemezlik, kimlik doğrulama ve elverişlilikten oluşan temel güvenlik servislerini sağlamaktadır. Geçit Kriptografik Mesaj Sözdizimi veri yapılarını üreterek ve gizli anahtarlı kriptografi ve açık anahtarlı yordamları kullanarak kimlik doğrulama, e-posta bütünlüğü sağlama ve gizlilik amacıyla alan

adına S/MIME sayısal imzası atmakta ve şifreleme yapmaktadır. Geçit alınan ve gönderilen e-postalara virus koruması, içerik kontrolü ve amaçsız e-posta kontrolünü elverişliliğe karşı yapılan saldırıları önlemek için yapmaktadır. Geçit ayrıca alan içi e-posta güvenliğini de sağlamaktadır. Geçit uygun e-posta kutularındaki iletileri şifreli olarak tutmaktadır. Güvenilir İleti Geçidi üzerinden geçen tüm e-postaları işler ve işlem sonuçlarını veritabanına noterlik verisi olarak kaydeder. Bunun dışında, diğer alanlarla güven ilişkileri kurar ve onlarla güvenli protokol aracılığı ile noterlik bilgisi alışverişi yapar.

Anahtar Kelimeler: S/MIME, güvenlik servisleri, posta protokolleri, alan, noterlik

## ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Attila Özgit for all his guidance, support and friendship during this study.

## TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER I

## INTRODUCTION

Electronic mail (E-mail) has been the most widely available and pervasively used Internet service for many years. Nowadays, the World Wide Web has threatened this dominance, but e-mail still holds a crucial role. Compared to other Internet services, e-mail reaches more Internet users. In many offices and enterprises, e-mail is used to replace numerous paper notices, paper reports and paper reminders. Also, e-mail has become widespread communication medium for people who are at distant places.

Modern Internet e-mail was originally developed on the ARPANET that was an experimental wide area network established by the government of the United States of America. Specifically, today's Internet e-mail

specifications were originated and developed for the ARPANET. Moreover, designers built a notion for a simple protocol for transferring e-mail into designated user files. E-mail in the age of ARPANET provided today's Internet e-mail with several important features. The features in question are about e-mail message format, addressing and delivery issues.

Internet e-mail performs delivery of messages between sender and receiver that should be named entities. Receiving party does not have to be immediately available for e-mail system to deliver its message. This characteristic makes e-mail different than other Internet protocols. Most protocols require sender and recipient, client and server to be on-line and available simultaneously. E-mail systems keep messages and attempt again at different points in the mail delivery process. Sending e-mail server will hold the e-mail message and try to send it later unless recipient mail server or some relay point in the delivery path is available. In order to sum up, it is claimed that e-mail systems make delivery of their messages through a staged delivery process by mail relays. This feature reflects the fact that e-mail has evolved as the first true Internet protocol. The Internet allows e-mail messages to be delivered directly between any pair of hosts using mail transfer protocols. However, staged delivery

through mail relays is still required since Internet e-mail still reaches many hosts through relays and also numerous systems have been developed that do not support all mail transport protocols but provide relaying for heterogeneous transport media and transfer protocols.

As stated in previous paragraphs, the e-mail is unlike other Internet protocols. In the e-mail system, e-mail is delivered from one client to another one, not between a client and a server. The clients compose e-mail messages to send and display messages received. The mail servers transfer e-mail messages to final destinations, intermediate mail relays or keep messages in transit. There are standard protocols for mail access, sending and transfer. The Simple Mail Transfer Protocol (SMTP) is required to send e-mail from client to sending e-mail server and to transfer e-mail between sending e-mail server and final destination e-mail server or mail relays. Also, receiving e-mail server using SMTP places e-mail messages in a mailbox file for actual recipient's address. Since it is impractical to hold SMTP agents on every workstation in a distributed environment, standard protocols are used to retrieve personal e-mail messages from a server host and to manipulate them. These protocols are the Post Office Protocol (POP) and the Internet Message Access Protocol

(IMAP). The Internet e-mail software architecture is roughly as follows: Clients accesses their mailboxes, retrieve and manipulate e-mail messages through POP or IMAP and send e-mail through SMTP. Mail servers use SMTP to accept e-mails from other domains, to copy received e-mails to recipient clients' mailboxes and to transfer e-mails to final or intermediate destinations or relays. SMTP, POP and IMAP are simple and text-oriented protocols. Protocol exchanges and e-mail message formats are based on textual data.

Prevalence, being mostly deployed and used, simplicity characteristics and text-oriented transfer and access protocols and textual data bring insecurity to the Internet e-mail. In an e-mail exchange process, receiving party cannot be certain that the e-mail is from the actual sender. The e-mail message might be fabricated by another entity on the network. Similarly, sending party is not able to make sure that intended recipient has obtained e-mail. Since e-mail is delivered over the same transport service as other Internet traffic, it is vulnerable to eavesdropping. Openness to other Internet traffic causes contents of e-mail to be changed or forged by malicious entities. Also, malicious content, codes, viruses and worms in e-mail content might enter user host through e-mail client.

Spam mails fill users' mailboxes and create unnecessary network traffic. All these result in loss of information, unavailability of network services and misspend of computing resources. As a result, secured and trusted e-mail delivery is a crucial requirement.

The thesis study titled as Trusted Mail Gateway is aimed to provide solution for security problems in e-mail delivery. The Trusted Mail Gateway offers domain security. It is compatible with existing e-mail clients and domain e-mail servers. It is architecturally between the domain mail server and the domain clients. The Trusted Mail Gateway provides domains with e-mail confidentiality, message content integrity, origin authentication, anti-virus protection, traffic logging, notary mechanism and content filtering. Also, Trusted Mail Gateway establishes trust relations with other domains deploying Trusted Mail Gateway. An enterprise that has geographically distributed offices communicating through the Internet might install the Trusted Mail Gateway to each of the offices in order to provide domain security among them. Security functionality of the Trusted Mail Gateway is based on standards. Trusted Mail Gateway accomplishes confidentiality, integrity, non-repudiation and authentication tasks by employing Public Key Infrastructure (PKI).

PKI includes secret key cryptography, public key cryptography, digital signatures and X509 public key certificates. Trusted Mail Gateway produces domain digital signatures and performs public key encryption for e-mail messages outgoing other trusted domains. Security services employed in domain security including digital signatures and encryption are based on Secure/Multipurpose Internet Mail Extensions (S/MIME). Signature verification and decryption is performed incoming e-mail messages if necessary. Moreover, incoming messages are written into users' mailbox files as encrypted e-mails. All public key usage including signature verification is based on public key certificates. Both incoming and outgoing e-mails are checked against viruses and worms by anti-virus software component of the Trusted Mail Gateway. Content filtering is applied to incoming and outgoing e-mails' content by using pre-defined text patterns. Security policies and control information are exchanged between the Trusted Mail Gateways over the Internet. The Trusted Mail Gateway holds information about clients' actions on their mailboxes, e-mails, access times and traffic logs in relational database. Trusted Mail Gateway carries out all the functions mentioned in a transparent way. That is, domain users do not need to be aware of the Trusted Mail

Gateway and its functions. Domain users shall only know the Trusted Mail Gateway as their mail server.

The following chapters consist of theoretical and technological information on electronic mail systems and information security, design and architecture of the system, software design and implementation of the system and conclusion.

# CHAPTER II

## EVOLUTION OF THE E-MAIL STANDARDS AND SOFTWARE TECHNOLOGY

In this chapter, Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP) and Internet Message Access Protocol (IMAP) are going to be explained. Also, e-mail message formats are discussed under the title Multipurpose Internet Mail Extensions (MIME). Theoretical background information on cryptography and S/MIME are given in the chapter. Previous studies carried out to establish secure e-mail systems in the literature are written as the last section of this chapter.

## 2.1 MAIL PROTOCOLS

### 2.1.1 SMTP (Simple Mail Transfer Protocol)

The objective of the Simple Mail Transfer Protocol (SMTP) is to transfer e-mail via reliable and efficient way [1]. SMTP operates independently from underlying transmission subsystem and requires reliable ordered stream channel [1]. In TCP/IP model, it works over the Transmission Control Protocol (TCP). A crucial characteristic of SMTP is its capability to transfer e-mail across networks. That is, a process can transmit e-mail to another process on the same network or to another network via a relay or gateway process accessible to both networks.

Having finished giving introductory SMTP information, basic SMTP model shall be explained. When an SMTP client has an e-mail message to send, it shall establish a two-way transmission channel (a TCP connection in case of TCP/IP network) to an SMTP server. The SMTP server might be the final destination or an intermediate relay or gateway. In case of relay destination, the relay assumes the role of SMTP client while transferring e-mail to other SMTP server. The relay does not modify message data other than adding trace information. If an intermediate destination is a gateway,

the gateway might send e-mail further by using protocols other than SMTP. If sending and receiving transport environments are disparate, the gateway performs necessary transformations. Hence, e-mail transfer occurs in a single connection or in a series of connections through intermediate systems. The responsibility of an SMTP client is to transfer e-mail messages to an SMTP server or to report failure. An SMTP server is required to deliver e-mail messages or to report failures while doing so.

Delivering an e-mail message from an SMTP client to an SMTP server is a mail transaction. Mail transaction consists of series of commands and replies. All commands and replies are textual data and transmitted in lines ending with characters “Carriage Return” (CR) and “Line Feed” (LF). Commands explain to whom the mail is going, who is originating the message and what the message content is. The SMTP server responds each command with a reply. Replies indicate that the command has been accepted or additional commands are expected or temporary or permanent error condition exists. Mail transaction is initiated by an SMTP client once a reliable ordered stream channel has been established (TCP connection). The SMTP client issues “HELO” or “EHLO” command in order to initiate mail transaction.

What SMTP transports is a mail object. A mail object consists of an envelope and content. The envelope part of the mail object is sent as a series of SMTP commands, which are “MAIL FROM” and “RCPT TO”. “MAIL FROM” and “RCPT TO” commands indicate originator and recipient of the e-mail message respectively. The content part of the mail object is transported after SMTP “DATA” command. Once the SMTP server responds positive reply to client’s “DATA” command, the SMTP client transmits e-mail header and body, which comprise content part of the mail object, in lines ending with (CR) (LF). The header forms a collection of (field, value) pairs. A line containing only (CR) (LF) “.” (CR) (LF) indicates end of content portion of the mail object. After that, the SMTP server keeps mail object for transmission. The SMTP client might initiate another transaction or issue “QUIT” command.

### 2.1.2 POP (Post Office Protocol)

The aim of the Post Office Protocol (POP) is to provide mail clients with the opportunity by which clients can list, download and delete their mails from corresponding mailbox files [2]. Deployment of mail transfer systems and SMTP server software to all nodes in the Internet is not efficient since most of the workstations in the Internet do not have

sufficient computing resources and network connectivity. However, it is desirable to manage mail on these workstations, which support user agent (UA) to help the mail handling tasks. In order to solve this problem, a more powerful node on the network gives mail-handling service to the weaker workstations. POP is intended to allow workstations to access their mailboxes dynamically.

Similar to SMTP, transactions between a client and a server occur in series of commands and responses. All commands and responses are textual data. A POP server starts the service on TCP port 110. A client establishes a TCP connection with the server host. The POP server sends a greeting. When the client obtain greeting, it initiates authorization state by issuing "USER" and "PASS" commands. After successful authentication, transaction state starts. The client can issue "LIST", "RETR", "DELE", "STAT", "RSET" and "NOOP" commands. "LIST" command returns the number of messages and scans listing of all e-mail messages in mailbox. "DELE" marks the specified message to be removed from the mailbox. "RETR" is employed to obtain specific e-mail message entirely. "NOOP" does not make server take action other than positive response. "RSET" changes the state of the messages from being marked as deleted to

undeleted. The server responds with "+OK", a positive response, or "-ERR", a negative response. Not all responses are single line. All commands, single line responses and lines in multi line responses end with (CR) (LF) pair. Multi line responses stop with a line containing "(CR) (LF) (.) (CR) (LF)" sequence only. Once the client decides to complete transaction, it issues "QUIT" command and transaction enters update state. The server deletes all messages that are marked as deleted by the client and releases all resources it is using. Finally, the server closes TCP connection. If transaction ends with a reason other than client issued "QUIT" command, then transaction does not enter update state and the server do not perform any deletion for marked mails.

### 2.1.3 IMAP (Internet Message Access Protocol)

The Internet Message Access Protocol (IMAP) allows an e-mail client to make access and manipulate e-mail messages on a server. Similar to the Post Office Protocol (POP), IMAP is a method of accessing e-mail messages kept on a mail server. IMAP permits e-mail clients to make access remote mailboxes as if they were local mailboxes [3]. For instance, an e-mail client can manipulate his/her e-mail messages stored on an IMAP server from a workstation at the office, a personal computer at

home and a notebook computer while traveling without transferring the mailbox file among computers. This feature is a difference between IMAP and POP. Another difference is that a client can create, rename and delete a mailbox. Also, IMAP provides a way for an offline client to synchronize with the server. Moreover, selective message retrieval based on message attributes and portions can be made. Thus, IMAP is a more complicated protocol than POP.

IMAP requires a reliable ordered stream channel. In case of TCP/IP, an IMAP server starts the service on TCP port 143 [3]. An IMAP connection is a client/server connection. Once the connection is established, a server sends a greeting. This client/server connection consists of a client command, server data and a server response. Similar to the Simple Mail Transport Protocol and POP, all commands, data and responses are in the form of lines ending with "(CR)(LF)". The client initiates an operation by issuing a command. Each client command is prefixed with an alphanumeric identifier. The alphanumeric identifier is called a tag. For each command, a client generates a unique tag. Status responses and data transmitted by the server to the client are also prefixed with tags. Those that do not indicate command results are prefixed with "*" and called

untagged. Command status responses indicate success or failure of client commands. A status response is tagged with the same tag that is prefixed to the corresponding client command at client side. Thus, the tag in a server response identifies the command to which response applies in case of multiple commands in progress. There are three status indicators, which are “OK”, “NO” and “BAD” indicating success, failure and protocol error respectively.

An IMAP client is ready to accept any server response at all times. This condition includes server data that an IMAP server sends unilaterally. That is, the server might send data as a result of the client command or independent from the client command. There is not any syntactic difference between the server data that is sent in case of command client and the server data that is transferred unilaterally.

There are four states defined for an IMAP server. The states are non-authenticated state, authenticated state, selected state and logout state. Certain commands are only valid in certain states [3]. It is a protocol error for the client to issue a command in an inappropriate state. In non-authenticated state, the client supplies authentication information to the server. In authenticated state, the client has been authenticated and selects

a mailbox before issuing IMAP commands. The selected state is entered once a mailbox has been successfully selected. In logout state, the client/server connection is terminated. The state is entered as a result of either client command or unilateral server decision.

### 2.1.4 MIME (Multipurpose Internet Mail Extensions)

An e-mail message consists of an envelope and content. The content part contains some number of header fields and body. As a standard, RFC822 [14] defines message header fields. Logically, each header field is composed of a single line of ASCII text containing the header field name, a colon, and usually field value. The main header fields related to message transport according to RFC822 are “To:”, “Cc:”, “Bcc:”, “From:”, “Sender”, “Received”, “Return-Path:”. The fields are self-explanatory. “To:” indicates to which recipient the message is being delivered. “From:” and “Sender:” tell who wrote and sent the message. “Cc:” and “Bcc:” fields give the addresses where carbon copies of the message are being sent. Each mail transfer agent along the way adds “Received:” field as a line, which consists of the identity of the agent, the date and time the message was received. The final mail server adds “Return-Path:” field to track a

way back to the sender. There are also informative header fields such as “Subject:”, “Message-Id:”, etc.

RFC822 performs very well when e-mail messages contain only English ASCII text message bodies [9]. Nowadays, requirements for e-mail message bodies are being developed. For instance, there are messages expressed in languages with accent, messages in non-Latin languages, e-mails in languages without alphabets and not containing textual data but multimedia and binary data. There is a solution described in RFC1521, which is Multipurpose Internet Mail Extensions (MIME) [6]. MIME continues to make use of the RFC822 header field format. MIME structures the message body. MIME adds necessary header fields identifying message body structure. The header fields added to RFC822 are “MIME-Version:”, “Content-Description:”, “Content-Id:”, “Content-Transfer-Encoding:” and “Content-Type:”. “MIME-Version:” identifies which version of MIME is used. If an e-mail message header does not contain MIME version information, it is assumed to be English plaintext message. “Content-Transfer-Encoding:” tells how the message body is encoded while transporting it through a network. Example encoding schemes are 7-bit, 8-bit and base64 encoding. Base64 encoding is the

correct scheme for binary message content. Since messages are transferred in lines of strings and base64 consists of uppercase and lowercase English letters, ten digits, “+”, “/”, “==” and ”=”, binary data has to be converted into base64. “Content-Type:” is the field that identifies structure of the message body with type and subtype information. Type and subtype information and encoding scheme permit the mail systems to transfer e-mails constructed in various formats.

## 2.2 Information Security

### 2.2.1 Cryptography

In today’s world, communication through computer networks and the Internet has superseded paper communication such as letters. There are countless numbers of hosts in the Internet. Messages are transported through several routers and links that are also open all Internet traffic. Normally, all data transported is plaintext. The plaintext data transported over open links results in disclosure of the transmitted data. A malicious user on a workstation might eavesdrop links and obtain the content of all the traffic listened. Thus, confidentiality is a crucial requirement. Cryptography meets that requirement. Cryptography provides techniques

transforming data in ways that are hard to mimic or reverse by someone who is not part of secret transmission. One of the parties in conversation takes plaintext and encrypts it. The result of the encryption is ciphertext, which is scrambled version of the plaintext data. The receiving party in the conversation applies the reverse process to ciphertext data in order to obtain original plaintext data. This process is called decryption.

A modern cryptographic technique has a number of necessary elements that determine how the technique works. The first element is the cryptographic algorithm [8]. The cryptographic algorithm is the specification of the mathematical transformation that is performed on data to encrypt or decrypt it [8]. Also, a cryptographic algorithm is a procedure that takes plaintext data and transforms it into ciphertext in a reversible way. Quality of a cryptographic algorithm comes from how much clue the ciphertext data yields about either the key or plaintext data that has produced it. This quality is the strength of a cryptographic algorithm.

