SECURE MAIL GATEWAY

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

BY

GÜVEN ORKUN TANIK

IN PARTIAL FULLFILMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2010

Approval of the thesis:

SECURE MAIL GATEWAY

submitted by **GÜVEN ORKUN TANIK** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen ______________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı ______________________
Head of Department, **Computer Engineering**

Dr. Attila Özgit ______________________
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Müslim Bozyiğit ______________________
Computer Engineering Dept., METU

Dr. Attila Özgit ______________________
Computer Engineering Dept., METU

Assoc. Prof. Dr. Özgür Barış Akan ______________________
Electrics&Electronics Engineering Dept., METU

Dr. Onur Tolga Şehitoğlu ______________________
Computer Engineering Dept., METU

Dr. Cevat Şener ______________________
Computer Engineering Dept., METU

Date: 04.05.2010

ABSTRACT

SECURE MAIL GATEWAY

Tanık, Güven Orkun

M.Sc., Department of Computer Engineering

Supervisor: Dr. Attila Özgit

May 2010, 82 Pages

Past few decades witnessed the birth and explosive growth of the internet and the communications –most notably the electronic mail- through it. Studies indicate that although security measures are deployed, their usage is very limited and problematic. This work proposes a system to address some of the shortcomings of present mail security systems, such as underutilization, complicated key management and unwanted immunization against filtering-scanning. A two-layer message encryption scheme with domain level keys is introduced and a performance analysis is presented. An analysis of the improvement on the key management is also presented. Results show that, e-mails can be secured without significant performance impact, and visible ease of key management, by using the system proposed in this thesis.

Keywords: E-mail, Cryptography, Security, PKI, Domain-level

ÖZ

GÜVENLİ POSTA GEÇİDİ

Tanık, Güven Orkun

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Dr. Attila Özgit

Mayıs 2010, 82 sayfa

Geçtiğimiz birkaç onyıl internetin ve internet üzerinde yapılan iletişimin –özellikle elektronik postanın- doğuşuna ve patlamasına tanıklık etti. Ancak, çalışmaların gösterdiği üzere, güvenlik araçları bulunsa bile bunların kullanımı nadir ve problemli olmaktadır. Bu çalışma elektronik posta güvenliği sistemlerinin bazı eksikliklerini – eksik kullanım, karmaşık anahtar yönetimi ve içeriğin filtrelemeye karşı istenmeden bağışıklanması gibi – çözmeyi hedeflemektedir. Bu çalışmada iki katmanlı bir kriptografi sistemi ile internet-alanı seviyesinde anahtarlar önerilmekte ve bu sistemin performans analizleri verilmektedir. Anahtar yönetimini geliştirmek ve kolaylaştırmak üzerine analizler de verilmektedir. Sonuçlar, performanstan önemli bir ödün vermeden ve anahtar yönetimini de kolaylaştırarak, elektronik postaların korunmasının mümkün olduğunu göstermektedir.

Anahtar Kelimeler: E-posta, Kriptografi, Güvenlik, Açık-Anahtar Altyapısı, İnternet Alanı seviyesi

ACKNOWLEDGEMENTS

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

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

While we, the world, came to depend on electronic mail (e-mail) heavily, we do so without using a significant e-mail security infrastructure. This work discusses e-mail security, offering a different solution; different than the traditional methods.

Perhaps, the electronic mail is the most popularly used system for exchanging information over the Internet or any other computer network. A study conducted by the Radicati Group estimates the number of e-mails sent each day to be around 210 billion in 2008. The number of Internet users is estimated to have reached and passed a quarter of the world's population, and almost every user has an e-mail account. From e-commerce to authentication to casual communications; for all kinds of electronic transactions, for receiving and using personal information, the uses of electronic mail are so diverse and critical.

Personal, financial and commercial, all kinds of confidential information are being transmitted in the e-mails. So safety should be a big concern, and many methods are present to secure the e-mail transactions. This however, does not mean e-mail transactions are secure.

Even if the means to protect the e-mail transactions exist, it is also possible that those means are the least utilized security features. That is partly because of the administrative, management and performance costs those security means impose both to the end users and to the administrators, and partly because of the human factors. This work will address both of those cases to increase the usage and thus security of the e-mail transactions.

Most basically, e-mail software can be divided into two categories: mail servers, which deliver, forward and store mail; and mail clients which interface with the users, allowing them to read, compose, store and send mails. Since the networking and computing technologies of the e-mail systems is so ubiquitous, it is well understood and often targeted by attackers, who develop attacks and find exploits against these systems. Mail servers are targeted to gain access to the servers and the network it is connected to, or to destroy its availability. Clients are targeted to insert malicious software to machines and to also propagate the malicious code easily to other machines. Also both are targeted to intercept, snoop on and change the contents of the communication, the mail in this case. So, this means e-mail and its systems are targeted occasionally.

We should also take note that, we are armed better than ever, against such attacks nowadays. Cryptography has never been this available, in this quality to casual users. Monitoring and reporting tools have never been this abundant. Then there are anti-virus, firewall and filtering technologies readily available. Also, helping technologies such as key distribution, routing and digest algorithms are publicly known. It is possible to utilize those technologies to protect the e-mails, in a coherent and secure system, and also make it such that they are easy to utilize.

The Internet is expanding. The number of Internet users is in the order of billions. And almost every Internet user uses electronic mail. Factoring in the spam mails, which constitute more than that of the legitimate mails, this means huge electronic mail traffic, which is mostly unprotected. Applications such as *"domainkeys"* may have helped mitigate the spam problem, but more problems lie in wait as we use electronic mail for all kinds of transactions. The security systems today should be capable of handling many e-mails a day, at the same time filtering out the unwanted mail out of them.

This work attempts to tackle these problems and demonstrates it is possible to build a secure system that increases encryption use, simplifies key management, allows frequent key changes, and still be able to provide filtering for the mail with minimal performance impact and end-to-end encryption.

This work introduces two different key sets for each mail transaction. The first set is the domain level key set, where the public key of the set is used for encrypting any mail that is intended for any user of the domain. A single key is used for all the domain's users. Mail is encrypted with this key for most of its life cycle, until it reaches the gateway to be downloaded by the user. The second set is the personal key set, which is utilized at this point, and is used to encrypt the mail on its way from the gateway to the user. This model alone makes the frequent key changes and filtering of the mail at the gateway possible.

From the sender's perspective, this model makes the mail server something like a proxy for the actual user the mail is intended. The mail is encrypted for the mail server to decrypt, not the user, using a single key for all users of a domain. The transfer of mail to the end user is still encrypted, encrypted by the server with the user's personal key, which is created by the user's log on procedure. This not only makes the frequent change of user keys so much easier, and also lets the server to establish a baseline for key strength, for mails meant for its users, ultimately increasing the security. Also the proposed system would eliminate the need for third party involvement; using phonebook like directory systems for key sharing is a possibility with this system, if the system designers opt for it.

Key management is made a mostly intra-domain issue with this model. And since all the domains use a user authentication method, the key management (and distribution) may be piggybacked to those methods, such that users would not even notice it is there. A side effect would be the decrease of the number of the keys on the Internet, as only the domains' keys are needed in inter-domain communications.