Cryptographic algorithms are classified into two broad classes, which are secret key algorithms and public key algorithms. A secret key algorithm is symmetric. That is, it uses the same key in both encryption and decryption. The security of a secret key relies on keeping the key itself

completely secret from outsiders. On the other hand, public key algorithms use different keys for encryption and decryption. One key, private key, has to be kept secret and should not be shared with anyone else. The other key, public key, can be shared with anyone. Public and private keys are mathematically related. Data encrypted with the public key is decrypted with the corresponding private key and vice versa.

#### 2.2.1.1 Secret Key Cryptography

The design motive of the secret key algorithms relies on usage of substitution and transposition [9]. In substitution, each portion of plaintext is replaced by some other data known in advance. Each part of the plaintext is scrambled. In contrast to substitution, transposition reorder parts of plaintext but it does not disguise portions of the plaintext. Substitutions are implemented by using data structure "S-boxes". Transpositions are based on "P-boxes", where "P" stands for permutation. When a series of those boxes are cascaded, the strength of them becomes apparent since they form a product cipher.

Secret key techniques are classified as stream ciphers and block ciphers. Stream ciphers are designed to take a key and a stream of plaintext as

arguments to produce a stream of ciphertext as output. Today, RC4 (Rivest Cipher #4) and SEAL (Software Optimized Encryption Algorithm) are well known stream ciphers. Block ciphers, on the other hand, take a data block of a particular size and a key in order to encrypt data block with the key to produce a block of ciphertext. Contemporary block ciphers produce a ciphertext block that is the same size as the plaintext block. Data Encryption Standard (DES) [27], Triple DES [27], and Advanced Encryption Standard (AES) [28] are the examples of well-known block ciphers. There are four basic modes used with block ciphers. The modes are electronic code book, cipher block chaining, cipher feedback and output feedback. Electronic code book is the trivial application of a block cipher to plaintext. In cipher block chaining, each plaintext block is combined with a ciphertext block from previously encrypted plaintext before actually encrypting the plaintext block. Cipher feedback mode feeds the ciphertext block back through the block cipher to change the encryption key constantly. Output feedback mode is similar to cipher feedback but ciphertext is not fed into block cipher. Deciding which block cipher mode should be selected depends on in which application the block cipher is being used.

### 2.2.1.2 Public Key Cryptography

There are two widely used technologies in the class of public key cryptography. One is Diffie-Hellman technique and the other is RSA. Diffie-Hellman technique is the first practical public key algorithm. Two entities in a communication can use the technique to generate a shared secret value as a common key for a secret key encryption. Each party in the communication produces a random value and this becomes a private key. Let the two entities be A and B. If two entities need to exchange data, they exchange some public data that comprises the public key. Entity A applies its private key to B's public key and B applies its private key to A's public key. Both parties reach the identical value. The strength of Diffie-Hellman technique is based on difficulty of computing discrete logarithms modulo a large prime number.

RSA is a public key algorithm invented by Rivest, Shamir and Adleman. It involves exponentiation modulo of two randomly generated large prime numbers. The letters "e", "d", "n", "p", "q" and "ϕ" indicate public exponent, private exponent, modulus, a large prime number, another large prime number and multiplication of one-decremented large primes

respectively. Public key consists of (n,e) pair and private key consists of (n,d) pair. Public-private key pair is generated as follows [10]:

1. Two large primes, p and q, are randomly generated with equal length in bits,

2. Public exponent "e" is produced, which is usually either 0x03 or 0x010001 hexadecimal,

3. n = p.q is computed,

4. $\phi = (p - 1).(q - 1)$ is computed,

5. Private exponent "d" is calculated from equation $e.d \equiv 1 \pmod{\phi}$.

In RSA, encryption is performed by the following equation: Let "a" be the plaintext data and "c" is the ciphertext data. Assume that encryption is performed with public key. Encryption equation is $c = a^e \bmod n$. Decryption equation is similar to encryption but it is carried out with private key: $a = c^d \bmod n$. As $e.d \equiv 1 \pmod{\phi}$, exponentiation operation with matching private key (in case encryption with public key) yields plaintext. Usually, RSA encryption is employed while encrypting a

symmetric encryption key. The strength of RSA is caused by the difficulty of factoring integers that are product of two equal sized large primes.

### 2.2.2 Message Digest Algorithms

RSA encryption process is a CPU bound process. Also, there is a limitation for data size in case of RSA encryption. Thus, it is usually required to compute condensed representation of data while performing RSA encryption. The idea of a one-way hash function meets these requirements. A one-way hash function takes a long piece of plaintext data and computes a fixed-length bit string from that [9]. A one-way hash function is also called a message digest.

A message digest has three important properties. Firstly, it is easy to compute digest of plaintext data. Second crucial point is that given digest, it should be computationally infeasible to retrieve plaintext from which the digest has been computed. Lastly, a message digest should not generate the same digest from two different plaintext messages. In order to meet the last criteria, the output length of the hash function should be at least 128 bits long.

There are a variety of one-way hash functions proposed. The most widely used functions are MD5 [30] and SHA [31]. SHA produces 160-bit output and MD5 produces 128-bit digest. Since output length of SHA is 32-bit longer than MD5, SHA is more secure than MD5 with a factor of $2^{32}$ if all other conditions are equal. However, SHA is slower than MD5.

### 2.2.3 Digital Signatures

The origin authentication and integrity of several financial, legal, official and other kind of documents are determined by the presence of authorized handwritten signatures. In the domain of computers and information systems, there is a requirement for a solution that supersedes the physical transport of paper and printed documents. There are three important requirements for signed data transported through a computer network. Firstly, the receiver can verify the identity of the sender. This indicates the data origin authentication. Then, the receiving party can make sure that the message just received has not been altered. This points to the integrity. Lastly, the sending party cannot repudiate the contents of the message and that (s) he has sent the message. The last property is non-repudiation.

The technology that the digital signature is based on utilizes RSA encryption with message digest functions. Let "M" be the data to be sent, "E" be RSA encryption routine, "D" be RSA decryption routine, "MD" be message digest, (e,n) be public key and (d,n) be private key. Signature is generated by the message sending side as follows [11]:

1. digest = MD(M), a fixed-length message digest is computed,

2. signature=E(digest,(d,n)), digest is encrypted with the private key of the sending party by using RSA encryption,

3. M and signature are both delivered to a receiving entity.

At the receiving side, signature verification is carried out as follows:

1. digest=D(signature,(e,n)), signature is decrypted with the public key of the sending entity by using RSA decryption routine,

2. digest′=MD(M), a fixed-length message digest is computed,

3. The receiving party checks whether digest and digest′ are equal. If they are the same, then signature is verified. Otherwise, message M is altered on the way or the message is not from the actual sending party.

### 2.2.4 Digital Envelopes

Digital enveloping is a mix of secret key and public key cryptography. Usually, secret key techniques are employed in encryption of large volumes of data and public key techniques, RSA, are used to encrypt content encryption keys used in secret key algorithms. If two parties desire a secure communication and the originator of the communication possesses the public key of its peer, then the originator party can use digital envelope scheme to deliver its message as encrypted. The following items reflect the production and interpretation of a digital envelope. Assume that M is a message to be sent, C is a secret key content encryption algorithm and (e,n) and (d,n) are public and private keys of the recipient respectively. Also, E and D are RSA encryption and decryption operations respectively.

1. The originator randomly generates a symmetric encryption key, k.

2. The originator encrypts M with C by using key k: M′ = C(M,k).

3. The originator applies RSA encryption to k with the public key of the recipient: k′ = E(k,(e,n)).

4. The originator sends M′ and k′ to the recipient.

5. The recipient applies RSA decryption routine to k′ by using its private key: k=D(k′,(d,n)).

6. The recipient applies secret key decryption to the M′ by using k: M=C(M′,k).

The secret key algorithm provides the confidentiality of the message and the privacy of the content encryption key is supplied by the RSA encryption. Since only the owner of the private key can decrypt the content encrypted with the public key, the security of the content encryption key is relied on the strength of RSA. The mail agents utilize digital envelopes to deliver e-mail messages encrypted.

### 2.2.5 Public Key Certificates

Users of a public key need to make sure that the correct system or person owns the corresponding private key. In other words, there is a confidence requirement about ownership of the private key. A public key certificate provides the confidence. A public key certificate is a data structure that binds a public key value to a subject [12]. A certificate is digitally signed data. The trusted certificate issuer, called certification authority (CA), digitally signs public key, name and other data of subject to produce a certificate. A certificate has a limited validity period indicated in its signed content. Certificates can be distributed via untrusted communication links and can be stored in unsecured caches since the certificate user independently carries out signature verification and validity period checking on a certificate. CA also owns a certificate that is either issued by another CA or self signed. A self-signed certificate is the certificate that is signed with its own private key. A self-signed certificate needs to be self-verified.

A certificate user requiring a public key has to obtain and validate a certificate that contains required public key. A certificate user also requires the CA certificate signed that certificate in order to verify the

signature on it. Signature verification might be applied to the CA certificate with a certificate that is owned by another CA. Thus, there is a trust hierarchy. The trust hierarchy is called a certification path.

A certificate consists of three main parts, which are "TBSCertificate", "SignatureAlgorithm" and "Signature". "TBSCertificate" portion of the certificate contains version information, subject name, issuer (CA) name, and public key of the certificate owner, validity period and other optional fields. "SignatureAlgorithm" indicates which message digest function and public key encryption algorithm are used to produce the signature. "Signature" field is the signature value computed from "TBSCertificate" data by applying a message digest function and public key encryption algorithm specified in "SignatureAlgorithm" field.

## 2.3 Secure Multipurpose Internet Mail Extensions (S/MIME)

In order to send and receive secure Multipurpose Internet Mail Extensions (MIME) data, there should be a consistent way. Based on the Internet MIME standard, S/MIME provides the required consistency with the cryptographic services, which are authentication, message integrity, non-repudiation and data confidentiality [7]. Using digital signatures provides

the first three of the cryptographic services and using encryption provides the last one. Both traditional mail user agents and message transfer agents can utilize S/MIME in a transparent way. Mail agents apply cryptographic security services to the mail that is delivered and interpret security services in mail that is received.

S/MIME is used to secure MIME entities [16]. A MIME entity consists of only the MIME headers and body, not RFC822 headers. If protection of RFC822 headers is required, the whole message is enveloped into a new MIME message whose content type is *message/rfc822* [16]. Thus, outer and non-content related message headers are protected. Prior to the message sending with S/MIME, a mail agent prepares a MIME message with suitable encoding scheme. Then, the agent applies cryptographic services according to S/MIME. What the receiving mail agent carries out is the reverse process of sending S/MIME message. The receiving agent firstly processes the cryptographic services applied to the message according to S/MIME. What the receiving agent obtains after S/MIME processing is the MIME message. If the receiving agent interprets that the top-level protected MIME message has a content type *message/rfc822*, it can be assumed that the intent was to provide RFC822 header protection.

S/MIME provides one data format for encrypted messages (enveloped data) and more than one data formats for signed messages. Since S/MIME is based on MIME, there are content types identifying S/MIME content. The *application/pkcs7-mime* content type is used carry S/MIME processed data. It is used to carry encrypted, signed and compressed messages. There are several types of *application/pkcs7-mime* objects. In order to simplify the work done by the receiving mail agent to decode S/MIME content, the sending agents use *smime-type, name* and *filename* parameters to identify the type of the S/MIME object [16]. "*name*" and *"filename"* parameters include file extensions identifying the kind of S/MIME object. *"application/pkcs7-mime"* content type with *name* and *filename* parameters *".p7m", ".p7c", ".p7z"* reflects encrypted messages (enveloped data), signed data, *"certs-only"* signed data and compressed data respectively. Also, the *"application/pkcs7-signature"* content type with *name* and *filename* parameter *".p7s"* reflects signed data. Moreover, *smime-type* parameters *"enveloped-data", "signed-data", "certs-only", "compressed-data"* gives identification for the S/MIME objects. All the S/MIME objects are transported in special data structures defined in Cryptographic Message Syntax (CMS) and the following subsection deals with CMS.

## 2.4 Cryptographic Message Syntax (CMS)

The Cryptographic Message Syntax (CMS) describes encapsulation syntax for data protection. It supports digital signatures, encryption and compression [17]. The CMS syntax is used to generate digital signatures, message digests, to provide authentication and to encrypt messages. Also, the syntax supports nesting of cryptographic operations [17].

CMS carries S/MIME objects in data structures represented as Abstract Syntax Notation (ASN.1) [26] data structures. The CMS is derived from Public Key Cryptographic Standards 7 (PKCS7) [17]. The CMS is generic enough to support diverse content types. However, CMS standard defines only one protection content that is *ContentInfo* [17]. What *ContentInfo* encapsulates are content type information and a content identified by the content type that can further encapsulate other contents. The CMS defines six content types, which are *data, signed-data, enveloped-data, digested-data, encrypted-data* and *authenticated-data*.

*Data* content type refers to arbitrary sequence of bytes, such as ASCII text files. Such byte sequences do not have to contain internal structure. S/MIME uses *data* content type to identify MIME content. The *data* content

type is generally encapsulated in the *signed-data, enveloped-data, digested-data, encrypted-data* and *authenticated-data*.

The *signed-data* content type consists of a content of any type and zero or more signature values. The common application of the *signed-data* content type is to represent signature of one signer on content of the *data* content type. In S/MIME, the *signed-data* content type is used to carry *application/pkcs7-mime* content type with *smime-type signed-data*, *application/pkcs7-signature* content type *signed-data* and *application/pkcs7-mime* content type with *smime-type certs-only* content. In the first case, the data to be signed is encapsulated in the *signed-data* with the signature generated on the data. In the second one, only the signature generated on the data is encapsulated in the *signed-data*, not the data that is signed. In the *certs-only* case, neither the data nor the signature is included in the *signed-data*. Only the X509 certificates that will be delivered are encapsulated in the *signed-data* content type.

The *enveloped-data* content type encapsulates encrypted content of any content type and encrypted content encryption keys. The combination of encrypted content and the content encryption key of that encrypted content is a digital envelope. Digital enveloping is employed to produce

*enveloped*-data. S/MIME uses *enveloped-data* content type to carry *application/pkcs7-mime* content type with *smime-type enveloped-data*. The *enveloped-data* provides means for exchanging encrypted messages among mail users.

## 2.5 Mail Security Schemes in the Literature

When a mail message is delivered between two sites, it usually transits dozens of intermediate hosts on the way. Any of these hosts can access, record and even modify the message being transported. Also, e-mail messages are transported over the Internet through the same transport media with other application protocol data. Many people who are not intended recipients would have the opportunity to intercept and alter messages. Thus, privacy (confidentiality) and integrity of e-mail messaged are required. There have been many studies to accomplish mail security issues. Here, four of those studies are going to be discussed. Two of them are older studies and two of them are most recent works. Older ones are Pretty Good Privacy (PGP) [9] and Privacy Enhanced Mail (PEM) [9] and the recent ones are SenderID Framework [24] and DomainKeys (Yahoo) [23].

### 2.5.1 Pretty Good Privacy (PGP)

Pretty Good Privacy (PGP) is the invention of Phil Zimmermann [9]. PGP is an e-mail security package that provides confidentiality, authentication, message integrity and non-repudiation. PGP utilizes existing cryptographic algorithms and schemes such as RSA as a public key technique, IDEA as a symmetric encryption routine and MD5 [30] as a message digest algorithm since all these methods and algorithms have undergone extensive public reviews [9].

Having completed background information about PGP, how PGP carries out security functions can be dealt with. Assume that there are two peers *A* and *B* where A wants to send its message to B in a secure way. Let *P* be original plaintext message, $D_x$ and $E_x$ be RSA private and public keys respectively. Also assume that both *A* and *B* have each other's public key. The following steps are performed prior to the message delivery:

1. The message digest of *P* is computed by using MD5.

2. The digest is encrypted with RSA encryption with $D_A$. The sum of step (1) and step (2) is digital signature computed on *P*.

3. *P* and digital signature of *P* are concatenated.

4. The concatenation from step (3) is compressed.

5. Party *A* generates a random 128-bit IDEA content encryption key *K*.

6. The compression result from step (4) is encrypted by using IDEA with key *K* from step (5).

7. Using RSA encryption with public key of B, $E_B$, *A* encrypts key *K*.

8. Results of step (6) and step (7) are concatenated.

9. The concatenation from step (8) is converted into base64 format.

The recipient performs reverse operations on the message with reverse order. In other words, *B*, the recipient, performs RSA decryption with $D_B$, IDEA decryption, and then performs signature verification with $E_A$.

The key management is accomplished by maintaining two separate data structures, which are public key ring and private key ring [9]. The public key ring consists of public keys of the e-mail user's correspondents. The

private key ring contains one or more personal private keys to permit e-mail user to change his/her private key periodically.

### 2.5.2 Privacy Enhanced Mail (PEM)

In contrast to PGP, Privacy Enhanced Mail (PEM) is an official Internet standard that is described in RFC1421, RFC1422, RFC1423 and RFC1424 [9]. What PEM provides are the similar security properties, which are confidentiality and authentication. PEM uses MD2 or MD5 to compute message digests and uses DES [27] secret key algorithm to encrypt messages [9]. Either triple DES or RSA protects the DES key.

Assume that *A* wants to send a secure mail message to *B*. The following are steps to produce PEM message and reverse of them are performed by the recipient to obtain the original message and to check its integrity:

1. The message to be sent is converted into a canonical form.

2. The message digest is computed on the message by using either MD2 or MD5.

3. The digest and the canonical form message are concatenated.

4. Party *A* generates a random 56-bit DES key *K*.

5. The result of step (3) is encrypted using DES with *K*.

6. Key *K* is encrypted by RSA.

7. The results of step (5) and step (6) are converted into base64 format.

Key management in PEM is achieved by employing X509 public key certificates [9]. Certification Authorities (CA) certifies keys by binding them to key owner's name with expiration time. Certification is verified via verification of the CA signature on a user certificate.

### 2.5.3 DomainKeys

DomainKeys creates a domain-level authentication framework for email by using public-key technology and the DNS to prove the provenance and contents of an email [23]. E-mails are digitally signed per domain basis. Ultimate goal is to prove and protect identity of mail sender. This assists in the control of spam and spoofing.

DomainKeys differs from the traditional hierarchical public-key systems in such a way that it leverages the DNS for public-key management with direct and complete control of the domain owner on production and management of domain keys [23]. The system does not require the use of certificate authority.

A domain owner generates one or more public/private key pairs that will be used to sign mails originated from that domain. The domain owner places the public key in a DNS record associated with that domain and makes the private key available to the domain mail server (outbound mail server). When e-mail is retrieved from an authorized user of the domain, domain mail server uses the private key to sign the message. The signature is added as a header to the mail. The e-mail is transferred to recipients in the usual way. When the receiving system obtains e-mail with domain signature, it shall verify the signature as follows:

1. Extract signature and claimed sender information from the e-mail,

2. Retrieve the public key from the originator's DNS record,

3. Verify the signature with that public key by using public key cryptography to prove whether the authorized user of the claimed sender domain sends e-mail.

### 2.5.4 Sender ID Framework

The SenderID Framework is an industry standard that is created against e-mail domain spoofing and to provide stronger protection against phishing schemes. The framework is a combination of Microsoft's Caller ID, Men Wong's Sender Policy Framework (SPF) [25] and a third specification from the Submitter Optimization.

Sender Policy Framework (SPF) defines a sender authentication scheme based on designated sender model [25]. In this model, a domain identifies certain hosts as designated senders. Messages from those hosts are considered to be legitimate. Messages from other hosts are not accepted as legitimate. SPF publishes the designated senders in the DNS.

E-mail verification process is as follows:

1. The sender domain sends e-mail to the receiver domain,

2. The receiver party obtains mail,

3. The Receiver's server checks for the SPF record of the sending domain published in the Domain Name System (DNS) record,

4. The inbound e-mail server determines if the sending e-mail server's IP address matches the IP address that is published in the DNS record.

### 2.5.5 Comparison of the Mail Security Schemes in the Literature

The mail security schemes in the literature could be classified into two classes, which are mail user agent oriented solutions and domain level solutions. PGP and PEM are mail user agent oriented solutions. Clients that want to perform secure mail transaction shall deploy either PGP or PEM separately. On the other hand, Microsoft Sender ID and Yahoo DomainKeys offer domain-level solutions. If secure e-mail transaction is required, Microsoft Sender ID or Yahoo DomainKeys solution is launched at a single controlled point in the domain and the solution provides security on behalf of the domain.

What PGP and PEM have in common is that both employ existing cryptographic techniques. However, PGP is more secure than PEM. Reasons for this is as follows:

- PGP appends signature to the message whereas PEM only attaches digest of the message. PGP provides better message content integrity and authentication.

- PGP uses 128-bit content encryption key but PEM employs 56-bit content-encryption key.

- PGP applies compression on concatenated message and its signature in order to perform encryption more efficiently and to decrease the possibility of discovering plain text from cipher-text.

There are also advantageous points of PEM over PGP. These points are as follows:

- PEM is an official standard.

- Usage of trusted-third party certification authorities (CA) in certificate and signature verification in PEM is more secured than using public key rings in PGP.

Yahoo DomainKeys and Microsoft Sender ID are both domain-based solutions. Yahoo DomainKeys employs digital signatures and thus it guarantees message integrity and origin authentication. Microsoft Sender ID does not use any cryptographic operation. Although they differ in the usage of cryptographic techniques, both of them rely on DNS. Yahoo DomainKeys keeps public keys that are used in domain signature verification in DNS. Microsoft Sender ID uses DNS to publish IP addresses of the hosts that are legitimate to deliver e-mails. However, DNS usage makes Yahoo DomainKeys and Microsoft Sender ID less secure since DNS servers and records that have been relied on are vulnerable to cache poisoning and address spoofing attacks. An attacker could easily alter the records containing public-keys in Yahoo DomainKeys and legitimate host IP addresses in Microsoft Sender ID and the security solution would become useless.

# CHAPTER III

# DESIGN OF TRUSTED MAIL GATEWAY

In this chapter, design motives and architecture of the system will be explained.

## 3.1 System Objectives

In this study, main objective is to develop a mail gateway that is capable of providing e-mail security services for a domain where it is deployed. Such a gateway provides domain-based solution to mail security problems. Security services involving confidentiality, authentication, non-repudiation and content integrity is provided by employing PKI methods especially S/MIME [7]. Besides security services, domain is supplied content-filtering, anti-virus control and protection and spam check. Also, trust relations with other trusted mail gateways are established and

control information is exchanged among them via secure protocol. E-mail traffic and user action logging is another property of the system. The trusted mail gateway shall interoperate with existing mail client software and message transport agents.

### 3.2 System Architecture

After its installation to a domain, which already has a mail server and a number of mail clients, the Trusted Mail Gateway is architecturally between the mail clients and the server. The clients connect to the gateway, instead of the real mail server of the domain, for sending and receiving mails. The gateway then handles the necessary communication with the mail server, and makes the necessary checks and changes to the mails on their way. The gateway is also between the outer network and the domain mail server for the incoming mails; that is, the mails coming from other domains are first received by the gateway and then passed to the mail server. This architecture is depicted in Figure 1.

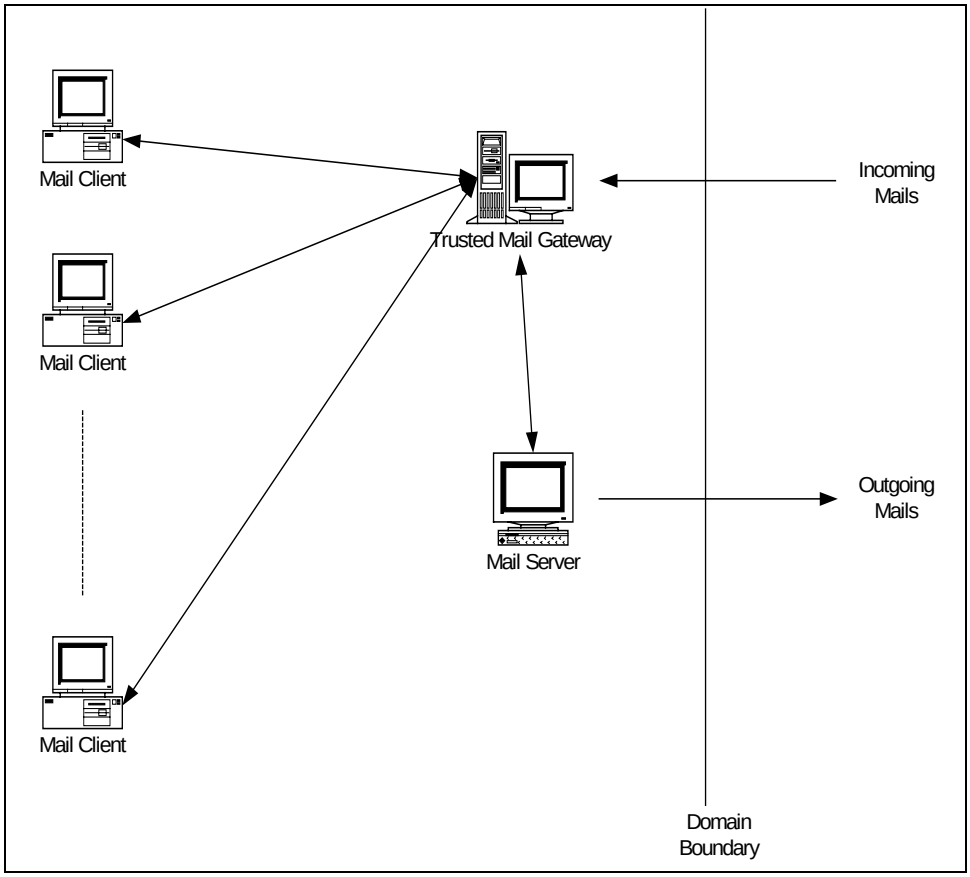


Figure 1: A domain with the Trusted Mail Gateway installed

The system possesses the following functionality:

- **Domain security using S/MIME:** The gateway can generate S/MIME signatures and perform S/MIME encryption on behalf of the domain, using X.509 certificates issued for the gateway and other domains' gateways.

- **Traffic logging and notary mechanism:** The gateway logs all the mail traffic of the domain. The local administrator can use the logs to inform sender domain's gateway about the delivery status of a mail, as well as to answer queries.

- **Anti-virus check:** All mails passing from the gateway are checked for viruses. Virus alerts are logged, and infected mails are stopped.

- **Anti-spam check:** The gateway does not let mails with spam characteristics to pass. Such mails are logged and blocked.

- **Content filtering:** The gateway can be configured to block mails that contain pre-defined text patterns or attachments with specific file types.

- **Communication with other gateways:** Security policies and control information are exchanged among the gateways via secure protocol.

- **Local CA (Certificate Authority):** The gateway employs local CA module in order to issue X509 certificates to clients and itself. Those certificates are only valid in the domain. They are used to produce

signatures, signature verification and encryption purposes in the domain.

- **User Account Management:** The gateway relates X509 certificates issued by local CA with client mail accounts. The gateway notifies local CA to issue a new certificate for a new client and informs local CA to revoke a client certificate if the client account is disabled.

- **Intra-domain S/MIME:** As an advanced feature, in domains with existing PKI systems, the gateway may use clients' X.509 certificates to verify personal signatures on outgoing mails, and encrypt incoming mails. This feature is employed to provide end-to-end security.

Trusted Mail Gateway handles security of an e-mail in such a way that end-to-end security is composed of three pieces, which are security of the e-mail between originator client and the originator domain's Trusted Mail Gateway, security of the e-mail between originator and destination domains' Trusted Mail Gateways and the security of the e-mail between the destination domain's Trusted Mail Gateway and the recipient of the e-mail. The following figure depicts the end-to-end security property of

Trusted Mail Gateway that has three portions. Thick lines represent the e-mail traffic secured by Trusted Mail Gateway.

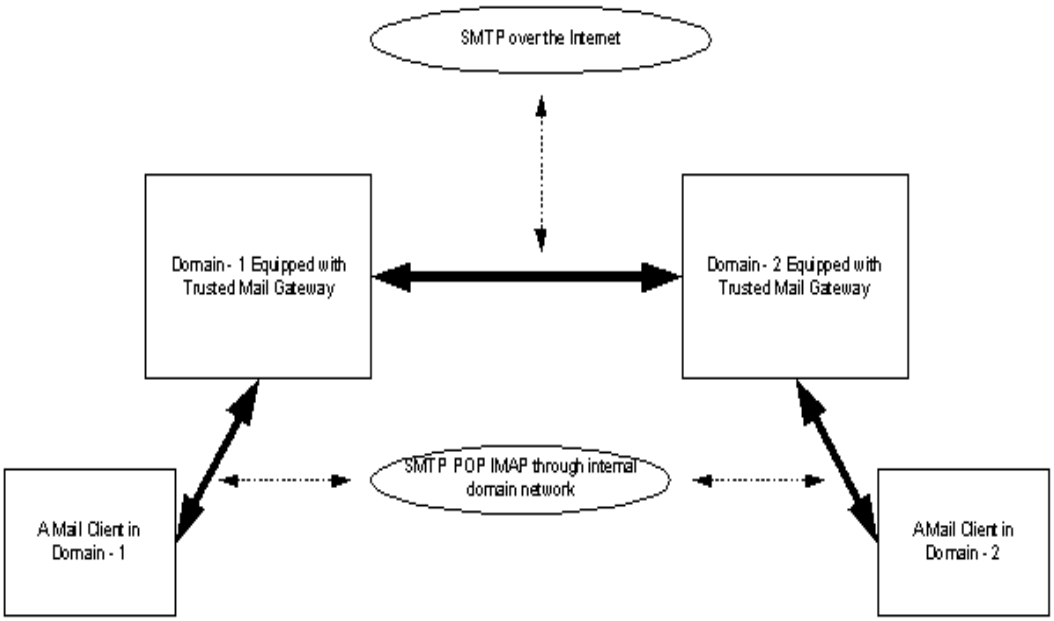


Figure 2: End-to-end security

The software pieces (modules) that form the gateway are shown in Figure 3. Besides the components for the main functions described above, the gateway also provides a web interface to the administrator, and has the ability to warn local or non-local administrators using SMS. The "Trust and Control Engine" controls the operation of the other engines and the interfaces.

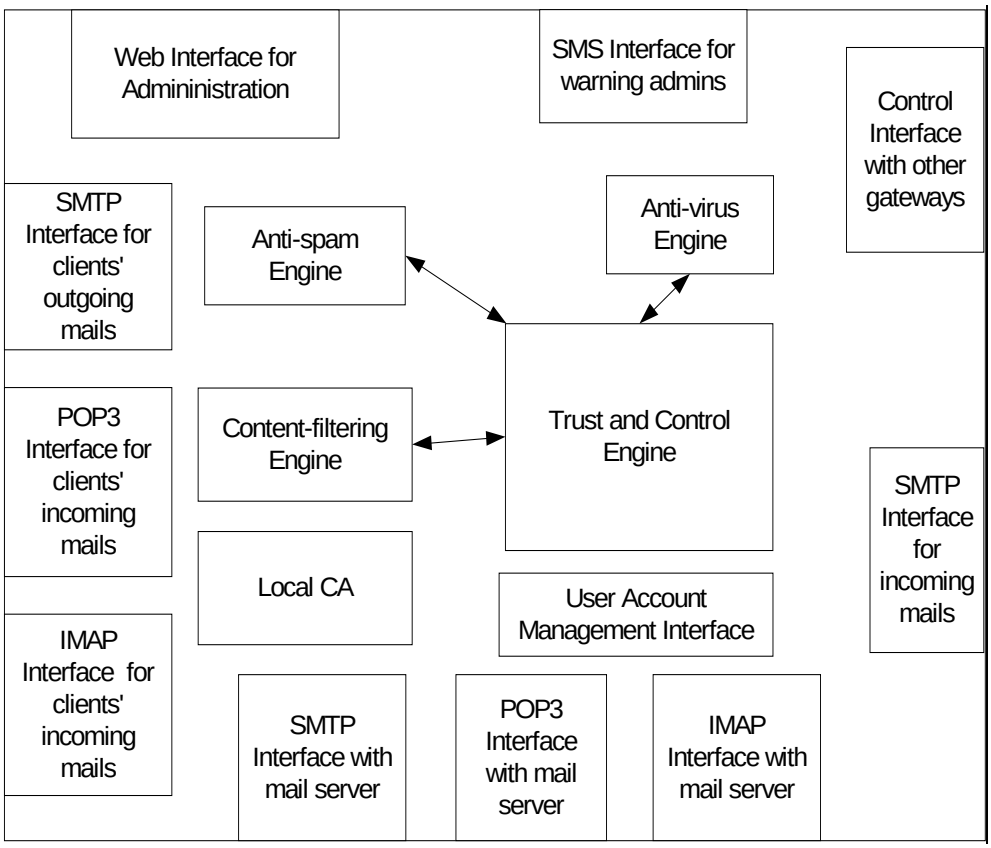


Figure 3: Modules of the Trusted Mail Gateway

## 3.3 Operation Scenarios of the Trusted Mail Gateway

### 3.3.1 Processing of Incoming Mails

a) The incoming mail is received by the SMTP interface of the gateway and it is given to the "Trust and Control Engine".

b) "Trust and Control Engine" checks the sender domain. If the sender domain is also using a Trusted Mail Gateway, there may be S/MIME

processing at this step, depending on the gateways' configuration (verification of the domain signature, domain-decryption, or both). Verification or decryption failures at this step cause the mail to be logged as invalid and the processing finishes.

c) If the mail is valid, "Trust and Control Engine" gives the mail to the "Anti-virus Engine". Based on the result of the anti-virus check, "Trust and Control Engine" either logs the mail as infected or continues with the next step (d).

d) "Trust and Control Engine" gives the mail to the "Anti-spam Engine". Based on the result of the anti-spam check, "Trust and Control Engine" either logs the mail as spam or continues with the next step (e).

e) "Trust and Control Engine" gives the mail to the "Content-filtering Engine". If the mail has some content that is not allowed to enter the domain, it is logged as disallowed; otherwise the processing continues with the next step (f).

f) The mail is sent to the mail server of the domain via SMTP, and if that communication is successful, the mail is logged as a "successful delivery".

The software blocks related with these steps are shown in Figure 4.