The proposed system uses the existing mail infrastructure, with minimal software add-ons. The proposed system only needs to be installed on the server, and is operational after an Internet address configuration. Thus, with minimal impact to the current mail software both on the clients and servers, this scheme is applicable.

A proof of concept implementation will be used to demonstrate all the claims are met, and performance measurements will be presented in the oncoming chapters.

The outline of this thesis is as follows:

The first chapter is a brief introduction to the current state-of-the-art at the time of writing of this thesis. The need for and the importance of security are emphasized and potential hazards are identified. Then a proposal for mitigating and solving those problems is offered.

Chapter 2 mentions the necessary background technologies that the secure mail gateway is founded upon. Starting with the basics, mail protocols are mentioned, including some brief history, followed by access standards and current access protocols. An introduction to cryptography, message signing and message digest systems are given, along with detailed description of some algorithms used, including the secure remote password (SRP) algorithm which is used directly in this thesis. This chapter ends with the discussion of the current mail security schemes, giving brief introduction to each.

Chapter 3, starts with stating the issues with the current systems, continues to set objectives and requirements for a solution to those issues presented. Then an architecture is proposed and its operation is detailed. The implementation of the proof of concept of the proposed solution is detailed afterwards and finally a comparison of the proposed solution, the secure mail gateway, with the current systems is given.

In chapter 4, the results and performance simulations are given for the proof of concept, and its possible impacts on the system is discussed briefly.

Finally, chapter 5 wraps up the thesis by stating the conclusions, giving a brief summary of the points made in the thesis, and identifying some possible future works.

CHAPTER II

THE TECHNOLOGY

Before going into detail about the secure mail gateway, we need to understand the technologies that are the foundation of this system. This chapter discusses the underlying technology of this study, other related technologies, and their brief history, their evolution. Evolution of mail and its underlying protocols will be discussed first, followed by information security technologies, including cryptography, message digest algorithms, digital signatures, digital envelopes and certificates. Key distribution algorithms will be discussed next followed by a brief discussion of current security systems, mainly the ones that inspired the design of this project.

2.1 Electronic Mail

Electronic mail (e-mail) has been around for many decades. It was not always like what it is today. First systems were simply consisted of file transfer protocols. These systems were extremely hardware (and system) dependant, and interoperability was a big issue. Experience led to better e-mail systems, and in 1982, the ARPANET e-mail proposals were published as request for comment (RFC) 821 (transmission protocol) [3] and RFC 822 (message format) [4] that became the de facto internet standards (Minor revisions came later: RFC 2821 & RFC 2822 [42,43]). Two years later International Telegraph and Telephone Consultative Committee (CCITT) drafted its X.400 recommendation. After a decade of competition, RFC 822 based systems were being widely used, whereas systems based on X.400 had disappeared [2]. We are still using RFC 822 based systems, and the rest of the discussion takes only those systems into account.

2.2 Mail Transport Standards

2.2.1 Simple Mail Transfer Protocol (SMTP)

This is the protocol that is the foundation of the e-mail systems used today, and is the integral protocol in this project. It will be explained in detail here. SMTP is described in RFC 821 (transmission protocol) and RFC 822 (message format). RFCs 2821 and 2822 replaced RFCs 821 and 822 later.

The aim of the simple mail transfer protocol (SMTP) is to transfer mail reliably and efficiently. SMTP is independent of the particular transmission subsystem and requires only a reliable ordered data stream channel. An important feature of SMTP is its capability to relay mail across different transport service environments [3]. Normally mail architecture consists of two subsystems: mail user agents (MUAs) and mail transfer agents (MTAs). Mail user agents are typically local programs whereas mail transfer agents are typically system daemons.

SMTP model can be summarized as follows; the MUA with an e-mail to send establishes connection to a mail server (MTA). MUA then transfers the mail to the MTA. The MTA may be the final destination, or an intermediate relay in which case the MTA acts as a client to another MTA. This may go on until the mail reaches its final destination. Each MTA is obligated to insert trace information, and not change the mail otherwise. The job of each agent (MUA or MTA) is to deliver the e-mail or to report failures [2].

The delivery of each message is a transaction. If we take a closer look to e-mail transactions, we would see it consists of a series of commands and replies. Once the transmission channel is established, the SMTP-sender sends a mail command indicating the sender of the mail [3]. If the SMTP-receiver can accept the mail, it responds with an OK reply. The SMTP-sender then sends a RCPT command identifying a recipient of the mail. If the SMTP-receiver can accept mail for that recipient it responds with an OK reply; if not, it responds with a reply rejecting that recipient (but not the whole mail transaction). The SMTP-sender and SMTP-receiver may negotiate several recipients. When the recipients have been negotiated the SMTP-sender sends the mail data, terminating with a special sequence. If the SMTP-receiver successfully processes the mail data it responds with an OK reply. The dialog is purposely lock-step, one-at-a-time [3].

Messages consist of a primitive envelope, some number of header fields, a blank line, and then the message body [1]. Logically, each header field consists of a single ASCII text containing the field name, a colon, and for most fields, a value [2]. The message format, using RFC 822, does not have a clear distinction between envelope fields and header fields. But in normal usage, the MUA builds a message, passes it to MTA, which then uses some of the header fields to construct the actual envelope [2]. Then like a traditional mail system, the MTA sends the mail to its destination using the envelope's data.

2.2.2 SMTP Extensions

As the number of email users grew, additional functionality was sought in mail clients and SMTP servers. For SMTP servers to support this additional functionality, extensions were added to SMTP. In 1993, RFC 1425 introduced the concept of SMTP service extensions. Subsequently, RFC 1425 was superseded by

RFC 1651 in 1994, RFC 1869 in 1995, and RFC 2821 in 2001. These RFCs added three pieces to the SMTP framework:

- New SMTP commands (RFC 1425)
- Registry for SMTP service extensions (RFC 1651)
- Additional parameters for SMTP MAIL FROM and RCPT TO commands (RFC 1869).

Not all SMTP servers supported (and still does not support) extensions. So there was a need for a method to allow the mail client application to determine whether the server supported extensions. This was accomplished through the *"enhanced hello"* (EHLO) command. When connecting to a SMTP server, a mail client could issue the EHLO command. If the server supported SMTP extensions, it would give a successful response and list the extensions that were supported. Many SMTP servers support a variety of extensions, but they need not be consistent, some may support many extensions, while some may support as much as one. If the server did not support SMTP extensions, it would issue a command failure response prompting the MUA to respond with the standard HELO command. SMTP servers that support SMTP extensions, also known as Extended SMTP (ESMTP), typically respond with ESMTP in their banner [26].

2.2.3 Multipurpose Internet Mail Extensions

In the early days of ARPANET, e-mail was mostly used in academia, and it consisted exclusively of text messages written in English, expressed in ASCII. Messages containing not only text, and messages in other languages (different accents or alphabets) were a problem. RFC 822 was not enough for this kind of problems. RFC 1341 proposed a solution for that (updated in RFCs 2045-2049): Multipurpose Internet Mail Extensions [1].

Basically MIME continues to use the RFC 822 format, but adds structure to the message body and defines encoding rules for non-ASCII messages [2]. RFC 822 (later obsoleted by RFC 2822 [43]) defines a message representation protocol that specifies message header with a lot of detail, but leaves the message content (body) as flat ASCII text [5].

The Multipurpose Internet Mail Extensions (MIME) was proposed to provide alternative ways to describe the structure of rich message content, making use of the headers in an RFC 2822 message. MIME uses the convention of content-type/subtype pairs to specify the native representation or encoding of associated data. Examples of content types include the following:

- Audio – for transmitting audio or voice data.
- Application – used to transmit application data or binary data.
- Image – for transmitting still image (picture) data.
- Message – for encapsulating another mail message.
- Multipart – used to combine several message body parts, possibly of differing types of data, into a single message.
- Text – used to represent textual information in a number of character sets and formatted text description languages in a standardized manner.
- Video – for transmitting video or moving image data, possibly with audio as part of the composite video data format.

The current MIME standards include five parts: RFCs 2045, 2046, 2047, 4289 (which replaced 2048), and 2049. They address message body format, media types, non-American Standard Code for Information Interchange (non-ASCII) message header extensions, registration procedures, and conformance criteria, respectively. With this added functionality, email features such as message attachments and inline hypertext markup language (HTML) are possible [26].

2.2.4 Proprietary Mail Transports

As mentioned previously, some messaging systems use MTAs that do not support either SMTP or ESMTP. These types of MTAs are designed to work within a closed messaging environment. Many large-scale government, academic, and private organizations have messaging systems that use these types of MTAs. However, these organizations still rely on SMTP or ESMTP-capable MTAs for communicating with external messaging systems. Microsoft exchange is an example.

When we are talking about mail protocols, we should also address the mail retrieval protocols for a more complete understanding. So here they are:

2.3 Client Access Standards

After MTA delivers the mails to their final destination, users need to access the mails. There exist several methods for that.

The simplest of all is the direct access to the mailbox. Many Unix hosts support this method. Using command based mail programs, users can access their mailboxes that exist in their home directories. It may look straightforward, but it requires each user having an account and a command line interface to the host system.

This practice however, happens to be a major security risk, especially considering external users. To prevent such a practice, mailbox access protocols were devised. We will cover the most ubiquitous client access standard.

2.3.1 Post Office Protocol – Version 3 (POP3)

POP is the oldest, and maybe the best known Internet mailbox access method. It was developed in 1984, when it was nothing more than a method to copy the mail to the mailbox of the user. It has undergone several improvements and now it is in its third iteration now.

There are three remote mailbox access paradigms: offline, online and disconnected. Of these three, POP is designed to support the "offline" mailbox access paradigm. In this paradigm, mail is delivered to a server, and the user periodically invokes a client program to connect to the POP server and download all the pending mail to the user's machine [6].

POP is useful for computers that do not have sufficient resources for connectivity and/or running an online client [7]. POP is not intended for extensive mail manipulations on the servers. Normal operation means mail is downloaded to the client machine and deleted from the server. Although POP supports the ability to download a specific set of messages and leave them on the server, most e-mail programs just download everything and empty the mailbox [2].

For some clients, this of course, is not enough.

Some of the issues of POP is, it makes the user solely responsible of maintaining message archives, which is not acceptable for some institutions and practices, as well as usage of several workstations make the mail archives dispersed. So other message access protocols are needed.

2.3.2 Internet Message Access Protocol – Version 4 (IMAP4)

When it was first introduced, in 1988, IMAP did not provide much more capabilities over POP. It was a functional superset of POP, and was introduced to allow for multiple MUAs to access a central mailbox. Now, it is in its fourth iteration and allows much more comprehensive functionalities [37].

IMAP is an internet-based message access protocol [8] designed for three paradigms of remote access, offline, online and disconnected [9].

In offline paradigm, mail is delivered to a server and the client user connects to the server to download the pending mails. After the download process is finished, the messages are deleted from the server and are not available for another client to access.

In online paradigm, mail is delivered to a server, but the client user does not download and deleted the messages from the server. Messages can stay in the server until explicitly removed by the user. User can manipulate the remote mailbox as if it is local.

In disconnected paradigm, a copy of the mail is downloaded to the mail client after which the client disconnects from the server [10].

In offline mode, IMAP minimizes the connection time and server resources. This is useful in environments where client machines are resource-rich and servers are resource-poor.

In online mode, IMAP provides the ability to fetch the structure of a message without downloading it, to selectively fetch individual message parts, and the ability to use the server for searching in order to minimize data transfer between client and server, e.g. A short text message with a few megabytes of video clip attached [6].

In the disconnected mode, the primary copies of all messages reside on the server. The client connects to the server, downloads the mails and "caches" them on the client side. During the next connection (and any subsequent connections), the client resynchronizes with the server [11].

2.3.3 Proprietary Mailbox Access Mechanisms

Proprietary mailbox access protocols are designed to work in closed environments. They usually support all standard protocols, as well as providing additional functionality when their clients are used. So, they can interoperate with all other MTAs or MUAs, but they are usually used in their own environments. Microsoft exchange is a good example.

2.3.4 Web-Based Mail Access

Webmail applications are being used more and more as "the" mail service, in delivery as well as access. Simply, a web browser is what is needed to access the mailbox. A user simply uses the web browser to connect to the website that hosts the web-based mail application. The connections are usually over HTTP (hyper text transfer protocol) or HTTPS (HTTP over Transport Layer Security, encrypting the transmission). It should be noted that usage of HTTP should be done with caution, as it does not offer any security or protection against attacks. Standard mailbox access protocols are used between the web servers and mail servers. It is not used between web servers and web browsers.

2.4 Cryptography

Cryptography is the study of “mathematical” systems involving two kinds of security problems: privacy and authentication. Privacy system prevents the extraction of information by unauthorized parties from messages transmitted over a public channel, assuring the sender of the message that it is being read only by the intended recipient. Authentication system prevents the unauthorized injection of messages into a public channel, assuring the receiver of a message of the legitimacy of its sender [12].

Cryptography has been around long before any computers existed (using rods, slide rules, or simple word replacement, as in Caesar cipher) [13], but the computer is the instrument that made cryptography this available, in this quality to casual users. At the same time, however, this era of computers brought new communication and computational capabilities, have given rise to new cryptographic problems. Now, the Internet poses a great challenge to cryptographic systems.

In our case, the e-mail’s native environment is the Internet, the biggest network there is, which can be viewed as a giant public channel. Thus we need to understand and utilize state of the art cryptography in our systems, just in order to preserve the security quality in our transactions, preserve privacy and authentication.

Cryptographic techniques are a great tool to prevent most of the attacks on the mail systems. However, it is also known that cryptographic systems can bring a computational burden on mail systems. Also most current systems need a third party to ensure the security of the e-mail transaction.

Following sub-sections take a broad look at the various cryptographic techniques available.

2.4.1 Symmetric Key Cryptography

This class of encryption algorithms uses the same key for encryption and decryption. In particular, block ciphers, which take n-bit blocks of plaintext and transform it into n-bit blocks of ciphertext, will be discussed here.