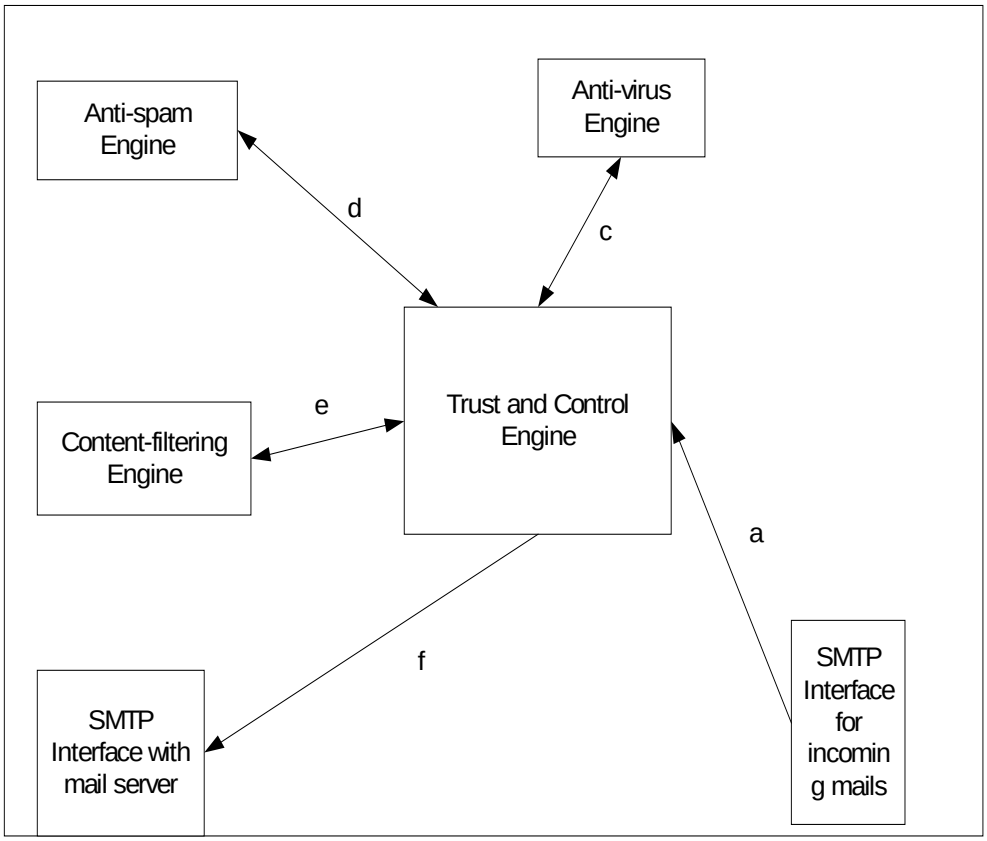


Figure 4: Software blocks that process an incoming mail

### 3.3.2 Client Access to Mails

a) The client connects to the POP3 or IMAP interface of the gateway.

b) The gateway connects to the respective port of the mail server of the domain.

c) Since the gateway stays transparent in this protocol communication, the clients get their mails as if they were directly connected to the mail server. The gateway just logs information regarding to which mails does the client machine retrieve.

The modules related with these steps are shown in Figure 5.

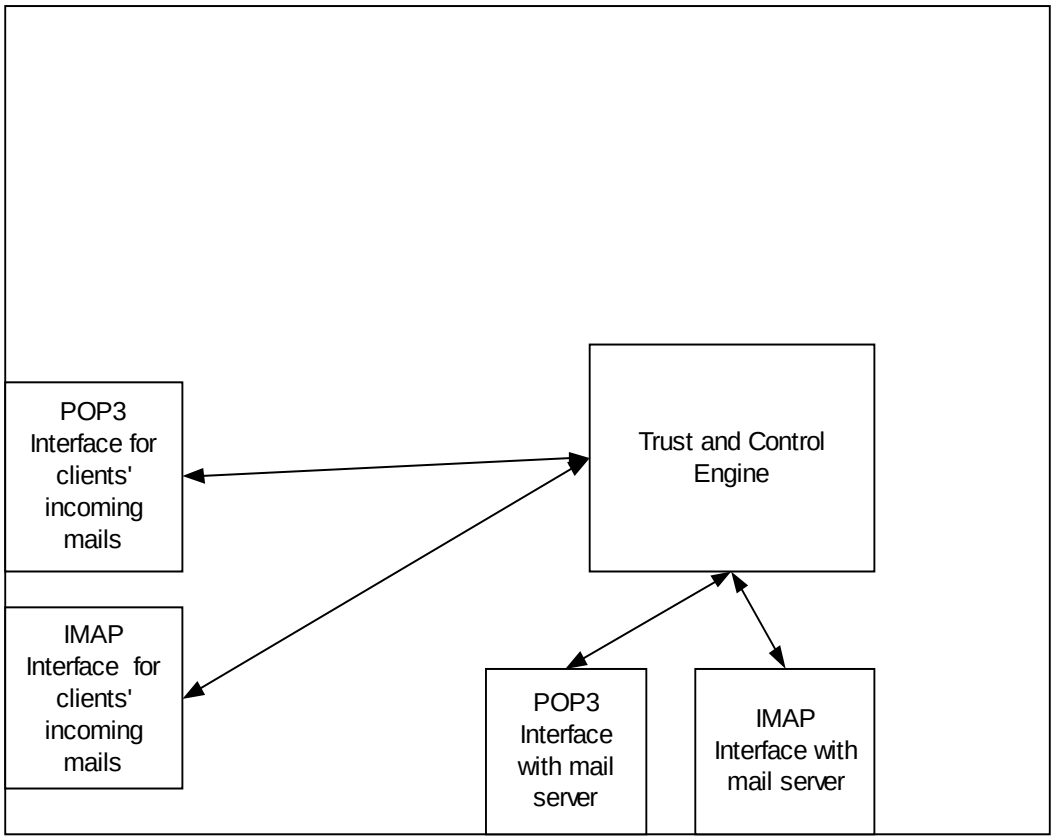


Figure 5: Software blocks that help clients retrieving their mails

### 3.3.3 Processing of Outgoing Mails

a) The client connects to the SMTP interface of the gateway.

b) "Trust and Control Engine" gets the clients mail, and hands it to the "Anti-virus Engine". If the mail is virus-free, the processing continues with the next step (c).

c) “Trust and Control Engine” gives the mail to the “Anti-spam Engine”. The mail is checked for spam properties. If there is no spam detection, the processing continues with the next step (d).

d) “Trust and Control Engine” gives the mail to the “Content-filtering Engine”. The mail is checked for content that is not allowed to leave the domain. If there is no such content, the processing continues with the next step (e).

e) If the recipient domain is also using a Trusted Mail Gateway, there may be S/MIME processing at this step, depending on the gateways’ configuration (generation of a domain signature, domain-encryption for the recipient domain, or both).

f) The mail is handed to the mail server of the domain, from where it will be sent to the recipient. The mail is logged as “sent”.

The software blocks related with these steps are shown in Figure 6.

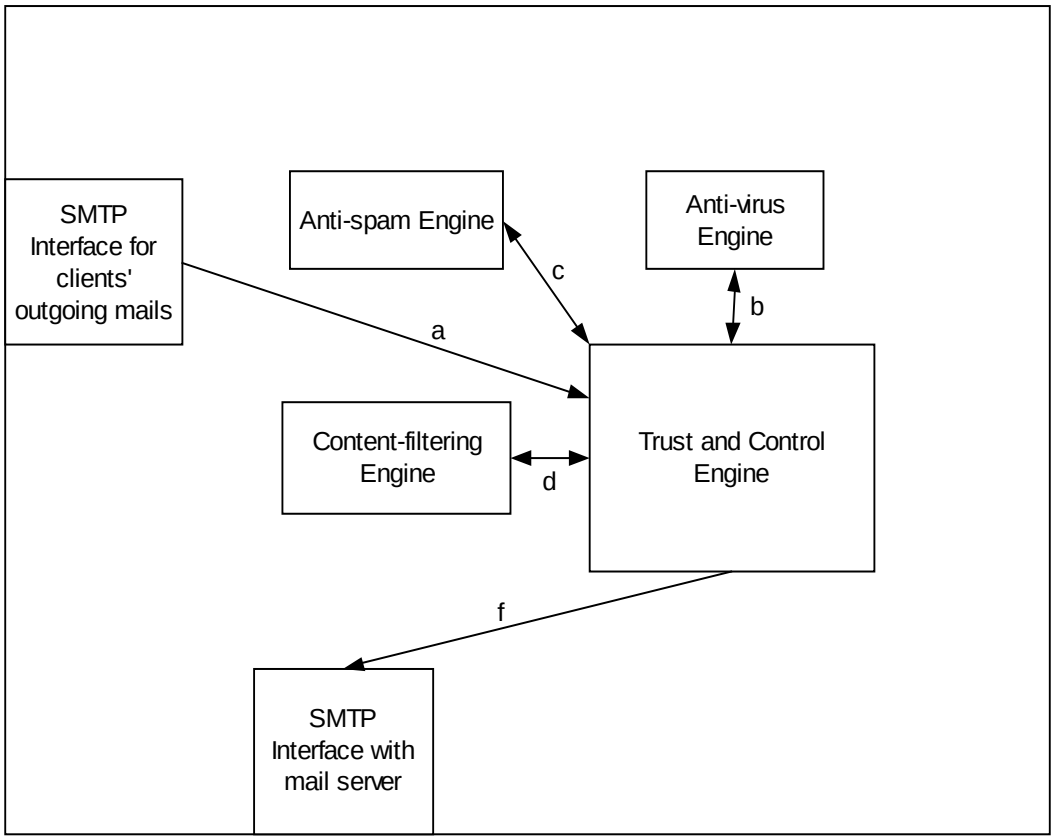


Figure 6: Software blocks that process an outgoing mail

## 3.4 Security Functionalities of the Trusted Mail Gateway

### 3.4.1 Security Mechanism Between a Client and the Trusted Mail Gateway

The following gives information on how the security between domain clients and the trusted mail gateway is provided. It may be concluded that transparency is given up but all the user has to do is to install the X509

certificate of the trusted mail gateway and its PKCS12 [13] on mail client software. In this way, domain users are just forced to encrypt their messages with trusted mail gateway certificate when they are sending e-mails if privacy on the line between client and the gateway is required. End-to-end security can be established if both domains in mail exchange use trusted mail gateway. That is, end-to-end security consists of three items, which are security between an originator client and the trusted mail gateway, security between mail gateways and the security between destination mail gateway and the final recipient. If only one of the parties uses the trusted mail gateway, security is built until the message leaves the domain. The mail clients should be capable of performing PKI operations such as public key encryption, symmetric encryption and digital signatures. MS Outlook, MS Outlook Express, Netscape Messenger and Mozilla are such mail clients. Also, those are the mostly used mail client software products.

There are two alternative approaches to issue X509 certificates for domain clients and relating client mail accounts to those certificates. If the trusted mail gateway is to be deployed in a small or medium sized organization, certification is done manually. That is, there is an administrator interface

for user account management. The administrator adds user account information in the database of the trusted mail gateway and the interface notifies local CA for certificate creation. Each certificate is kept in the database with user account information and client's certificate and the private key is packed together in a PKCS12 file. Clients obtain their PKCS12 files in ternary storage devices (floppy disks, optical disks, compact disks, etc). If the trusted mail gateway is installed in a large organization, then user account management will be run automatically. For each client connected to the trusted mail gateway to send (SMTP interface) or receive mail (POP3 or IMAP interface), the trusted mail gateway behaves transparently and uses the authentication information to authenticate the corresponding service on the domain mail server. If the trusted mail gateway successfully authenticates the corresponding service on the domain mail server, it makes user account management interface query the user account name in the database. If no record exists for a given account information, then user account management interface employs local CA to issue an X509 certificate and a matching private key for a user in PKCS12 format. New certificate is recorded in the database with the user account information and the trusted mail gateway sends its X509 certificate and client's PKCS12 file to the client as a mail attachment.

If the trusted mail gateway could not log on the corresponding service on the domain mail server, the trusted mail gateway closes the connection for the service that the client has connected and logs the client authentication information and IP address as invalid connection information.

The clients that desire end-to-end security firstly provide privacy and optionally integrity until mail reaches the trusted mail gateway. In case of sending mail, the client encrypts message with public key obtained from the certificate of trusted mail gateway. If integrity and authenticity are required, then it signs message with its private key. When the trusted mail gateway takes mail from SMTP interface, it checks to see whether it is plaintext message. If mail is plaintext, then procedure continues from steps of the processing of outgoing messages. Otherwise, if the mail sent is encrypted data, the trusted mail gateway decrypts it using its private key. If decryption is failed, the trusted mail gateway aborts sending operation, notifies sender about failure and logs the mail as invalid. If the message is decrypted successfully, the trusted mail gateway extracts the original message and checks to see if it is signed. The trusted mail gateway verifies the signature and the sending operation goes on from steps of the processing of outgoing messages as unsigned messages. If verification

fails, then trusted mail gateway aborts the sending operation and logs. Similar scenario applies when a client receives mail. The client establishes connection to POP3 or IMAP interface on the trusted mail gateway. The trusted mail gateway applies steps from steps of clients' mail access. If privacy is required for the client, the trusted mail gateway encrypts the mail with the client's public key. If the integrity of the message and the authentication of the trusted mail gateway is a must, then the trusted mail gateway signs the message before transferring the mail to the client. The mail client performs necessary decryption and verification routines if the mail obtained from POP3 or IMAP interface of the trusted mail gateway is secured.

### 3.4.2 Security of Mailbox Contents

Due to wrong configuration or bad intention, the system architecture may be damaged and clients directly establish connection with the domain mail server by-passing the trusted mail gateway. In case of outgoing mails, there is nothing to be done. However, a measure outside the scope of the trusted mail gateway can be proposed. A firewall may be placed between the domain mail server and the clients. By means of the firewall, domain users are prevented directly connecting SMTP service on the

domain mail server. Without any prevention, domain clients that do not pass through the trusted mail gateway shall send their mails insecure via SMTP. In case of incoming mails, there is a precaution in the scope of the trusted mail gateway. Since the trusted mail gateway is architecturally between domain mail server and incoming mails, it stores an incoming mail at the end of steps from processing of incoming messages as an encrypted mail. The domain mail server is issued a certificate and a matching private key from the local CA for this task. If a client tries to access its mail via POP3 or IMAP bypassing the trusted mail gateway, it will obtain scrambled characters since the mail in its mailbox is encrypted by the trusted mail gateway. If clients make access their mails following the steps from clients' mail access process, the trusted mail gateway decrypts the mail by using its corresponding private key and then transfer mail to the client.

### 3.4.3 Protocol for Exchanging Security Policies and Control Information with other Domains

The protocol has three phases:

- Key exchange phase,

- Secure channel establishment phase,

- Actual policy and information exchange phase.

Trusted mail gateways grow the network among them by establishing binary trust relations. Thus, two trusted mail gateways might exchange their keys manually and then use those pre-shared keys to communicate through secure channel. Key exchange is also achieved by employing standard protocol, ISAKMP [4] or Internet Key Exchange (IKE) Protocol [5]. Communicating peers can dynamically produce a pair of secret keys with agreed lifetimes. Secure channel is built by using the keys produced by IKE. Another solution to key exchange and secure channel establishment problem is to bundle key exchange and secure channel establishment by Secure Socket Layer (SSL) protocol. Two trusted mail gateways in the exchange might use their domain certificates in SSL handshake to establish authenticated and private channel. Policy and information exchange occur through SSL channel.

Having finished proposing solutions on key exchanges and secure channel establishment phases, it is time to describe how information exchange occurs. Two trusted mail gateways exchange information while

mail transaction is going on. The originator gateway connects pre-defined port on the recipient gateway to inform the recipient that it sends a mail. Then, the originator sends mail via SMTP. After the recipient gets mail, the recipient informs the originator about status of the mail such as successful delivery acknowledgement, anti-virus, anti-spam checks and content filtering results. Then, the information exchange finishes. The information exchange also occurs in an asynchronous way. One of the parties in the trust relation might initiate exchange and might query information and/or send data about mail exchanges between the domains. For instance, trusted mail gateway $T_1$ might desire to verify whether a user in the domain of the trusted mail gateway $T_2$ sent a message at a specific time to a user in the domain of $T_1$. $T_2$ delivers related information to $T_1$ and $T_1$ compares that information with the one in its database.

In order to exchange reliable time information among the trusted mail gateways, time-stamping protocol should be utilized in the system. There are two alternative methods: One is to deploy a trusted third party time stamping authority which does not belong to the group of trusted mail gateways. The second is that one of the trusted mail gateways declares itself as the time stamping authority. Another solution that does not

involve time stamping is to make each of the trusted mail gateways adjust its time itself.

### 3.4.4 Security of SMTP, IMAP and POP Authentication Phases

In the original forms, SMTP, POP3 and IMAP exchange all user authentication data (usernames, codes, passwords) and other protocol information in plaintext. To access incoming mails and to send mails through a secure channel, mail access (POP3 and IMAP) and mail sending (SMTP) protocols might be used over SSL. SMTP and IMAP have defined standard protocol extensions for SSL support. Also, POP3 has an SSL extension. In this way, client-to-the trusted mail gateway and the trusted mail gateway-to-the domain mail server SMTP and IMAP transactions is carried out via authenticated and private channel provided by SSL. However, user mail agents and mail transfer agents might not have reliable support for such protocol extensions. If SSL extensions to SMTP, POP3 and IMAP are made obligatory, then the interoperability property of the trusted mail gateway with existing user mail agents and mail transfer agents is damaged.

## 3.5 Traffic Logging and Relay

### 3.5.1 Notary Mechanism

The trusted mail gateway assigns a unique identifier to each of operations performed by the clients in the domain through the trusted mail gateway. In the database, it keeps the following information for each of the rows:

- Unique identifier for a mail operation as the primary search key,
- A mark that identifies a mail as incoming or outgoing,
- Mail delivery status for outgoing mails,
- Results of anti-virus, anti-spam checks and content filtering for both incoming and outgoing mails,
- Time when a mail has arrived,
- Time when a client makes access mail via POP3 or IMAP,
- Time when a client has sent a mail through SMTP,

- Authentication logs (invalid user codes, successful logon, etc.) for a client connecting to services SMTP, POP3 or IMAP,

- Mail access commands and their results (such as POP3 *retr* command),

- Some of the RFC822 and MIME header fields such as *From, To, Date, Subject, Content-Type.*

### 3.5.2 Relay Property

Let $T_1$, $T_2$ and $T_3$ be the trusted mail gateways. $T_1$ and $T_2$ have binary trust relationship. Similarly, $T_2$ and $T_3$ have binary trust relationship. A client in the domain of $T_1$ cannot send secured mail by the trusted mail gateway system to a client in the domain of $T_3$. However, mails between $T_1$ and $T_2$ are secured by the trusted mail gateway system. Similar case holds for mails between $T_2$ and $T_3$. If the trusted mail gateway $T_2$ has relay property, then mails between $T_1$ and $T_3$ shall be secured by the trusted mail gateway $T_2$. $T_1$ will send mail to $T_2$ via SMTP securely. After that, $T_2$ sends mails obtained from $T_1$ targeted $T_3$ to the trusted mail gateway $T_3$ in a secure way. In that case, there are two spliced SMTP connections secured by the trusted mail gateways.