These ciphers use substitution and transpositions as basic elements, cascaded to produce a complex system to take in the key and the plaintext and scramble the plaintext into the ciphertext. As expected, these ciphers are reversible when the key is known. The plaintext can be produced, given the key and the ciphertext.

Substitution ciphers change each letter, or group of letters with another letter or group of letters, effectively preserving the order of the plaintext symbols, but disguising them. Transposition ciphers only reorder the letters, but do not disguise them. When used in combination, these basic elements can produce extremely complex functions of the plaintext [2].

It should be noted, though, Kerchoff's principle states that the secrecy (privacy) of the communications solely depends on secrecy of its keys [14]. So the composition of the basic elements, namely substitution and transposition, should not be secret. It is the key, or more precisely the length of it, is critical in determining the strength of a cryptographic system. The composition of the basic elements only aids in the security effort.

Many symmetric key systems are developed, some well known ones are: Data encryption standard (DES) [15], Triple DES [15], advanced encryption standard (AES- the Rijndael) [16] and RC5 [17].

2.4.2 Asymmetric Key Cryptography

The key is the primary security factor in cryptography. No matter how strong the cryptosystem, if the key is compromised in any way, it is worthless. So the inherent problem in symmetric key algorithms (also the whole cryptosystems) is key distribution. Keys need to be protected, but they need to be distributed to the users too.

In 1976, a new and radically different cryptosystem was proposed by Diffie and Hellman [18]. This system had different keys for encryption and decryption. And the decryption key could be feasibly derived from the encryption key. Also the system, namely the encryption key was to be resistant to chosen plaintext attacks. Since the encryption key (public key) for the party that wants to receive secret messages is planned to be made public to everyone. Having the decryption key (private key), only the intended recipient can decode the secret messages. When each user has two keys, a public key and private key, all communication can be encrypted safely.

The only catch is to find such algorithms.

There exist a few schemes, depending on the difficulty of factoring large numbers or computing discrete logarithms modulo a large prime number or on elliptic curves. But the two major categories are the large number factoring and discrete logarithms modulo a large prime.

Maybe the most widely known such algorithm is the RSA (Rivest, Shamir, Adleman, the initials of its three discoverers) [19]. It survived more than a quarter of a century of attempts to break it, and is considered very strong. Its only major disadvantage is the size of its key; it requires at least 1024 bits for good security, whereas 128 bits may be enough for symmetric key algorithms [2]. The size of the key makes it quite slow.

Giving a brief summarization of the algorithm:

- Choose two large primes, p and q.
- Compute n = p x q and z = (p-1)x(q-1).
- Choose a number relatively prime to z and call it d.
- Find e such that e x d = 1 mod z.

Dividing the plaintext into blocks smaller than n, each block is encrypted/decrypted using the power to the e/d modulo n formula.

The security of the method depends on the difficulty of factoring the publicly known n, finding p, q and z, since e is known d can also be figured. It may seem that as the computers increase in power, the algorithm may become obsolete, but even then just by choosing p and q larger, the security can be maintained [2].

2.5 Digital Signatures

The public key encryption does not solve all the problems of secure communication. A major problem in public key and other cryptographic systems is the need to confirm that the sender of the received message is actually the person/entity named in the message.

So, we need some more qualities for secure communications:

- Receiver should be able to verify the claimed identity of the sender.
- The sender can not be able to later repudiate the contents of the message.
- The receiver can not possibly produce the message itself.

These qualities guarantee that no party can deny singing the message later; this is called nonrepudiation [2]. The “digital signature” technique is used to realize these qualities.

There are different approaches to solve this problem. Among them are symmetric key signatures and public key signatures.

Symmetric key signature scheme involves each party trusting a central authority, carrying their secret keys to the authority. Then each sends the message to the central authority, using the central authority as a relay. Each message should contain a random number and a timestamp to avoid replay attacks [2].

The public key signature scheme involves a different approach. For this approach, the public key encryption and decryption algorithms need to have the property that the plaintext encrypted by the decryption key can be decrypted using the encryption key. The scheme works like this: A user who has a publicly accessible public key can “digitally sign” a message by decrypting the message with its private key before sending the message. The recipients of the message can verify the message or the signature by encrypting the message using the sender’s public key. Thus, the “digital signature” process is essentially the inverse of the typical cryptographic process, that the message is decrypted first, and then encrypted. Anyone who has the sender’s public key can read the message, but only the sender could have created it. Still serious problems exist in this scheme. The sender later can change its keys, denying sending the message later, or publicize its private key, claiming it can be sent by anyone. Or more simply, the owner of the keys may not be the person/entity he claims to be. The technique mostly used to overcome these problems is to rely on a trusted authority.

The trust scheme works like this: The trusted authority creates a digital message which contains the claimant's public key, and the name of the claimant (which is accurate to the authority's satisfaction), and a representative of the authority signs the digital message with the authority's own digital signature. Any recipient of the claimant's message can trust the signature, provided that the recipient recognizes the authority's public key, to the extent that the recipient trusts the authority [20].

In principle, any public-key algorithm can be used for digital signatures. The de facto industry standard is the RSA algorithm, but many other algorithms exist.

2.6 Message Digests

Message digests or cryptographic hash functions as they are sometimes called, have many information security applications. They may be used as checksums to detect corruption, fingerprints to uniquely identify files or ordinary hash functions. But the use we are more interested is in authentication. They can be used in digital signatures, or in message authentication codes. Digital signatures provide authentication and secrecy, but sometimes only authentication may be needed, hence the use of message authentication codes.

Message digest algorithms take a variable length of input, and give a fixed length output.

Message digests should have four properties:

- Given a message, it should be easy to compute its hash value
- It should be infeasible to find a message corresponding to a given hash value
- It should be infeasible to find two messages with the same hash
- It should be infeasible to change a message without changing it hash value; even 1 bit of change in a message should change the hash value thoroughly.

Computing a message digest from plaintext is much faster than encrypting the plaintext with a public key algorithm. So only the digest may be signed, instead of the whole message to speed up to signing process.

Most widely used message digest algorithms are the SHA [21,23] and MD [22] variants.

2.7 Key Exchange

Key exchange protocols are the mechanisms by which two parties communicating over a public channel can generate a common secret key. They are essential to almost all our private communications over insecure networks (i.e. the internet). They are maybe the most commonly used cryptographic protocols. Examples include SSL, IPSec and SSH, which many other applications depend on [25].

There are many methods for key exchange. The public key approach is one of them. But the revolutionary method, which is the foundation to many of the key exchange algorithms, is the Diffie-Hellman key exchange algorithm [18]. There are many variants of Diffie-Hellman key exchange algorithm; the original depending on discrete logarithm problem, or depending on elliptical curves, or one that depends on a pre-shared key.

We will look closely to the Diffie-Hellman algorithm, and the one variant that is used in the building of secure mail gateway, secure remote password (SRP) method.

2.7.1 The Diffie-Hellman Algorithm

The Diffie-Hellman algorithm [36] can be summarized as follows:

The protocol has two system parameters p and g. They are both public and may be used by all the users in a system. Parameter p is a prime number and parameter g (usually called a generator) is an integer less than p, with the following property: for every number n between 1 and p-1 inclusive, there is a power k of g such that n = g^k mod p.