In order to make relay property work in the trusted mail gateway system, the gateways that will take part in secured and relayed mail exchange should know trust relations between the gateways in advance. It is difficult to update such topology since topology might change often. Another problem is that the recipient domain might not want to be a part of secure transaction with the originator domain. Even, the recipient domain might not prefer to get mail from the originator domain.

## 3.6 Comparison of Trusted Mail Gateway against PGP, PEM, Yahoo DomainKeys and Microsoft Sender ID

In this section, Trusted Mail Gateway is compared against PGP, PEM, Yahoo Domain Keys and Microsoft Sender ID studies obtained from the literature.

Trusted Mail Gateway is different from the PGP and PEM in such a way that they are the mail user agent oriented solutions and Trusted Mail Gateway is a domain-based solution. It is in the same category as Yahoo DomainKeys and Microsoft Sender ID, which are also domain-level solutions. If the solutions are categorized according to the techniques

used, then Trusted Mail Gateway falls in the same category as PGP and PEM since it utilizes similar cryptographic methods to PGP and PEM.

Trusted Mail Gateway offers stronger cryptographic content protection than both mail user agent oriented and domain-level solutions. It uses 168-bit content-encryption key with triple-DES algorithm. PGP uses 128-bit content-encryption key and PEM uses 56-bit content-encryption key. Since Microsoft Sender ID and Yahoo DomainKeys do not carry out content encryption, Trusted Mail Gateway is naturally more secure than both. Trusted Mail Gateway encrypts messages content when it writes e-mails in corresponding mailboxes and it transfers e-mails to their intended recipients, which are domain users. None of the example solutions offers that property. Trusted Mail Gateway provides message integrity and origin authentication based on S/MIME digital signatures. It uses X509 certificates to verify signatures and consults certification authority, root certificate, to verify the certificates. Signature generation is the same as PGP and Yahoo DomainKeys and signature verification and key management scheme is the same as PEM. Similar to PEM and in contrast to PGP, data structures in which the protected form of message content is being carried and message formats are based on the Internet standards

describing S/MIME and CMS. There are also protection mechanisms, which are anti-virus, anti-spam and content-filter, against malicious content included in Trusted Mail Gateway whereas the example solutions in the literature do not apply such content inspection and protection techniques.

Besides cryptographic protection and content protection, Trusted Mail Gateway processes both incoming and outgoing e-mails and records processing results in database as notary data. PGP, PEM, Yahoo DomainKeys and Microsoft Sender ID do not include notary functionality. Trusted Mail Gateway establishes trust relations with other domains and applies cryptographic protection techniques according to the trust relation between itself and the other trusted domain. Also, the notary data is exchanged between Trusted Mail Gateways. Trust relation concept is not present in any of the example solutions and also there is no notary data exchange.

There is one property present in Trusted Mail Gateway and not present in PGP, PEM, Yahoo DomainKeys and Microsoft Sender ID. Trusted Mail Gateway implements proxies for SMTP and POP staying transparent to domain users. Also, it functions like domain SMTP server for incoming

mails outside the domain. The example solutions do not implement any proxy mechanism.

# CHAPTER IV

# TRUSTED MAIL GATEWAY SOFTWARE DESIGN AND IMPLEMENTATION

In this chapter, Trusted Mail Gateway software architecture, design and implementation will be explained. In the software design and implementation phase of Trusted Mail Gateway, IMAP protocol implementation is excluded since it is an alternative way of accessing mails in user mailboxes as POP3.

## 4.1 Software Architecture

Trusted Mail Gateway is architecturally between domain mail server and domain clients and is also between domain mail server and incoming mails from outside the domain. The following figure aims to remind the

position of the gateway in e-mail system and depicts interfaces of the gateway provided for domain mail clients, domain administrator, domain mail server, other trusted domains and incoming SMTP traffic.

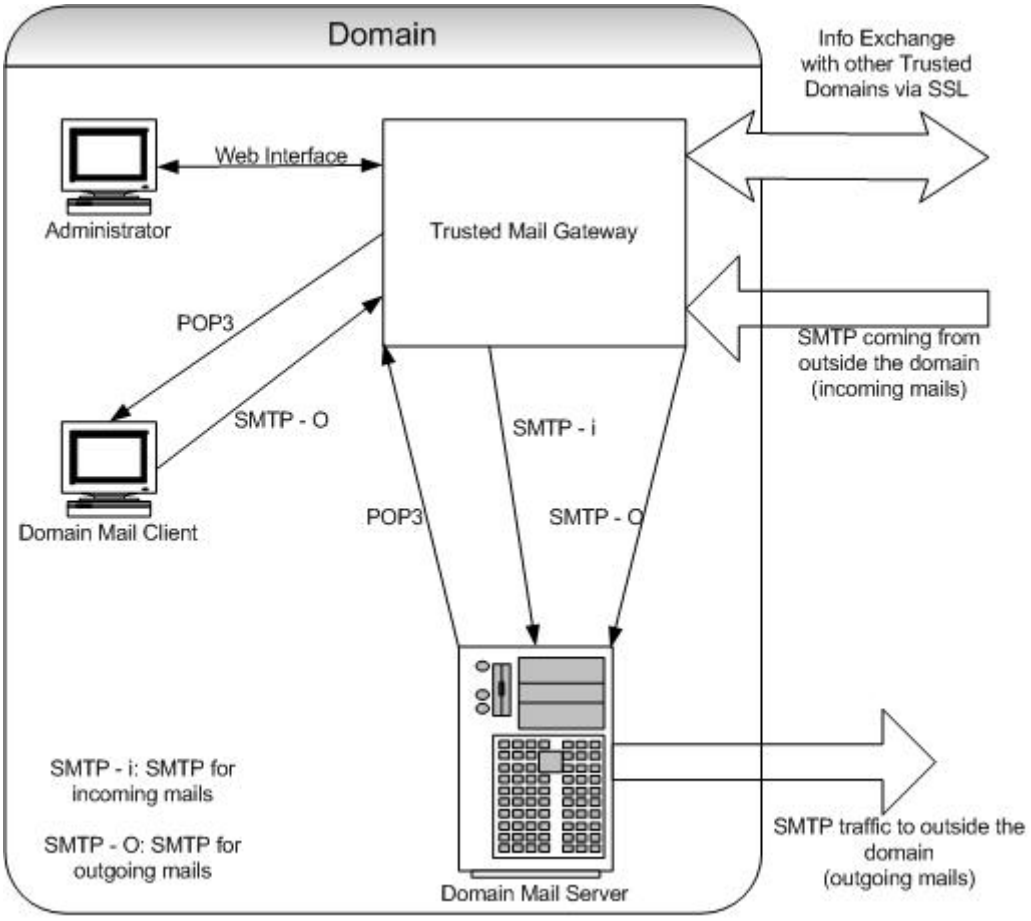


Figure 7: Position of Trusted Mail Gateway in a domain

Trusted Mail Gateway architecture and operation scenarios were explained in the previous chapter. Hereafter, detailed software architecture and tasks of the items in the architecture will be given. Items

in the software architecture shall map class implementations, libraries and third-party products. The following figure illustrates the software architecture of Trusted Mail Gateway.

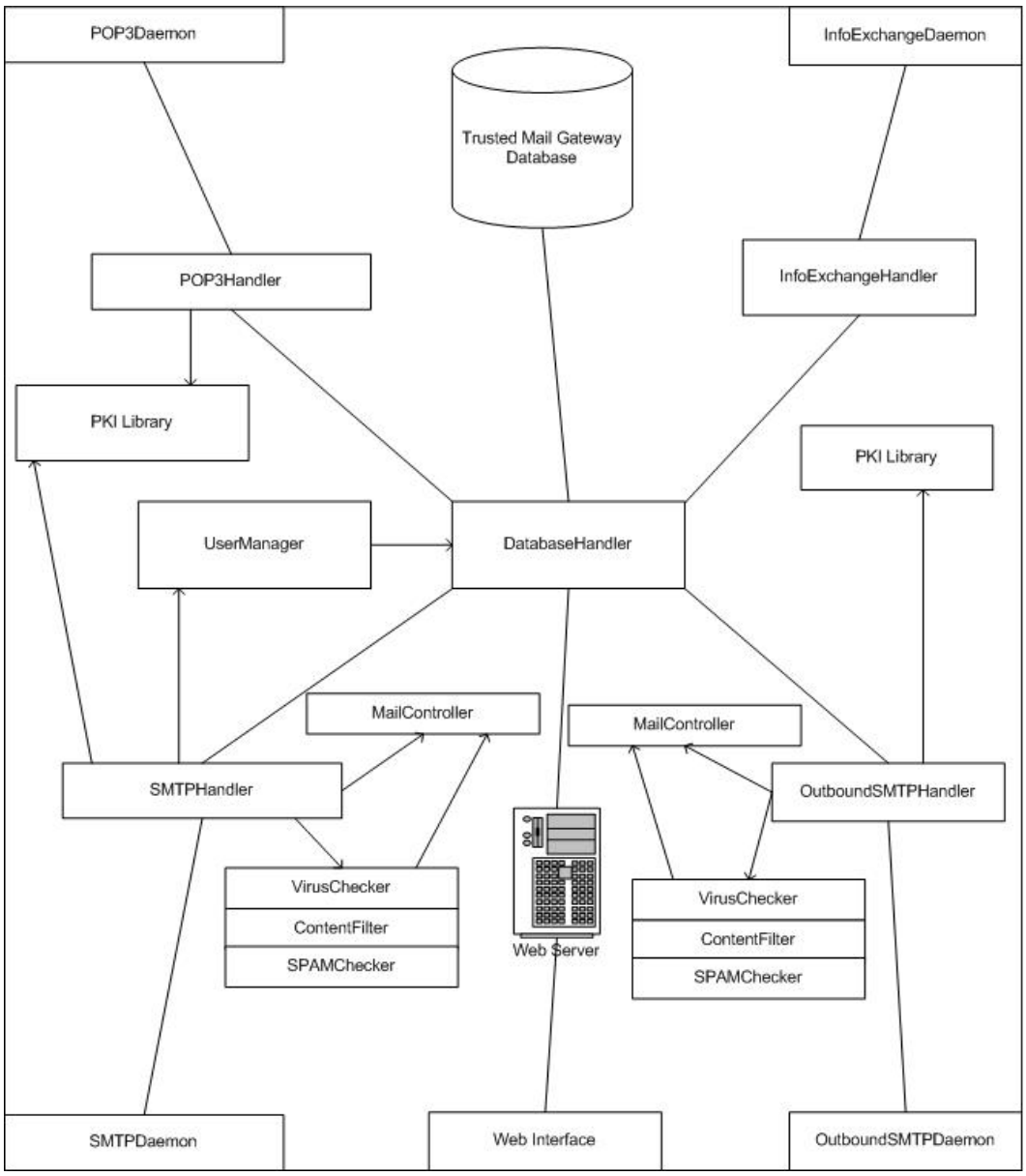


Figure 8: Trusted Mail Gateway Software Architecture

Trusted Mail Gateway software architecture is based on client-server architecture. Also, the architecture possesses similar characteristics with three-tier architectures. That is, database in the architecture is separate and handled independently and servers include the agent portion of the system that performs main system functions. In the figure, software items on the edges of the figure indicate the external interfaces of the gateway. External interfaces provide domain clients with POP3 and SMTP daemons for incoming and outgoing mails. Also, incoming mails from other domains are handled via connections provided by external interfaces. Moreover, external interfaces accomplish notary information exchange with trusted domains. Lastly, administrative tasks in domain are handled via web interface. Software items that comprise internal parts of the figure carry out main Trusted Mail Gateway functionality, which are S/MIME operations, mailbox security, notary mechanism, traffic logging, virus checking, anti-spam and content-filtering. Here, functionality of each of the items in the software architecture is being explained.

- **Trusted Mail Gateway Database:** Trusted Mail Gateway Database stores domain user information, trusted domains information, X509

certificates of the gateway and processed incoming and outgoing mail information.

- **Web Server:** Web Server runs servlets behind Web Interface. The servlets carry out administrative tasks and record results in Trusted Mail Gateway Database.

- **Web Interface:** Web Interface consists of HTML forms that obtain data from the administrator and pass the data to servlets that run behind to carry out domain administration tasks. Domain administration tasks include adding trusted domain, discarding a trusted domain and to view notary information in Trusted Mail Gateway Database.

- **DatabaseHandler:** DatabaseHandler is responsible for managing tasks on the Trusted Mail Gateway Database. The tasks include inserting, updating, deleting and querying information about domain users, trusted domains, processed mails and the gateway X509 certificates. Software items that need to access the database should employ DatabaseHandler. In other words, all database operations are accomplished via methods that DatabaseHandler provides.

- **SMTPDaemon:** SMTPDaemon accepts connection from domain mail clients for outgoing mails. This daemon creates a new thread that handles gateway operations on mail to be delivered. After creating a new thread, SMTPDaemon continues accepting new connections.

- **SMTPHandler:** The thread that is created by SMTPDaemon upon accepting a new connection from a domain client runs SMTPHandler. SMTPHandler opens a connection to the domain SMTP server. SMTPHandler exchanges commands and inspects the results of the commands between domain client and domain SMTP server. Once "DATA" command has been issued, SMTPHandler reads whole mail from the client. After obtaining mail, it applies steps from "3.3.3 Processing of Outgoing Mails". As a final action, it passes the processed mail to the domain SMTP server to deliver. SMTPHandler applies PKI operations on mail. Also, it accesses Trusted Mail Gateway Database via methods of DatabaseHandler to record notary data. Another function of SMTPHandler is to issue X509 certificates to domain users by calling methods of UserManager. SMTPHandler checks whether domain client exists in Trusted Mail Gateway Database by querying client's mail address via DatabaseHandler

methods. If there is no record, then it calls methods from UserManager to generate X509 certificate and to send the certificate and the corresponding private key to the client in PKCS12 file as mail attachment in an automatically generated mail. The password protecting privacy of the PKCS12 file shall be learnt from administrator.

- **OutboundSMTPDaemon:** OutboundSMTPDaemon accepts connection from other domains for incoming mails. This daemon creates a new thread that handles gateway operations on mail to be written in a user mailbox. After creating a new thread, OutboundSMTPDaemon continues accepting new connections from other domains.

- **OutboundSMTPHandler:** The thread that is created by OutboundSMTPDaemon upon accepting a new connection from outer domain's SMTP server runs OutboundSMTPHandler. OutboundSMTPHandler opens a connection to the domain SMTP server. OutboundSMTPHandler exchanges commands and inspects the results of the commands between the outer domain's SMTP server and domain SMTP server. Once "DATA" command has been issued,

OutboundSMTPHandler reads whole mail from the outer domain's SMTP server. After obtaining mail, it applies steps from "3.3.1 Processing of Incoming Mails". Final step is to encrypt the mail with gateway mailbox security certificate as enveloped data and to pass the processed and enveloped mail to the domain SMTP server to write corresponding user mailbox. OutboundSMTPHandler applies PKI operations on mail. Also, it accesses Trusted Mail Gateway Database via methods of DatabaseHandler to record notary data.

- **POP3Daemon:** POP3Daemon accepts connection from domain mail clients for mail access. This daemon creates a new thread that handles gateway operations on mail to be accessed. After creating a new thread, POP3Daemon continues accepting new connections.

- **POP3Handler:** The thread that is created by POP3Daemon upon accepting a new connection from a domain client runs POP3Handler. POP3Handler opens a connection to the domain POP3 server. POP3Handler exchanges commands and inspects the results of the commands between domain client and domain POP3 server. Once "RETR" command has been issued, POP3Handler reads whole mail from the domain POP3 server. After obtaining mail, POP3Handler

applies steps from “3.3.2 Client Access to Mails”. Then, it decrypts the mail that was encrypted with the gateway’s mailbox security certificate by using the private key matching the mailbox security certificate. Finally, POP3Handler envelops mail by using client’s X509 certificate if the client has a certificate and then passes it to the client. If the client issues “DELE” command, POP3Handler issues “RETR” command to the domain POP3 server and obtains mail to be deleted. After obtaining mail, it passes “DELE” command to the domain POP3 server. POP3Handler records notary data of both “RETR” and “DELE” commands as mail retrieval and mail deletion respectively into Trusted Mail Gateway Database via DatabaseHandler.

- **InfoExchangeDaemon:** InfoExchangeDaemon accepts connection from other trusted domains for notary information exchange. This daemon creates a new thread that handles information exchange operations. After creating a new thread, InfoExchangeDaemon continues accepting new connections. InfoExchangeDaemon provides authenticated SSL channel and accepts connections if client authentication is successful.

- **InfoExchangeHandler:** The thread that is created by InfoExchangeDaemon upon accepting a new connection from another trusted domain runs InfoExchangeHandler. InfoExchangeHandler records states of both incoming and outgoing mails exchanged between two trusted domains. When a trusted domain is going to deliver a mail to another trusted mail, it connects InfoExchangeDaemon of other trusted domain and issues that it is going to deliver e-mail. InfoExchangeHandler parses message, extracts peer domain name and checks if the peer is a trusted domain. If the result of the control is positive, it records incoming mail information into the database via DatabaseHandler methods. Also, InfoExchangeHandler stores mail retrieval and deletion information. InfoExchangeHandler, on the other hand, can be used to send notary data, which could be mail delivery, mail reception, mail access and deletion by a client.

- **PKI Library:** PKI Library is an external library that provides Trusted Mail Gateway software with methods that perform cryptographic operations. It supports Public Key Cryptographic Standards (PKCS), symmetric encryption algorithms, public key encryption, X509

certificate and CRL, attribute certificates, message digests, CMS and ASN1. SMTPHandler and POP3Handler use PKI Library to generate S/MIME signatures and to perform S/MIME encryption. Also, UserManager utilizes the library to produce client certificates and private keys and to bundle them in a PKCS12 structure. In Trusted Mail Gateway implementation, triple-DES [27], DES [27] and RC2 [29] symmetric encryption algorithms are supported in S/MIME enveloped message generation. Also, RSA is used as public key encryption algorithm in the implementation. RC2 symmetric encryption algorithm is used in CMS enveloped data structure with "RC2CBCParameter" [15] structure. Although PKI Library is an external library, I have coded 60% of the library and also whole of the new version of the CMS package.

- **UserManager:** SMTPHandler uses UserManager methods to produce certificates and private keys for domain clients. UserManager generates public-private key pair and certificate from the public key and records certificate with user e-mail address information into Trusted Mail Gateway Database. It returns PKCS12 structure to

SMTPHandler in order to transfer PKCS12 structure as a file attachment to the corresponding domain client.

- **MailController:** MailController is a monitor object that is responsible for thread synchronization. SMTPHandler, OutboundSMTPHandler, VirusChecker, SPAMChecker and ContentFilter use MailController methods.

- **VirusChecker:** VirusChecker runs in a thread created by both SMTPHandler and OutboundSMTPHandler for outgoing and incoming mails respectively. It is responsible for detecting viruses and worms.

- **SPAMChecker:** SPAMChecker runs in a thread created by both SMTPHandler and OutboundSMTPHandler for outgoing and incoming mails respectively. It is responsible for eliminating junk mail content.

- **ContentFilter:** ContentFilter runs in a thread created by both SMTPHandler and OutboundSMTPHandler for outgoing and incoming mails respectively. It is responsible for blocking undesired

mail content by using pre-defined text patterns, file extensions and attachment types.

## 4.2 Realization of the Software Architecture

### 4.2.1 Packages and Classes

The software architecture of Trusted Mail Gateway is realized as JAVA [22] classes. There are one top-level package, TMG, and nine sub-packages, which are AntiSPAM, AntiVirus, ContentFiltering, processors, common, InfoExchange, pop3, smtp, SSLConnectors, under the top-level package. Classes are placed in those packages. The following diagram depicts packages and classes of the Trusted Mail Gateway software. In the figure, rectangles with three rooms indicate classes and the rectangles with a small rectangle on their top indicate packages.

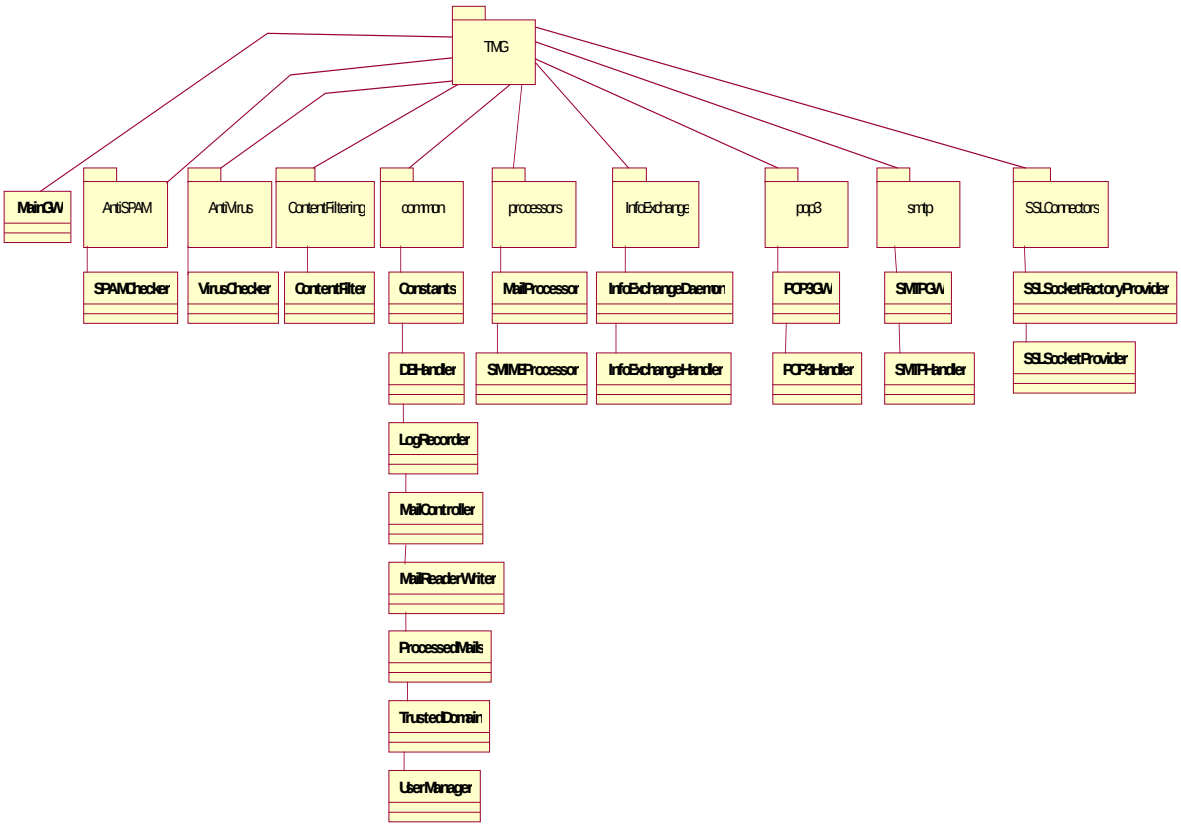


Figure 9: Trusted Mail Gateway Software Packages and Classes

Hereafter, functionality of each of the classes is explained. Also, correspondence between the classes above and software items in the software architecture is going to be given.

- **MainGW:** MainGW is the main class that initializes POP3GW, SMTPGW twice with different constructors for accepting domain connections and connections from outside the domain and InfoExchangeDaemon runnable classes. After initialization of these

runnable classes, MainGW runs them in four threads. Also, it initializes DBHandler class that immediately establishes connection to Trusted Mail Gateway Database upon initialization. MainGW is directly contained in TMG package. In the software architecture, MainGW is not visible but it creates daemons at the edges of the software architecture.

- **SPAMChecker:** SPAMChecker is a runnable class that carries out junk mail content control. It is placed in AntiSPAM package. SPAMChecker class corresponds to SPAMChecker item in the software architecture.

- **VirusChecker:** VirusChecker is a runnable class that is responsible for virus and worm detection and prevention. It is places in AntiVirus package. VirusChecker class corresponds to VirusChecker software item in the software architecture.

- **ContentFilter:** ContentFilter is a runnable class that eliminates mail content which is undesired. It uses pre-defined text patterns and file extensions in attachments. It is placed in ContentFiltering package. ContentFilter class corresponds to ContentFilter software item in the software architecture.

- **Constants:** Constants class holds public and unchanged constant values. It is placed in common package.

- **DBHandler:** DBHandler is responsible for all database operations. It provides methods to make query, update and insert records in Trusted Mail Gateway Database. Upon creation, it establishes connection to Trusted Mail Gateway Database. SMTPHandler, POP3Handler, InfoExchangeHandler and UserManager accesses Trusted Mail Gateway Database via methods of DBHandler. DBHandler corresponds to DatabaseHandler item in the software architecture. It is placed in common package.

- **LogRecorder:** LogRecorder writes operation logs of the gateway in a file in a pre-defined format. It is placed in common package.

- **MailController:** MailController is a class providing monitor object for SMTPHandler, ContentFilter, SPAMChecker and VirusChecker. That is, it is responsible for thread synchronization. The thread running SMTPHandler stops once it calls a method from MailController and threads running ContentFilter, SPAMChecker and VirusChecker resumes the thread that has stopped by another method from

MailController. MailController is placed in common package. MailController corresponds to MailController item in the software architecture.

- **MailReaderWriter:** MailReaderWriter provides methods for sending and receiving lines in a uniform way. It creates input and output streams from session socket. Via the streams, it writes and reads streams of characters in a format that POP and SMTP use. SMTPHandler and POP3Handler employ this class in mail protocol transaction. It is placed in common package.

- **ProcessedMails:** ProcessedMails class holds data in runtime that is persisted in Trusted Mail Gateway Database. It is used by servlets behind the Web Interface, as depicted in the software architecture. ProcessedMails is placed in common package.

- **TrustedDomain:** TrustedDomain class holds data in runtime that is persisted in Trusted Mail Gateway Database. It is used by servlets behind the Web Interface, which is depicted in the software architecture. TrustedDomain is placed in common package.

- **UserManager:** UserManager class provides methods for SMTPHandler class to create certificates and private keys for domain users, to insert certificate and user information into Trusted Mail Gateway Database and to prepare and send e-mail for user PKCS12 bundling a certificate and a corresponding private key. UserManager calls methods of classes in PKI Library in the software architecture to produce public-private key pairs and X509 certificates. It corresponds to UserManager item in the software architecture. UserManager is placed in common package.

- **MailProcessor:** MailProcessor class provides methods to obtain RFC822 and MIME headers and mail bodies. Also, it is used to extract values of header fields, to find out content type such as S/MIME message, MIME message, signed data and enveloped data. It can combine MIME, RFC 822 headers and body to build a whole mail message. SMTPHandler and POP3 Handler call methods of MailProcessor. It is placed in processors package.

- **SMIMEProcessor:** SMIMEProcessor class provides methods to build S/MIME signed and S/MIME encrypted messages. Also, it performs signature verification on signed S/MIME messages. Moreover, it

decrypts S/MIME enveloped messages. SMTPHandler and POP3 Handler call methods of SMIMEProcessor to sign, envelope, verify and decrypt MIME and S/MIME messages. It is placed in processors package. SMIMEProcessor class calls methods of classes in PKI Library in the software architecture to generate and verify S/MIME signatures and to perform S/MIME encryption and decryption.

- **InfoExchangeDaemon:** InfoExchangeDaemon is a runnable class that is run in a thread created by MainGW. It accepts connections from other trusted domains to exchange notary information. InfoExchangeDaemon calls methods of SSLSocketFactoryProvider and SSLSocketProvider classes to obtain sockets that establish secure and authenticated SSL channel. It corresponds to InfoExchangeDaemon item in the software architecture. It is placed in InfoExchange package.

- **InfoExchangeHandler:** InfoExchangeHandler is a class that is run in a thread created by InfoExchangeDaemon upon accepting a new connection. InfoExchangeHandler corresponds to InfoExchangeHandler item in the software architecture. Over secure and authenticated channel, it exchanges mail delivery, retrieval, deletion and reception information with other trusted domains. It

updates mail states in Trusted Mail Gateway Database by calling relevant methods in DBHandler when a peer trusted domain sends information about mail being delivered, mail received, mail retrieved and deleted. It is placed in InfoExchange package.

- **POP3GW:** POP3GW is a runnable class that MainGW initializes and runs in a thread. It accepts connections from domain clients that want to list, retrieve and delete their mails via POP3 protocol. Once a connection is obtained, it passes that connection to a new thread by creating POP3Handler and continues to accept new connections from domain. POP3GW corresponds to POP3Daemon in the software architecture. It is placed in pop3 package.

- **POP3Handler:** POP3Handler is a class that is run in a thread created by POP3GW upon accepting a new POP3 connection. POP3Handler implements POP3 commands, which are “USER”, “PASS”, “DELE”, “RETR”, “LIST”, “RSET”, “NOOP”, “STAT” and “QUIT”. POP3Handler establishes a connection to the domain POP3 server like a client. It functions transparently between the client and the domain POP3 server. It exchanges commands and inspects command results between the client and the domain POP3 server. In case of “RETR”

command, it performs S/MIME decryption by calling methods from SMIMEProcessor and reforms mail by calling methods from MailProcessor. Also, it encrypts reformed mail if client's X509 certificate exists in Trusted Mail Gateway Database. When a client issues "DELE" command, POP3Handler issues "RETR" command to the domain POP3 server. It extracts special header field values. After that, it issues "DELE" command to the domain POP3 server. In case of both "RETR" and "DELE" commands, POP3Handler updates status of the mail in Trusted Mail Gateway Database by calling relevant methods of DBHandler class. If the mail being retrieved or deleted has come from a trusted domain, then POP3Handler sends retrieval or deletion information to the trusted domain by calling a method of InfoExchangeHandler. POP3Handler corresponds to POP3Handler item in the software architecture. It is placed in pop3 package.

- **SMTPGW:** SMTPGW is a runnable class that MainGW initializes twice and runs in two different threads for both incoming and outgoing mails. It accepts connections from domain clients that want to send mail and connections from outside the domain to receive mail via SMTP protocol. Once a connection is obtained, it passes that

connection to a new thread by creating SMTPHandler and continues to accept new connections from domain and outside the domain. SMTPGW corresponds to both SMTPDaemon and OutboundSMTPDaemon in the software architecture. It is placed in smtp package.

- **SMTPHandler:** SMTPHandler is a class that is run in a thread created by SMTPGW upon accepting a new SMTP connection. SMTPHandler implements SMTP commands, which are “HELO”, “EHLO”, “MAIL (FROM)”, “RCPT (TO)”, “DATA”, “RSET”, “NOOP” and “QUIT”. It is run in two modes: One of the modes handles SMTP connections from domain for outgoing mails and the other one handles SMTP connection from outside domain for incoming mails. The mode handling outgoing mails uses MailProcessor methods to reform mails and SMIMEProcessor methods to generate S/MIME signatures or S/MIME envelopes if recipient domain is trusted. The other mode handling incoming mails also uses MailProcessor methods to reform mails and SMIMEProcessor methods to decrypt enveloped S/MIME messages or to verify S/MIME signed data if originator domain is trusted. SMTPHandler creates three threads and runs SPAMChecker,

VirusChecker and ContentFilter in them. It then stops by calling a method in monitor object MailController. When the three threads complete their tasks, they set results of the controls and resumes SMTPHandler thread. SMTPHandler enters mail delivery and mail reception states into Trusted Mail Gateway Database by calling DBHandler methods. If incoming or outgoing mail is originated or destined for a trusted domain, then SMTPHandler calls an InfoExchangeHandler method to inform the peer-trusted domain. In both of the incoming mail handling and outgoing mail handling modes, it stays transparent between peer, which is either domain client or outside domain, and the domain SMTP server. Once SMTPHandler is activated, it establishes a connection to the domain SMTP server. It exchanges commands and inspects results of the commands between domain clients or outside domain SMTP server and the domain SMTP server. It activates security functionality when “DATA” command is issued. SMTPHandler corresponds to both SMTPHandler and OutboundSMTPHandler in the software architecture. It is placed in smtp package.

- **SSLSocketFactoryProvider:** SSLSocketFactoryProvider class creates SSLServerSocketFactory and SSLSocketFactory objects by using JAVA Secure Socket Extension (JSSE) API [20]. InfoExchangeDaemon class calls methods of SSLSocketFactoryProvider class to obtain SSL socket factories.

- **SSLSocketProvider:** SSLSocketProvider class creates SSLServerSocket and SSLSocket objects by using JSSE API [20]. InfoExchangeDaemon and InfoExchangeHandler classes call methods of SSLSocketProvider class to obtain SSL server socket and SSL socket. These sockets are used to establish an SSL channel. SSLServerSocket created by SSLSocketProvider requires client authentication.

SMTPHandler and POP3Handler classes accomplish the functionality of Trust and Control Engine in Figure 3.

### 4.2.2 Administrator Interface Implementation

In the software architecture, Web Interface and Web Server items handle the administrator interface. Web Interface HTML forms and servlets run by Web Server are realized by JAVA Server Pages (JSP) [33] technology.

The JSP files that constitute Web Interface and servlets are “index”, “addDomain”, “addHandler”, “deleteDomain”, “deleteHandler” and “viewMails”. “index” is the main page that activates “addDomain” to add a trusted domain, “deleteDomain” to delete trusted domains and “viewMails” to display the processed mails incoming and outgoing. Domain administrator enters domain name, information exchange IP and port, S/MIME operation to perform when exchanging mail with the domain being added and certificates of that domain. “addHandler” and “deleteHandler” perform the domain addition and deletion operations. All JSP pages use methods of DBHandler class in TMG common package to connect, insert and change Trusted Mail Gateway Database. Web Server item in the software architecture is realized by deploying third-party product, TOMCAT 4 (version 4.1.31) [32] web server. It runs ◎ervlets codes in the JSP pages.

### 4.2.3 Trusted Mail Gateway Database Realization

Physically, Trusted Mail Gateway Database is realized by using third-party product, MySQL (version 4.0.21) [18] database server. A dedicated view for Trusted Mail Gateway Operations named “TMG” is created. Under “TMG” view, four tables are placed. The tables under view “TMG”

are “PROCESSEDMAILS”, “DOMAINUSERS”, “TRUSTEDDOMAINS” and “GWCERTIFICATES”. Here, logical data model of the database is illustrated. There is not any identifying or non-identifying relation between tables.

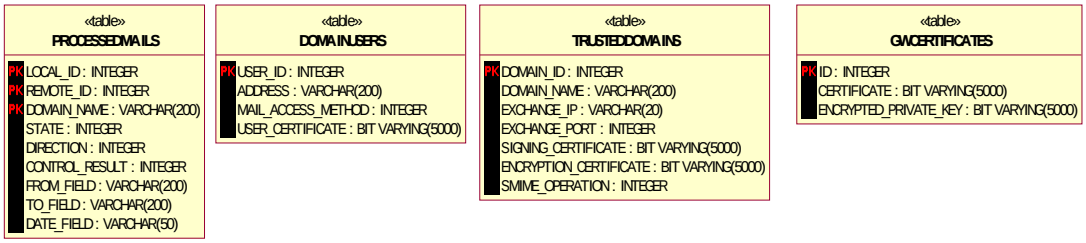


Figure 10: Trusted Mail Gateway Database Tables

Having illustrated logical data model of the Trusted Mail Gateway Database, what the tables above persist is explained.

- **PROCESSEDMAILS:** PROCESSEDMAILS table holds data of both incoming and outgoing mails passing through Trusted Mail Gateway. Each mail in the table has LOCAL_ID. REMOTE_ID comes with incoming mails from other trusted domains in a special RFC822 header in the protected part of the e-mail. In the database, only incoming mails have REMOTE_ID. DOMAIN_NAME indicates where a mail comes from or is destined to. LOCAL_ID, REMOTE_ID and

DOMAIN_NAME constitute primary key, which uniquely identifies a row. Status information of the mails is hold in STATE column. DIRECTION depicts whether a mail is incoming or outgoing or intra-domain mail. DIRECTION information also gives whether a mail is from or to trusted domain. FROM_FIELD and TO_FIELD represent sender and receiver of the mail respectively. DATE_FIELD is the time information when a mail has been received from SMTP system or a mail has been delivered.