Suppose Alice and Bob want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They proceed as follows: First, Alice generates a random private value a and Bob generates a random private value b. Both a and b are drawn from the set of integers. Then they derive their public values using parameters p and g and their private values. Alice's public value is g^a mod p and Bob's public value is g^b mod p. They then exchange their public values. Finally, Alice computes $g^{ab} = (g^b)^a$ mod p, and Bob computes $g^{ba} = (g^a)^b$ mod p. Since $g^{ab} = g^{ba} = k$, Alice and Bob now have a shared secret key k.

The protocol depends on the discrete logarithm problem for its security. It assumes that it is computationally infeasible to calculate the shared secret key k = g^{ab} mod p given the two public values g^a mod p and g^b mod p when the prime p is sufficiently large. Maurer [24] has shown that breaking the Diffie-Hellman protocol is equivalent to computing discrete logarithms under certain assumptions.

There are many variants of the Diffie-Hellman algorithm, but we will focus on one, which is directly utilized in the building of the secure mail gateway.

2.7.2 The Secure Remote Password (SRP) Algorithm

Even though cryptographically very strong, Diffie-Helmann is still vulnerable to man-in-the-middle attacks. A variant, known as the secure remote password (SRP) protocol uses a pre-defined password to nullify such a vulnerability [38, 39, 40].

It can be summarized as follows:

As defined in [27]:

Steve and Carol are to establish a secure communication line. As the example before, p and g are public. First, a password needs to be established. To establish a password *PW* with Steve, Carol picks a random salt *s*, and computes

$$x = H(s, PW)$$

$$v = g^{x}$$

where H() is a one-way hash function. Steve stores Carol's password verifier and salt. For authentication, the protocol happens as follows:

1. Carol sends Steve her username,
2. Steve looks up Carol's password entry and fetches her password verifier *v* and her salt *s*. He sends *s* to Carol. Carol computes her long-term private key *x* using *s* and her real password *P*.
3. Carol generates a random number a, $1 < a < p$, computes her ephemeral public key $A = g^{a}$, and sends it to Steve.
4. Steve generates his own random number b, $1 < b < p$, computes his ephemeral public key $B = v + g^{b}$, and sends it back to Carol, along with the randomly generated parameter u.
5. Carol and Steve compute the common exponential value $S = g^{(ab + bux)}$ using the values available to each of them. If Carol's password *P* entered in Step 2 matches the one she originally used to generate *v*, then both values of *S* will match.

6. Both sides hash the exponential S into a cryptographically strong session key.
7. Carol sends Steve $M1$ as evidence that she has the correct session key. Steve computes $M1$ himself and verifies that it matches what Carol sent him.
8. Steve sends Carol $M2$ as evidence that he also has the correct session key. Carol also verifies $M2$ herself, accepting only if it matches Steve's value.

After the successful completion of the transaction, both sides may use the key generated on the Step 6.

SRP is a way to implement strong authentication methods into secure communication, which until recently used either symmetric key cryptography, public key cryptography, or a combination of both. And also SRP does this without depending on any external infrastructure.

It is also the choice of key exchange in this work.

2.8 Mail Security Schemes in Literature

We sometimes want to protect the confidentiality and integrity of our e-mail messages. There are various ways for this:

- Sign the message to protect integrity and confirm its sender
- Encrypt the body to provide confidentiality
- Encrypt the communication to protect the confidentiality of the header and the body

First two methods are complementary. If the message is encrypted, it is usually signed too, to provide authentication and integrity check. If the mail is signed, it may not be encrypted if the contents' confidentiality is not necessary.

The third method is not widely employed, as it is difficult to establish. It may be used between mail servers of organizations that regularly send mails to each other, in the form of a virtual private network. But still, this can not provide a signing mechanism, or end to end protection.

Most mail security schemes in use protect individual messages. The most widely used mail security methods are the open pretty good privacy (OpenPGP) and Secure Multipurpose Internet Mail Extensions (S/MIME). They both use public key cryptography as their part. Since public key cryptography is computationally intense, it is used sparingly. The much more efficient symmetric key method is heavily used. Often only the symmetric key is encrypted using the public key and added to the encrypted message. The communication is then carried on using the symmetric key.

2.8.1 Open Pretty Good Privacy (OpenPGP)

PGP was created by Philip Zimmermann and first released, in Version 1.0, in 1991. Subsequent versions have been designed and implemented by an all-volunteer collaborative effort under the design guidance of Philip Zimmermann [28]. In 1996, version 5.x of PGP was defined in IETF RFC 1991 [28], PGP Message Exchange Formats. Subsequently, OpenPGP was develop as a new standard protocol based on PGP version 5.x. OpenPGP is defined in RFC 2440 [29], OpenPGP Message Format, and RFC 3156, MIME Security with OpenPGP [30].

OpenPGP uses a symmetric key algorithm for the actual encryption of the message body. OpenPGP also uses public key cryptography, such as digitally signed message digests.

The following is a brief description of signing and encrypting a message with OpenPGP:

- OpenPGP compresses the plaintext, which reduces transmission time and strengthens cryptographic security by obfuscating plaintext patterns commonly searched for during cryptanalysis.
- OpenPGP creates a random session key (in some implementations of OpenPGP, users are required to move their mouse at will within a window to generate random data).
- A digital signature is generated for the message using the sender's private key, and then added to the message.
- The message and signature are encrypted using the session key and a symmetric algorithm (e.g., 3DES, AES).
- The session key is encrypted using the recipient's public key and added to the beginning of the encrypted message.
- The encrypted message is sent to the recipient

The recipient should reverse the steps to obtain the message and to verify the signature [26].

2.8.2 Secure Multipurpose Internet Mail Extensions (S/MIME)

Secure Multipurpose Internet Mail Extensions was first proposed by RSA Data Security, in 1995. It was based on their public key cryptography Standard (PKCS) #7 [25] for data format of encrypted messages and the X.509 version 3 standard [31] for digital certificates.

By its version 2, S/MIME managed to gain industry wide adoption in the Internet mailing. Although it is not a recognized IETF standard, it is specified by RFCs 2311, 2312, 2313, 2314, 2315, and 2268 [26].

S/MIME version 3 was developed by the IETF and adopted as an IETF standard in July 1999. It is defined in RFCs 2631, 2634, 3850, 3851 and 3852.

S/MIME's actual process is very similar to that of the OpenPGP. But its most significant attribute is, because of the heavy industry involvement, it is built-in in nearly all well-known mail clients where as a plug-in is needed for OpenPGP.

2.8.3 Domain Keys Identified Mail (DKIM)

Domain Keys is a message signing infrastructure. It is a domain level authentication framework that allows the signing and verification of the source and the contents of the mails by either MUAs or MTAs. Its first version is a synthesis and enhancement of the Yahoo Domain Keys and Cisco Identified Internet mail.

Domain keys use public key cryptography for its functions, and it allows an organization to take responsibility for transmitting a message, and have it verifiable by the recipient.