- **DOMAINUSERS:** DOMAINUSERS table holds mail addresses, mail access methods and X509 certificates of mail users in the domain. USER_ID is a unique identifier for each user in the database. ADDRESS holds user's mail address. MAIL_ACCESS_METHOD indicates whether a user accesses his/her mails via POP3 or IMAP. USER_CERTIFICATE holds user's X509 certificate in binary format.

- **TRUSTEDDOMAINS:** TRUSTEDDOMAINS table holds trusted domain information. Each domain in the table is given a unique number, which is DOMAIN_ID. Also, DNS name of the domain is recorded in DOMAIN_NAME column. In order to exchange notary information with the trusted domain persisted in a row, information

exchange daemon IP address and port are stored in EXCHANGE_IP and EXCHANGE_PORT respectively. To verify signed S/MIME messages coming from a trusted domain, signing certificate of that trusted domain is taken from SIGNING_CERTIFICATE column. In order to send enveloped S/MIME message to that trusted domain, X509 certificate for key encryption is obtained from ENCRYPTION_CERTIFICATE column. SMIME_OPERATION column depicts whether to sign or encrypt outgoing messages to that trusted domain. Also, it is also an option not to perform S/MIME operation.

- **GWCERTIFICATES:** GWCERTIFICATES table persists domain certificates and encrypted private keys in binary format. ID is the unique identifier. CERTIFICATE and ENCRYPTED_PRIVATE_KEY columns hold X509 certificate and a corresponding private key encrypted with pre-defined password respectively. Intra-domain root certificate, mailbox security certificate, intra-domain S/MIME certificate, inter-domain signing certificate and inter-domain encryption certificate with corresponding encrypted private keys are placed in this table.

### 4.2.4 Technology Used to Implement Trusted Mail Gateway Software

The following technology is used to build Trusted Mail Gateway Software.

- JAVA 2 Software Development Kit (J2SDK) Version 1.4.2_06 [21] to implement software in JAVA programming language,

- JAVA Server Pages (JSP) [33] to implement the web interface forms and servlets,

- TOMCAT Version 4.1.31 [32] Web Server to display forms and to run servlets in the web interface,

- MySQL Version 4.0.21 Database Server [18] to realize Trusted Mail Gateway Database,

- JAVA Database Connectivity (JDBC) [19],

- MySQL Connector Version 3.0.15 [18] for JAVA to connect and manipulate the database via JDBC,

- JAVA Secure Sockets Extension (JSSE) [20] to establish SSL channel for notary information exchange.

As test, deployment and realization environment, the following technology is utilized:

- Linux Slackware Version 8.0 Kernel Version 2.4.5 [34] operating system to host the Trusted Mail Gateway software,

- Sendmail version 8.13.1 [35] mail server to provide domain with SMTP server,

- GNU pop3d [36] to provide domain with POP3 daemon.

## 4.3 Trusted Mail Gateway Software in Operation

In this section, how Trusted Mail Gateway software operates is explained. Trusted Mail gateway is run with a configuration file consisting of domain SMTP and POP server addresses and ports, database address, port, user name, password and database view information, IP addresses and port numbers for inbound and outbound SMTP gateways and POP gateway and IP address and port number for "InfoExchangeDaemon" in order to

listen for notary data exchange requests. When Trusted Mail Gateway is launched, it creates and runs four threads that are SMTP thread for domain users (“SMTPGW”), SMTP thread for incoming mails (“SMTPGW” for incoming mails), POP thread for domain users to access mailboxes (“POP3GW”) and a thread for notary information exchange (“InfoExchangeDaemon”). It also establishes connection to the database where notary data, trusted domains information, domain user information and certificates of the gateway are kept. Besides configuration file, there is also PKCS12 file and password given as application parameter in order to establish secure SSL channel with peer authentication.

### 4.3.1 Incoming Mail Scenario

1. If an originator domain is a trusted mail gateway, it establishes SSL connection to InfoExchangeDaemon. If peer authentication is not successful, InfoExchangeDaemon closes the connection and logs the unsuccessful SSL connection by using LogRecorder methods.

2. Once SSL connection is established, InfoExchangeDaemon creates InfoExchangeHandler thread.

3. Originator gateway passes “Mail Being Delivered” state information with “Remote_ID” data to InfoExchangeHandler.

4. InfoExchangeHandler closes connection and inserts a new record with “Local_ID” –1 and “Remote_ID” just received by issuing a method from “DBHandler”. State information of new record is “Mail Being Received”.

5. After a while, SMTP connection comes to “SMTPGW” running in the mode to accept incoming mails.

6. “SMTPGW” creates “SMTPHandler” in a thread and “SMTHandler” starts SMTP transaction.

7. Once the originator domain issues “MAIL FROM”, “SMTPHandler” extracts the domain name from the mail address obtained in “MAIL FROM” and checks in “TRUSTEDDOMAINS” table whether the domain sending this mail is a trusted domain.

8. Once the originator domain issues “RCPT TO”, “SMTPHandler” searches the address extracted from “RCPT TO” in the

"DOMAINUSERS" table to check whether the domain user has an X509 certificate. If not, "SMTPHandler" employs "UserManager". "UserManager" generates public-private key pair, creates a certificate and inserts certificate with user e-mail address to "DOMAINUSERS" table and bundles the certificate with the private-key in a password protected PKCS12 structure. "UserManager" issues this PKCS12 structure to the domain user in a separate e-mail message as an attachment.

9. Once the peer domain issues "DATA" command, "SMTPHandler" receives e-mail in lines. After reception is completed, "SMTPHandler" checks the operation on the message according to whether the originator domain is a trusted domain obtained in step (7). If it is an enveloped data S/MIME message, "SMTPHandler" gets its inter-domain S/MIME encryption private-key from the "GWCERTIFICATES" table and decrypt the message. If the message is S/MIME multipart signed message, then "SMTPHandler" verifies and removes the signature. If either of decryption or signature verification is failed, "SMTPHandler" issues a mail message warning the recipient

that the mail has been blocked and logs the malformed S/MIME message information using “LogRecorder” and the operation finishes.

10. “SMTPHandler” creates three threads for “VirusChecker”, “SPAMChecker” and “ContentFilter” and passes the original mail message to them. Thread running “SMTPhandler” stops until three threads finish their tasks. If there is any infection or spam or undesired content, “SMTPHandler” issues a mail message warning the recipient that the mail has been blocked and logs necessary information about failure using “LogRecorder” and operation finishes.

11. “SMTPHandler” generates a new “Local_ID” for the incoming mail. If the mail is coming from a trusted domain, “SMTPHandler” extracts “Remote_ID” information from the protected part of the mail.

12. “SMTPHandler” updates the record created in step (4) with new “Local_ID”, mail sender and recipient addresses and reception date of the mail. State information of the mail is updated to “Mail Received”. The old state was “Mail Being Received”. If the mail is not sent by a trusted domain, a new record is created with newly generated

"Local_ID" and "Remote_ID" –1. The state information shall be "Mail Received" directly.

13. "SMTPHandler" inserts "Local_ID", "Remote_ID" and sender domain name in a special custom-built header field into the e-mail message". Also, "SMTPHandler" encapsulates the message into content-type "message/rfc822".

14. "SMTPHandler" obtains mailbox security certificate from the database and produces S/MIME enveloped data from the mail obtained in step (13) using that certificate. "SMTPHandler" passes the enveloped mail to the domain SMTP server in order to write in corresponding user mailbox.

15. If the originator domain is a trusted domain, then "SMTPHandler" issues a static method from "InfoExchangeHandler". "SMTPHandler" connects to the peer domain and sends "Mail Received" information with "Remote_ID" that has sent in the protected part of the mail message.

If the sender domain is not a trusted domain, the steps performing mail state information exchange (notary data exchange) and carrying out S/MIME signature verification or S/MIME decryption shall be absent. The automatically generated mails are also written in mailbox as S/MIME enveloped data encrypted with mailbox security certificate.

### 4.3.2 Outgoing Mail Scenario

1. A domain client connects to “SMTPGW” for domain users.

2. “SMTPGW” creates a thread that runs “SMTPHandler”.

3. Domain client starts SMTP transaction with “SMTPHandler”.

4. Once the client issues “MAIL FROM”, “SMTPHandler” searches the address extracted from “MAIL FROM” in the “DOMAINUSERS” table to check whether the domain user has an X509 certificate. If not, “SMTPHandler” employs “UserManager”. “UserManager” generates public-private key pair, creates a certificate and inserts certificate with user e-mail address to “DOMAINUSERS” table and bundles the certificate with the private-key in a password protected PKCS12

structure. “UserManager” issues this PKCS12 structure to the domain user in a separate e-mail message as an attachment.

5. “SMTPHandler” creates three threads for “VirusChecker”, “SPAMChecker” and “ContentFilter” and passes the original mail message. Thread running “SMTPhandler” stops until three threads finish their tasks. If there is any infection or spam or undesired content, “SMTPHandler” issues a mail message warning the sender that the mail has been blocked and logs necessary information about failure using “LogRecorder” and operation finishes.

6. Once the client issues “RCPT TO”, “SMTPHandler” extracts the domain name from the mail address obtained in “RCPT TO” and checks in “TRUSTEDDOMAINS” table whether the domain where the mail is destined to a trusted domain.

7. “SMTPHandler” generates a new “Local_ID” for the mail that is going to be delivered.

8. Once the client issues “DATA” command, “SMTPHandler” receives e-mail from the client in lines. After reception is completed,

"SMTPHandler" checks the operation that shall be applied on the message according to whether the destination domain is a trusted domain obtained in step (6). If the operation with the destination domain is encryption, then "SMTPHandler" gets encryption certificate of the peer domain from the "TRUSTEDDOMAINS" table. If the operation is S/MIME signing, then "SMTPHandler" gets its inter-domain signing certificate and private key from "GWCERTIFICATES" table. "SMTPHandler" inserts "Local_ID" number into the mail in a special custom-built header fields as "Remote_ID" for destination trusted domain. In both cases, "SMTPHandler" employs "SMIMEProcessor" to produce S/MIME enveloped or multipart-signed message with the certificates or the private key obtained from the database. If there is no operation related to that trusted domain or the destination domain is not a trusted domain, then there shall be no S/MIME operation on the mail.

9. If the destination domain is a trusted domain, then "SMTPHandler" creates a new record in the database by using methods from "DBHandler" with state "Mail Being Delivered" with local id number "Local_ID" generated in step (7). It also employs

"InfoExchangeHandler" in order to send a message via SSL channel to destination domain containing "Remote_ID" generated in step (7) and state "Mail Being Delivered". If the destination domain is not a trusted domain, a new record in the database is created with again "Local_ID" from step (7) and state "Mail Delivered" and there is no information exchange.

10. "SMTPHandler" passes the processed mail to the domain SMTP server and the domain SMTP server shall deliver the mail according to its internal configuration.

11. If the destination domain is a trusted domain and the e-mail has been received, the destination domain issues "Mail Received" state information with "Remote_ID" that is "Local_ID" in the originator domain via secure channel through a connection to "InfoExchangeDaemon". "InfoExchangeDaemon" passes the connection to "InfoExchangeHandler" and "InfoExchangeHandler" alters e-mail state in the database as "Mail Delivered" using "Remote_ID" information as "Local_ID".

If the destination domain is not a trusted domain, the steps performing mail state information exchange (notary data exchange) and carrying out S/MIME signature generation or S/MIME encryption shall be absent. The automatically generated mails are also written in mailbox as S/MIME enveloped data encrypted with mailbox security certificate.

### 4.3.3 Accessing E-mails in Mailboxes

1. Domain client establishes connection to “POP3GW” service of Trusted Mail Gateway software.

2. “POP3GW” creates “POP3Handler” object and runs “POP3Handler” in a thread. “POP3Handler” shall handle POP transaction with the client. “POP3Handler” stays transparent in the POP protocol transaction.

3. Once the client issues mail retrieval command “RETR”, “POP3Handler” obtains corresponding e-mail from the domain POP3 server.

4. “POP3Handler” decrypts the e-mail by using the corresponding private key of mailbox security certificate obtained from the

"GWCERTIFICATES" table in the database. If a client reaches the domain POP3 server directly without passing through Trusted Mail Gateway, there would be no decryption and the client would obtain useless e-mail.

5. "POP3Handler" extracts header fields carrying "Local_ID", "Remote_ID" and sender domain name. "POP3Handler" extract the mail from the "message/rfc822" structure. "POP3Handler" updates the record in the database related to the retrieved e-mail by using "Local_ID", "Remote_ID" and sender domain name data. The state of the mail is changed to "Mail Retrieved".

6. "POP3Handler" checks whether the sender domain is a trusted domain by issuing a query in the database. If the sender domain is a trusted domain, then "POP3Handler" uses an "InfoExchangeDaemon" method to exchange notary data about mail state "Mail Retrieved" with "Remote_ID".

7. "POP3Handler" gets user certificate from "DOMAINUSERS" table and employs "SMIMEProcessor" to produce S/MIME enveloped data. "SMIMEProcessor" removes custom-built headers about "Local_ID",

"Remote_ID" and sender domain name and generates S/MIME enveloped data by encrypting the mail with the user certificate. The user should install PKCS12 file carrying the private key and the corresponding X509 certificate before attempting to read the e-mail.

8. "POP3Handler" transfers the S/MIME enveloped e-mail to the client.

9. If the client issues mail deletion command "DELE", "POP3Handler" follows the same steps performed in "RETR" case besides transferring e-mail to the user. "POP3Handler" updates state of the e-mail in the database as "Mail Deleted" and exchanges this notary information if the sender domain of the e-mail is a trusted domain.

## 4.4 Test Results of Trusted Mail Gateway Software

Trusted Mail Gateway software has been deployed on two workstations running Linux Slackware 8.0 operating system, MySQL 4.0.21 database server, TOMCAT 4.1.31 web server and servlets engine, J2SDK 1.4.2_06, sendmail 8.13.1 SMTP server and GNU pop3d POP3 server. Both workstations are on the same network segment. The workstations host two domains, which are named as "seclab1.ceng.metu.edu.tr" and

"seclab2.ceng.metu.edu.tr" respectively. Domain "seclab1.ceng.metu.edu.tr" performs S/MIME signing when sending e-mail to "seclab2.ceng.metu.edu.tr" and the domain "seclab2.ceng.metu.edu.tr" performs S/MIME encryption when sending e-mail to "seclab1.ceng.metu.edu.tr".

```
From: "Erkut Seclab1" <erkut@seclab1.ceng.metu.edu.tr>
To: <besiktas@seclab2.ceng.metu.edu.tr>
Subject: a test mail
Date: Tue, 4 Jan 2005 10:07:28 +0200
MIME-Version: 1.0
Content-Type: text/plain;
        format=flowed;
        charset="iso-8859-9";
        reply-type=original
Content-Transfer-Encoding: 7bit
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180

this is a test mail
issued from one trusted domain to another
```

Figure 11: Original e-mail message

The figure above is the original form of the mail being sent from "seclab1.ceng.metu.edu.tr" to "seclab2.ceng.metu.edu.tr". The part of the mail that is going to be protected using S/MIME is shown below.

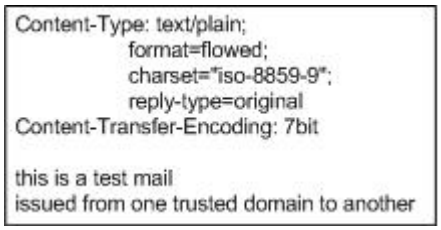


Figure 12: Protected part of the e-mail message

Trusted Mail Gateway software deployed on "seclab1.ceng.metu.edu.tr" shall insert "X-TMG-Remote_ID: 32" header into the data above and sign it to generate S/MIME signature. The domain "seclab1.ceng.metu.edu.tr" passes the following S/MIME multipart-signed mail to "sendmail" software installed on the same workstation to deliver.

```
MIME-Version: 1.0
Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1;
boundary=essek
Message-ID: <000501c4f234$6ed2b3a0$9d03040a@havelsan.intranet>
From: "Erkut Seclab1" <erkut@seclab1.ceng.metu.edu.tr>
To: <besiktas@seclab2.ceng.metu.edu.tr>
Subject: a test mail
Date: Tue, 4 Jan 2005 10:07:28 +0200
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2900.2180
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2180

--essek
Content-Type: text/plain;
        format=flowed;
        charset="iso-8859-9";
reply-type=original
Content-Transfer-Encoding: 7bit
X-TMG-Remote_ID: 32

this is a test mail
issued from one trusted domain to another


--essek
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s

MIIDNAYJKoZIhvcNAQcCoIIDJTCCAyECAQExCzAJBgUrDgMCGgUAMAsGCSqGSIb3
DQEHAaCCAgkwggIFMIIBcKADAgECAgEBMAsGCSqGSIb3DQEBBTBKMTswOQYDVQQD
EzJUcnVzdGVkIE1haWwgR2F0ZXdheSBJbnRlciBEb21haW4gUm9vdCBDZXJ0aWZp
Y2F0ZTELMAkGA1UEBhMCVFIwHhcNMDQxMjA2MTQ1MTE1WhcNMTAxMjA1MTQ1MTE1
WjA3MSgwJgYDVQQDEx9UTUcgSW50ZXItRG9tYWluIFMvTUlNRSBTaWduaW5nMQsw
CQYDVQQGEwJUUjCBnjANBgkqhkiG9w0BAQEFAAOBjAAwgYgCgYBwHzSNc+9Lh3Z6
H+BvbtLRrC1Ok3Q+A4Aw5jyzQhc8ntN3e2RpHuUUeFLBY7Z8saVvotJhkksISq4b
/CKCCKPXKhfjTqFqJWnmmEnJja40YJCb2d8+RS6dFqckt3IO9GNb13hPW8OkvwqJ
Qh+YuzV8gDRxiosUdMz/r2Fri7EvxwIDAQABoxMwETAPBgNVHRMBAQAEBTADAQEA
MAsGCSqGSIb3DQEBBQOBgQAX0doexg/Tkxrdevl4HLjcRdmfFU96VMxXMfTgjFcA
d0SZCneOV+6+OaXrDCS1TkUEi0P7mk0q2VPjD+1hDLw3GtZdAnZshBuMf1sG10kn
4gjkq7cr9KnQIGKEPXCoQFhJwUOXw5iA6fSEOzupY5N9kkmWuivT6UNXs6ZYy4r/
HDGB9DCB8QIBATBPMEoxOzA5BgNVBAMTMlRydXN0ZWQgTWFpbCBHYXRld2F5IElu
dGVyIERvbWFpbiBSb290IENlcnRpZmljYXRlIMQswCQYDVQQGEwJUUgIBATAJBgUr
DgMCGgUAMA0GCSqGSIb3DQEBAQUABIGAGn90EwbXfL1DxLT+EpaLAOqHDZYcXrnA
ewRd6MCAIm815wAVLm3Qaqj+suqnqLVfFwY3uEMyMcdKz1pCQ91KTYze0aMnI3cE
+Z20PeKTZbQuf/G8rWeYzbFl9kGRqMserMUy6Wtrd3ShyhMibtlddSPrNEx9H4dj
92XadAIJ1eI=

--essek--
```

Figure 13: S/MIME Multipart-signed e-mail message

Signature part of the message is in base64 format. Once the domain "seclab2.ceng.metu.edu.tr" receives and verifies the above message, it

produces the S/MIME enveloped data to write into the recipient user mailbox from the following data.