The ultimate outcome of this framework is to prevent mail fraud, protecting the mail sender identity, the signer identity, and help take spam and phishing under control, while retaining the e-mails functionality.

The framework basically signs e-mail such that it may be verified by MTAs and/or MUAs. But it is never had encryption as a design goal or as a part of its mechanism. Also the failure of a mail’s verification does not force the rejection of the mail.

CHAPTER III

SECURE MAIL GATEWAY DESIGN AND IMPLEMENTATION

This study set out to demonstrate that, it is possible to build a secure system that increases encryption use, simplifies key management, allows frequent key changes, and still be able to provide filtering for the mail with minimal performance impact and comprehensive encryption using the technologies presented in the previous chapter.

This chapter will start with identifying the problems of the current systems, and then continue with identifying the objectives and requirements of the system. The proposed architecture and the operation algorithms as well as a comparison with other systems will follow.

3.1 Current Issues with Electronic Mail

Although SMTP is a well proven and widely used protocol, it sends mails out in the open. There are a few methods to increase the security, each with their own drawbacks. In all of them, other than having a leased line network, every single one uses cryptography.

Using a virtual private network may be another solution, but even then there is no end-to-end protection, no signing mechanism, and the system is completely open to attacks that originate from inside sources, which happen to be the majority of the hostile attempts at any system.

So, mail encryption is the way to go. But encryption comes at a cost. So, it must be weighted carefully.

Scanning mails for malware and viruses and filtering the content is significantly more complicated when the mails are encrypted. When the content is encrypted, the mail servers or the firewalls have no way of decrypting the mail, thus have no means to act on the contents of the mail. Then the burden of content filtering, malware and virus detection has to be done on the client. Also those systems may not be up to date, or even present on the clients. Even if they are present, no uniform protection can be achieved.

Also encryption and decryption need hardware resources, namely CPU time to be executed. Thus if it is to be done in bulk, such as in mail servers, it is an additional burden, and may force hardware upgrades.

Organization wide, or wider use of encryption requires significant administrative overhead, such as key assignment, distribution, recovery and revocation, key management in general.

Third parties need to be trusted, and the security directly depends on their trust level, which is not acceptable by many organizations. It may also incite additional costs as in certificate fees.

Law enforcement or other investigative parties may have a hard time reviewing the mail traffic. Although it may not be seen as a problem, as privacy of communication is also a right.

The symmetry of encryption can not be guaranteed. An organization may utilize weak encryption methods, which may compromise the whole communication process.

Maybe the most critical of all, the security of communication is left to the users, who usually fail at maintaining the required encryption standards or may refrain from using them against common sense.

3.2 Objectives and Requirements

The main objective of this study is to develop an infrastructure to provide e-mail security to and between the domains it is deployed. Mail should be protected from end-to-end. Security services of confidentiality, non-repudiation, authentication and content-integrity should be provided. Also services such as spam filtering, anti-virus and content checking should be provided by the gateway. Relationships with other gateways may be established, allowing concerted efforts to increase security via information exchange. Also notary information as well as activity logs should be kept at the gateway to be used if needed. The gateway should be transparent to any mail client or mail transfer software, thus it should be compatible with all of them. The system should be easy to deploy, manage and use.

This work is to propose a solution to almost all these issues presented in the previous part, and to add extra features on top of it, providing a versatile solution to e-mail security.

To address the problem properly, the solution must not depend on user input to work, since experience shows that the existence of the current security features and cryptographic techniques does not mean that they are used, even if they are deployed and available in the users' computers.

The system should not require extensive deployment, installment and/or resources. So that it can be easily reachable by masses, with minimal burden on organizations that need electronic mail security.

The solution should be adaptable, so that it may be on the cutting edge of the security technology (cryptography, anti-virus, content filtering, etc.), providing cutting edge security, since an obsolete security is like no security at all. This can be done by a modular design allowing for other products to be implemented into the system.

The solution should be resource conscious, so that it may increase availability of the systems in use, not burdening the existing mail servers, if not helping them.

The solution should support the existing security systems, by log keeping, and notary information retaining.

Finally, the solution should be stand-alone, not needing any third parties and/or other systems, and low cost.

3.3 Proposed Architecture

The server (or mail server), in this context, is referred as Mail Transfer Agent (MTA) which is external to our system (but internal to the domain). The mail server has two distinct parts, SMTP server for outbound connections, and POP3/IMAP4 server for intra-domain connections. These are indicated when the distinction is necessary. The gateway means our system, which can also be considered a MTA, but one that transfers mail only from the server to the user. The client and the client software (MUA) refer to the users and any software (or web application) they may use to create, send or access electronic mail. The agent software means the component deployed by the gateway to the users' computers.

To satisfy the requirements above, the architecture chosen is a gateway to sit in between the mail server (POP3/IMAP4 server) and the clients to provide the security functions (Figure 1). This gateway is invisible to the users and does not require any modification on the mail server software, making it portable and compatible with any system.

The computational overhead may be carried away from the mail server hardware, into the mail gateway's machine if it is so needed, as the gateway does not need to be on the same computer as the mail server. The spam and content filtering and virus checking can be carried away from the mail server also, to the gateway, lowering the burden on the server machine, and also different programs may be used for each of these functionalities, increasing versatility and strengthening the security of the overall system allowing increased customizability, and allowing constant updates on each of these parts.

Locally (intra-domain), the gateway does not need certificates signed by any third parties, eliminating the need for third parties or the need to purchase certificates for users (clients). For inter-domain transactions, however, a different approach should be adopted. It may still be third party certifications, a trust relationship between gateways, or a trusted directory. Third parties can still be eliminated if so desired.

Also with a small client software, which can be made optional, end-to-end protection is obtained with all of the mentioned qualities. Inter-domain security is supplied seamlessly and transparently between the systems that have the secure mail gateway deployed, eliminating most of the burdens of key management, with more and easier access to cryptographic techniques, consequentially increasing not only the theoretical security, but also the actual security.

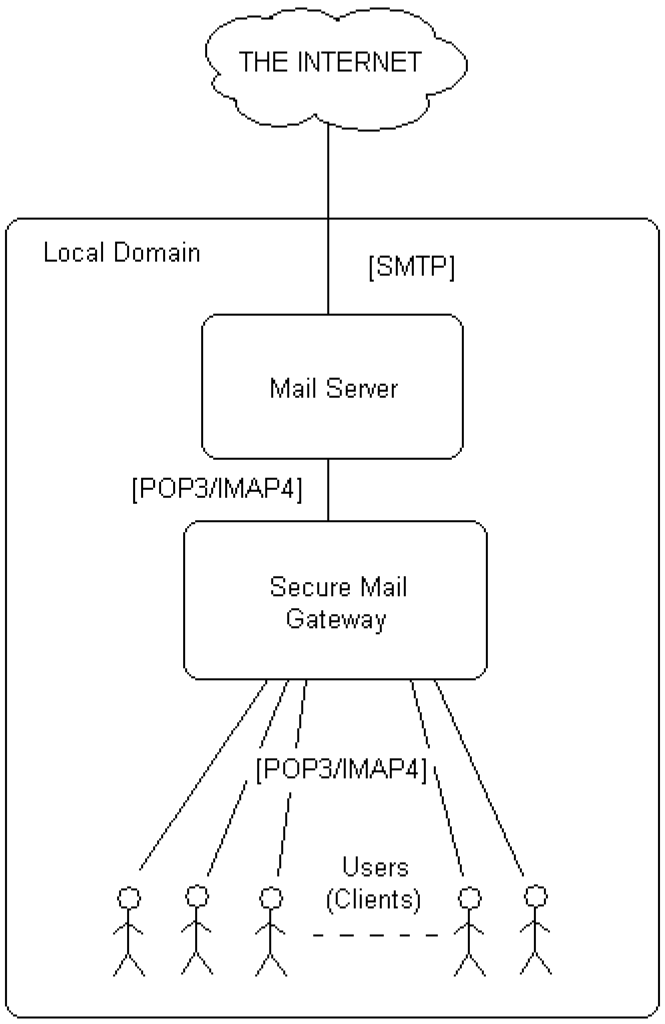

Figure 1 - Gateway Installed Domain

So, how can we increase the availability of cryptographic systems?

By lowering the administrative costs of those systems, thus making the system administrators more likely to deploy and maintain such systems, by adding advantages to their use over other methods, thus making the users want to use such systems and by making them easier to use, which will not make users to resort to easier but less secure ways of communication.

The main idea behind the secure mail gateway is the use of domain specific public keys, instead of individual user keys. This not only drastically decreases the number of keys needed on the internet, but the whole system makes key management automatic, so almost no conscious effort is needed for the increased security. Still keeping the filtering and scanning features is also a plus.

Let’s validate why we need such an approach via examples:

Thinking from a company’s point of view, let’s call it company A; there are thousands of employees, each with their own e-mail addresses.

It may not seem to be a huge security gap to use unencrypted e-mail in the company network to avoid the effort and the cost it entails, but it should be noted that majority of the assailants are in that network too. Adding outside contact into the mix, the complexity and the need for security soars. So a protection mechanism is needed.

Assuming most of those employees are participating in e-mail transactions that need to be secured, each would need a key for encryption. It amounts to thousands of keys to be managed bringing along the effort and cost of their management. The needed effort can be identified as administrative overhead, such as key assignment, distribution, recovery and revocation, key management in general. Then the problem is still not solved, at least completely.

As an example, when the employee A1 needs mail another employee A2, he needs A2's key to do so securely. Then he needs to ask A2 for his key, or find it from a company directory, then register it in his own computer, encrypt the mail, and send it afterwards. Actually, as Sans Institute's "Solving Healthcare's e-Mail Security Problem" [35] points out, the users are very reluctant to use such measures, even if they are readily deployed, and even where the setting is healthcare and the confidentiality and security of the patients are at stake. The effort needed to be expended by the users is considered too much to be done by the users. The users simply choose not to use the deployed systems or security features.

The previous example demonstrates the lack of user diligence to employ the security measures at hand. The same vulnerability may cause even a bigger hazard. The same lack of security measures may cause the user to compromise its key (along with his/her computer), adding to the security holes of the system.

These are just part of the problem. As another example, considering another company, company B, which can be a large company much like company A, that does business with company A. Any number of contacts may be formed by the employees of each company.

We may assume the companies as e-mail domains for this context. It is hard to manage keys and enforce their use in any single domain by itself. But now, we have two domains that need to do this, and they do this independently. The complexity and proposed security risks are much higher now. Also other pitfalls are introduced. Added to the already problematic environment, now, unbalanced use of encryption may compromise the whole communication. Weaker encryption (or no encryption) on one side may fully destroy the security of the transactions.

A central key, for each domain, can be a solution to many of the above problems. With this scheme, the users only need one key that they will use to communicate with the destination domain.

This also simplifies the management of the keys both for the users and system administrators. Administrators need to deal with one key per domain they tend to. Users only need the key for their recipients' domain, which will be one in intra domain communications and two if company B is also involved. As the number of domains a user interacts with will be almost always fewer than that of other users, it is guaranteed to be simpler or at worst at the same complexity. Also a client agent can greatly reduce the complexity for the users to as few as a single click in this project's proof of concept, and maybe to no extra work by the merit of future works on this concept.

So how does this framework provide an end-to-end protection like classical encryption, if the mail is encrypted for the domain? First, only the gateway has the domain private key, so no other party can decrypt the mails encrypted with that domain's public key. Only the gateway can decrypt the mail.

The ease of applicability of the solution comes from the fact that each user in any domain already has a username and password associated with them. So the gateway uses that authentication information to encrypt the mail on the remaining part of the mails' journey.

This brings another benefit for the users; the mails can still be screened for malicious software, filtered by content, checked for viruses or identified as spam and blocked before the users have to deal with them.

The current systems need to associate keys with the address the mail is being sent. So in order to make this framework work, a client agent is needed. On a plus side, this option lets us to include explicit timestamps (besides the ones the encryption algorithms use as seeds) that help with log keeping. With the gateway in place, it can be precisely recorded when a mail is created, and when it reaches its destination, untampered. This can also be used as notary information.

If we take a closer look how this framework is implemented, the main software (gateway) sits between the users and the mail server (POP3/IMAP4 server). The gateway and the mail server form an aggregate mail server (POP3/IMAP4 server) for clients. The secure mail gateway is not associated with any SMTP transactions the mail server participates in.

The mail server needs to be no different than any other mail servers (MTAs) that are encountered in everyday life, it has no special properties. The gateway has absolutely no effect on the mails that pass through the mail server, with a destination other than the domain the gateway is attached to. The concept is shown in Figure 2 – The Aggregate Mail Server.

The mail that is addressed to the users of the domain has to go through the gateway. The gateway is the only part of the mail server that the users interact for receiving mail. It is a relay for the mails that are addressed for the domain's users.