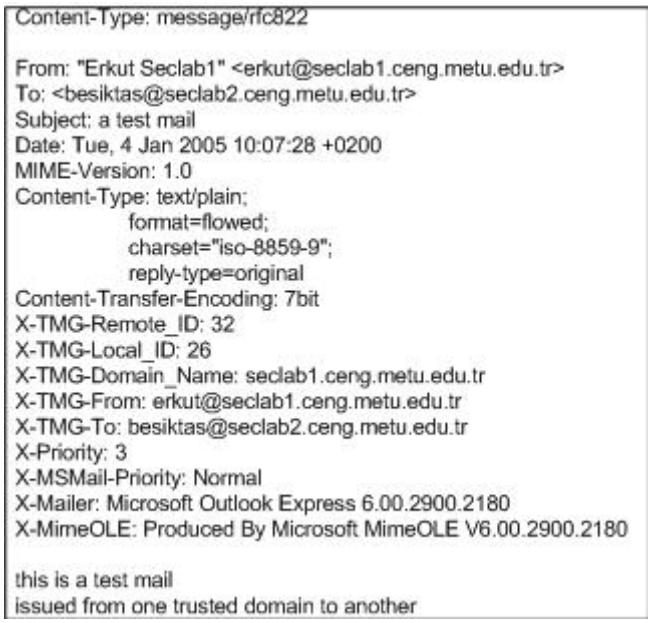


Figure 14: Format of the e-mail message prior to being written into mailbox

The following figure shows the S/MIME enveloped data produced using mailbox security certificate to be written in user mailbox.


Figure 15: S/MIME enveloped form of the e-mail message written into mailbox

If the same e-mail were sent from “seclab2.ceng.metu.edu.tr” to “seclab1.ceng.metu.edu.tr”, inter-domain S/MIME processing would change to S/MIME enveloping and Trusted Mail Gateway in the domain “seclab2.ceng.metu.edu.tr” would produce the following e-mail.


Figure 16: S/MIME enveloped form of the e-mail message delivered

The base64 portion of the message indicates the encrypted form of the original message.

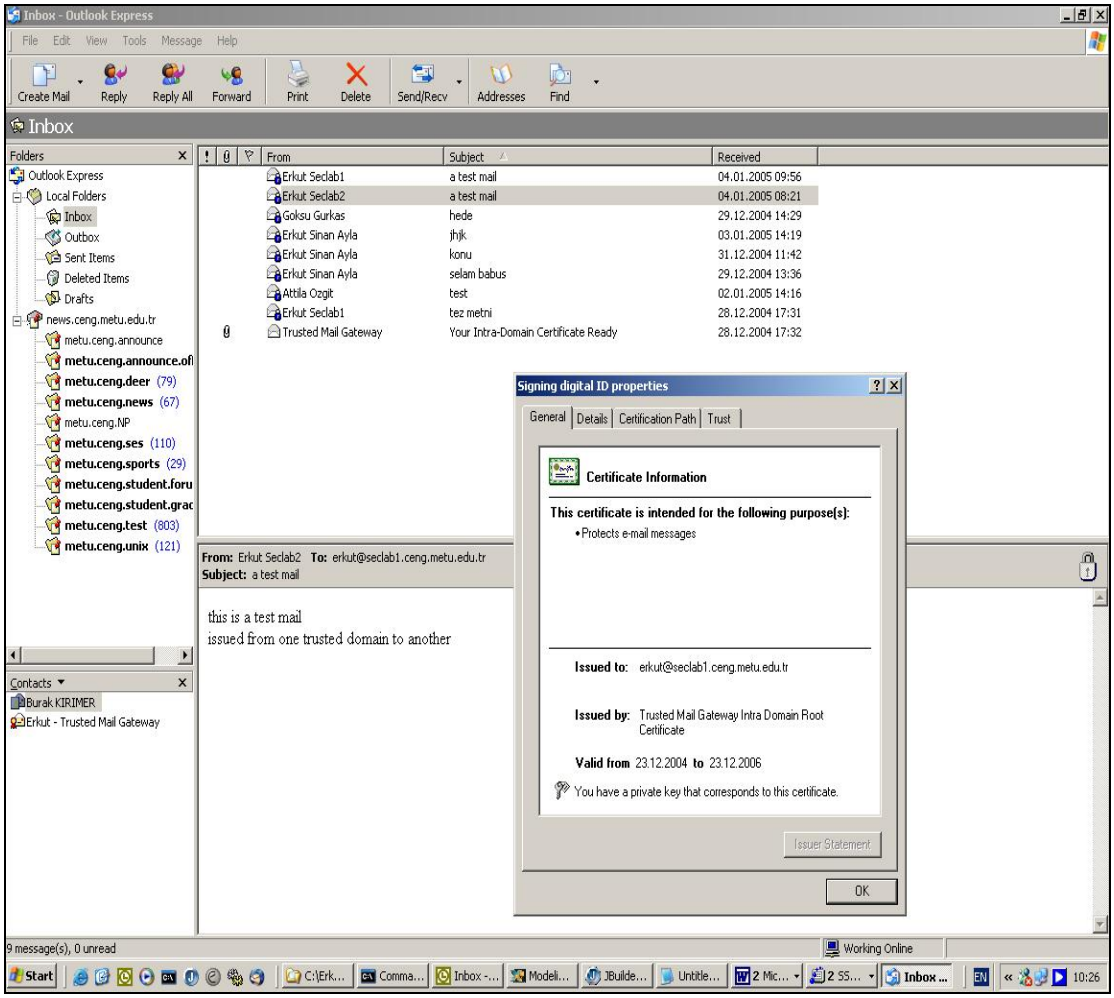


Figure 17: MS Outlook Express window

The MS Outlook Express screen shot figure above shows e-mail message that domain "seclab2.ceng.metu.edu.tr" delivered to domain "seclab1.ceng.metu.edu.tr". Although the mail sender user in domain "seclab2.ceng.metu.edu.tr" did not perform any S/MIME enveloping operation, mail recipient in domain "seclab1.ceng.metu.edu.tr" has received e-mail as S/MIME enveloped data that was produced using the

recipient's X509 certificate that is also shown in the figure. There is a "lock" image on the figure that indicates the e-mail is encrypted. Since the corresponding private key is installed on the mail client software, the recipient can open the e-mail.

# CHAPTER V

## SUMMARY, CONCLUSIONS AND FUTURE WORK

In this chapter, a summary of what has been researched and done, conclusions reached at the end of the study and possible future enhancements for the Trusted Mail Gateway study are explained.

### 5.1 SUMMARY

The Trusted Mail Gateway provides domains with e-mail security solutions. It is deployed into a domain that already possesses mail server(s) and clients. The Trusted Mail Gateway is placed between the domain mail server and the domain clients. It is interoperable with existing mail server software and user agent software.

The Trusted Mail Gateway provides four security services, which are confidentiality, non-repudiation, authentication and integrity. It generates

S/MIME signatures and decrypts encrypted mails from other the trusted domains on behalf of the domain. Also, the Trusted Mail Gateway encrypts messages before transporting them to other trusted domains. Through the insecure communication network, such as the Internet, those S/MIME signatures and encryption provide the four security services. Also, the Trusted Mail Gateway keeps messages in the mailbox as encrypted mail to enhance intra-domain security. Intra-domain S/MIME, moreover, supplies the four security services across the domain. There are precautions against malicious content. Anti-virus checks and virus protection are applied to both incoming and outgoing messages. Based on pre-defined text patterns and attachment extensions, there is a content-filtering mechanism. Requirement to defeat spam mail is also met. In addition to all these properties, Trusted Mail Gateway keeps notary information in the database. It records mail status, mail direction, sender and receiver information and mail control result indicating virus check, spam control and content-filtering. Through authenticated and secure SSL channel, Trusted Mail Gateway exchanges the notary information in the database with other trusted domains.

## 5.2 CONCLUSIONS

It is advantageous to place a security gateway, in the case of this study Trusted Mail Gateway, between mail clients and domain mail server and between domain mail server and incoming mails from the outside the domain. Here, security concept consists of all tasks that are application of S/MIME enveloping on e-mails, generation of S/MIME signatures from e-mails, signature verification and enveloping e-mail decryption, exchanging notary information, performing virus check, content filtering and spam control on e-mails entering and leaving the domain. By issuing all the security responsibility to a single node in the domain, enterprises do not have to install security functionalities to each of the client nodes. It is expensive to deploy such functionalities to each of clients in the domain. Also, it is hard to maintain such complex system.

Trusted Mail Gateway does not prevent mail traffic of the domain with domains that have not coupled with Trusted Mail Gateway. Trusted Mail Gateway checks destination domain and if the destination domain is not trusted domain, e-mail is delivered without any S/MIME processing and any notary information exchange.

It is easier to apply security policies launched at single point in the domain. For instance, there would be a requirement to deliver all e-mails as encrypted messages to some destination domain. All clients should have been warned that they would encrypt e-mails being sent to that domain. Every time security policy changes, such a warning would be repeated to all the domain clients. In case of Trusted Mail Gateway, however, security policies are set without client awareness and they are applied transparently. By using administrative web interface, domain administrator specifies a trusted domain with a security operation. Trusted Mail Gateway processes incoming and outgoing e-mails according to such policy specifications.

Trusted Mail Gateway keeps mailbox contents secure by carrying out S/MIME enveloping on e-mails prior to recording them into mailboxes. Since only Trusted Mail Gateway can access the corresponding private key used to decrypt mailbox contents, contents cannot be disclosed. Such a mechanism prevents domain users accessing mailboxes via POP3 without passing through the gateway. If attempted, useless content that cannot be decrypted shall be obtained.

Trusted Mail Gateway listens for incoming mails on behalf of the domain SMTP server. This interface of the gateway for incoming mails prevents any malformed S/MIME data, infected content, and undesired information and spam mail to enter users' mailboxes.

It is advantageous to have notary and traffic logging mechanism in the domain. Trusted Mail Gateway records delivery and reception status information of all e-mails, security control results, sender and recipient data, peer domain name, delivery and reception time with unique identifiers. In this way, all events occurring during mail transaction processes are under control.

Security functions that are virus control, spam check, content filtering and S/MIME operations require much processing power. It is expensive to supply such hardware offering huge memory and high processing power for each of clients in a domain to perform the security functions fast. Instead, the hardware with huge memory and high processing power is installed once to host the security gateway that is Trusted Mail Gateway. Hence, security launched at single point lowers the cost paid for security functionality.

It is a risk to keep private keys of intra-domain root certificate, inter-domain S/MIME signing certificate and inter-domain S/MIME enveloping certificate although they are in encrypted form. It would be difficult to recover damages that were arisen from stolen private key in case of security gateway since security gateway carries out security functions on behalf of the domain where it is installed. However, it is easier to recover from losses caused by stolen private key of a client since client is only responsible for his/her security.

In general, it is a good and easily adapted solution to place a mail security gateway in a domain, which is Trusted Mail Gateway in the scope of the study. It does not bring any serious alteration on the existing e-mail system infrastructure of the domain. In other words, it works with existing mail server and mail clients in the domain. Since the cryptographic functions of Trusted Mail Gateway are based on the standards and content protection functions are based on state-of-the-art technologies, it is efficient to set up and to spread domains that run Trusted Mail Gateway. This would be a great leap against mail security problems.

## 5.3 FUTURE WORK

This study, the Trusted Mail Gateway, could be enriched by involving relay property with security options. The gateway shall explore the topology consisting of trust relations and shall serve as a relay server between the trusted mail gateways that do not have trust relation but have a trust relation with the relay. Another development area is to secure authentication phases of the SMTP, POP and IMAP. It is important to provide a generic method to secure authentication phases of the mail protocols since they transport user account authentication data. It shall be very useful to implement virtual domain support in Trusted Mail Gateway also. The most immediate but the least crucial addition to the Trusted Mail Gateway software is IMAP gateway implementation.

# REFERENCES


[1] Klensin J, Editor AT&T Laboratories. Network Working Group Request for Comments:2821 "Simple Mail Transfer Protocol", 2001

[2] Myers J, Rose M. Network Working Group Request for Comments: 1939 "Post Office Protocol – Version 3", 1996

[3] Crispin M. Network Working Group Request for Comments: 2060 " INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", 1996

[4] Maughan D, Schertler M, Schneider M, Turner J. Network Working Group Request for Comments: 2408 “Internet Security Association and Key Management Protocol (ISAKMP)”, 1998

[5] Harkins D, Carrel D. Network Working Group Request for Comments: 2409 “The Internet Key Exchange (IKE)”, 1998

[6] Borenstein N, Freed N. Network Working Group Request for Comments: 1521 “MIME (Multipurpose Internet Mail Extensions)”, 1993

[7] Dean T, Ottaway W. Network Working Group Request for Comments: 3183 “Domain Security Services Using S/MIME “, 2001

[8] Smith, Richard E. “Internet Cryptography” pp: 267 – 294, 1997

[9] Tanenbaum, Andrew S. “Computer Networks (Third Edition)” pp: 577 – 620 and pp: 643 – 669, 1996

[10] An RSA Laboratories Technical Note – Public Key Cryptographic Standards # 1 (PKCS #1)

[11] An RSA Laboratories Technical Note – Public Key Cryptographic Standards # 7 (PKCS #7)

[12] Hously R, Ford W, Polk W, Solo D. Network Working Group Request for Comments: 2459 “Internet X.509 Public Key Infrastructure Certificate and CRL Profile “, 1999

[13] An RSA Laboratories Technical Note – Public Key Cryptographic Standards # 12 “Personal Information Exchange Syntax” (PKCS #12)

[14] Crocker, David H. Network Working Group Request for Comments: 822 " STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", August 13 1982.

[15] Housley R. Network Working Group Request for Comments: 3370 “Cryptographic Message Syntax (CMS) Algorithms”, August 2002

[16] Ramsdell B. Network Working Group Request for Comments: 3851 “Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification”, July 2004

[17] Housley R. Network Working Group Request for Comments: 3852 “Cryptographic Message Syntax (CMS)”, July 2004

[18] “The World’s Most Popular Open Source Database”, “http://dev.mysql.com”, last accessed at 2004, 15 December.

[19] “JDBC Technology”, ”http://java.sun.com/products/jdbc/, last accessed at 2004, 15 December.

[20] “Java Secure Socket Extension (JSSE)”, “http://java.sun.com/products/jsse/”, last accessed at 2004, 15 December.

[21] “Download Java 2 Platform, Standard Edition, v 1.4.2 (J2SE)”, “http://java.sun.com/j2se/1.4.2/download.html”, last accessed at 2004, 15 December

[22] “Overview (Java 2 Platform SE v1.4.2)”, “http://java.sun.com/j2se/1.4.2/docs/api/”, last accessed at 2004, 15 December

[23] Delany M. “Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys)”, August 2004.

[24] Wong M W, Lentczner M. “The SPF Record Format and Test Protocol draft-ietf-marid-protocol-00”, 2004, July 11.

[25] Lentczner M, Wong M W. “Sender Policy Framework (SPF) A Convention to Describe Hosts Authorized to Send SMTP Traffic”, May 2004.

[26] Kaliski B S Jr. “A Layman’s Guide to a Subset of ASN.1, BER and DER An RSA Laboratories Technical Note”, 1993, 1 November.

[27] FIPS PUB 46-3 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION U.S. DEPARTMENT OF COMMERCE / National Institute of Standards and Technology. “DATA ENCRYPTION STANDARD (DES)”, reaffirmed 1999 October 25.

[28] Federal Information Processing Standards Publication 197. “ADVANCED ENCRYPTION STANDARD (AES)”, 2001 November 26.

[29] Rivest R. Network Working Group Request for Comments: 2268 “A Description of the RC2(r) Encryption Algorithm”, March 1998.

[30] Rivest R. Network Working Group Request for Comments: 1321 "The MD5 Message-Digest Algorithm", April 1992.

[31] Eastlake D, Jones P. Network Working Group Request for Comments: 3174 "US Secure Hash Algorithm 1 (SHA1)", September 2001.

[32] "The Jakarta Site – Binary Downloads". "http://jakarta.apache.org/site/binindex.cgi", last accessed at 2004, 16 December.

[33] "JavaServer PagesTM v1.1 Syntax Reference". "http://java.sun.com/products/jsp/syntax/1.1/syntaxref11.html", last accessed at 2004, 16 December.

[34] "The Slackware Linux Project". "http://www.slackware.org", last accessed at 2004, 27 December.

[35] "Sendmail – 8.13". "http://www.sendmail.org/8.13.1.html", last accessed at 2004, 27 December.

[36] “GNUMailutils”. “http://www.gnu.org/mailutils/mailutils.html”, last accessed at 2004, 27 December.