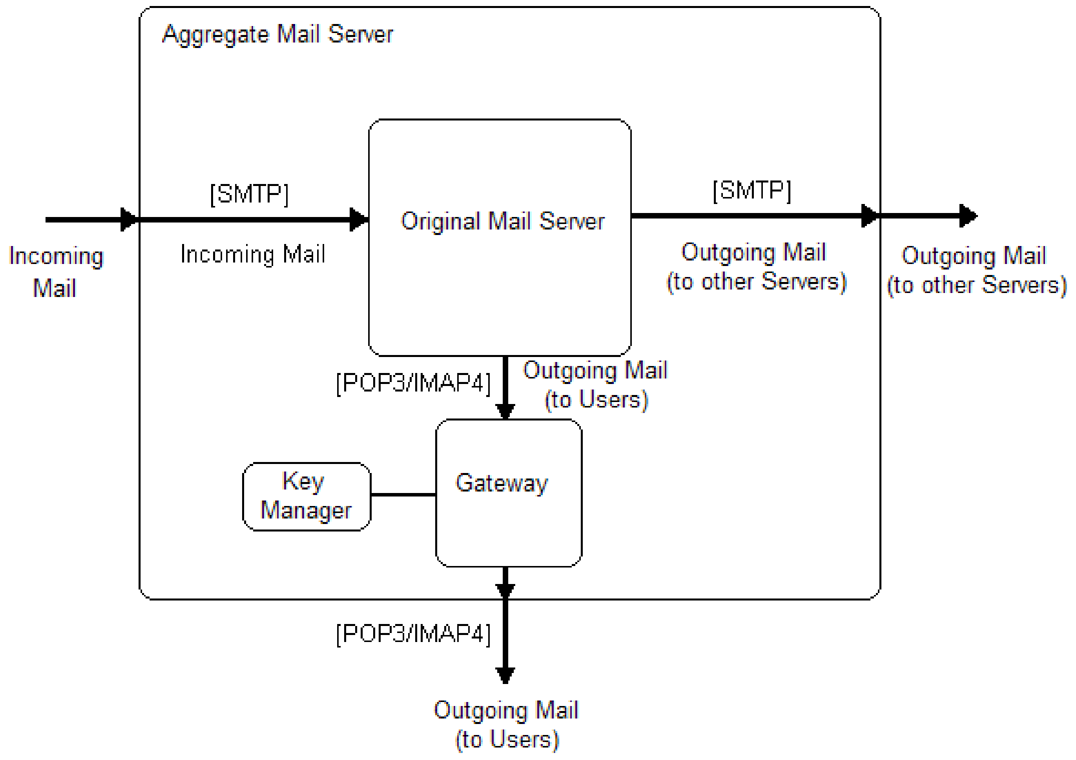

Figure 2 – The Aggregate Mail Server

For sending mail, the original mail server SMTP server) is used. The gateway is not involved in sending the mail. However, the gateway may be involved (supplying the encryption key) in creating the mail to be sent. It should also be noted that the client agent is involved in creating the mail, in the proof of concept.

The journey of a mail after its creation may go through multiple MTAs, whether those servers (SMTP servers) have the gateway installed or not, is not important. Only whether the destination domain has a secure mail gateway or not is of importance. If a domain has the secure mail gateway installed, the mails should go through the gateway regardless of the domain they originated. This allows for the gateway to screen the mail for any harmful or undesired content, which increases the security for the domain.

There are two other actors that help the operation of the secure mail gateway. The first one is the key manager. It handles all things key related, except encryption and decryption. This is the part that handles key creation, assignment, distribution, and validation.

The key manager secures its connections with SRP (using usernames and passwords), and uses this connection to communicate the users' own private and public keys. It is also the party that supplies the destination public keys to the client agents at the time of mail creation.

The other part of this framework is the user agent. The agent provides the users with the capability to encrypt the mail for the domain the recipient(s) of the mail is members of. It also helps by simplifying the encryption procedure that will be apparent in its operation and implementation.

However, it should be noted that, the secure mail gateway system is not a stand-alone mail server (or a POP3/IMAP 4 server); it uses "a" mail server for mail storage and delivery. The gateway itself does not store mails. It only produces and stores key information and notary/logging information. Otherwise it is just a relay that the mails pass through.

3.4 Operation

Let's follow the basic usage of this system explaining the operation concepts on the way. We will go over the journey of a secured e-mail.

As a brief summary of how the system works, the following is an example of how the secure mail gateway may be used as depicted in Figure 3.

It should be noted that the operation is as in the proof of concept, and the numbers in parenthesis are references to the figure.

The user starts the client, puts in his username and password, writes his mail into the client and hits the encrypt button.

Then the client agent connects to the key server using SRP, demands the key to the destination domain (1). Then the key server of the gateway retrieves the destination gateway's public key (2&3 – Note that this part can be done in two different ways; either the key server can directly contact the destination gateway, or look up the destination gateway's key from a central directory). Then the key server relays the key to the client, and client encrypts the mail (4).

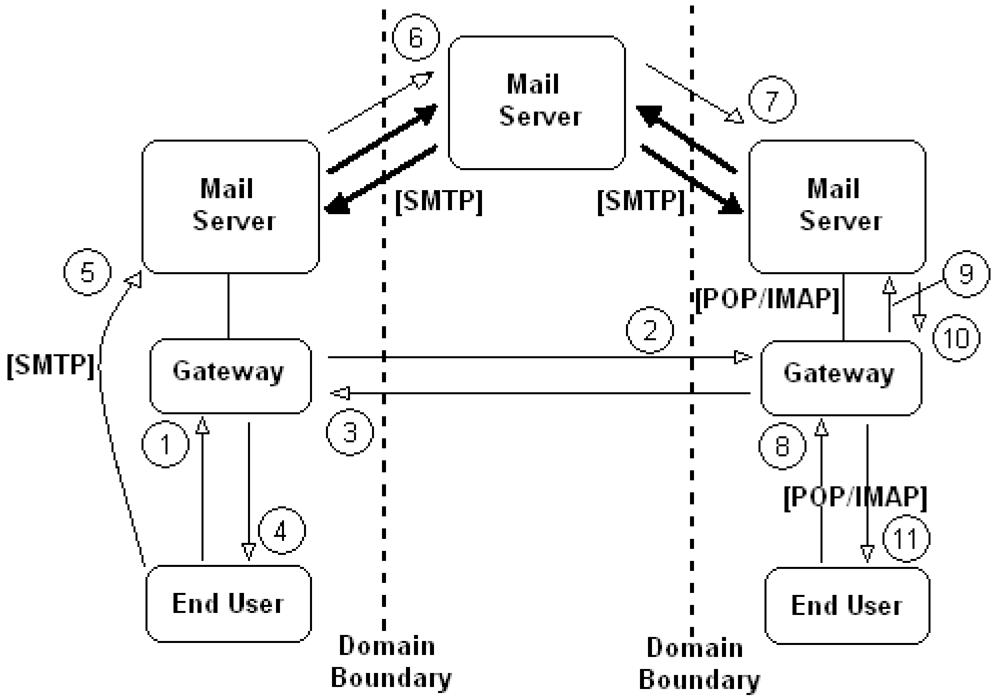

Figure 3 - An Example of Operation

Then the user has two options: either directly presses the send button in the client to send the mail to the mail server (SMTP server), or copy-paste the encrypted body to a mail client (like outlook or any webmail e.g. Gmail) and send it (5).

Then the mail moves through the web from any number of mail transfer agents (SMTP servers) to the destination mail server that stores the mail (6&7). Here the mail is saved until the client requests the mail.

The end-user sees the gateway as its mail server (POP3/IMAP4 server), and makes the request for its mails to the gateway using its username and password (8). The gateway relays the request to the real mail server (POP3/IMAP4 server) (9). The mail server responds with the mail (10). The gateway then decrypts the mail with its own private key, and encrypts it using the user's public key. The mail is then transmitted to the user (11).

Now, let's look into the secure gateway's operation more closely.

3.4.1 Mail Creation

The user should use the client agent to gain the full benefits of this system. Without it, most of the current systems would not allow the user to encrypt his mail using any key but the recipients. So without the client agent, the target domain not being a user (or has a valid mail address), the sender would not be able to encrypt the mail even if he had the domain's current key. It should be noted that, once the destination key is retrieved from the key manager (or it may be already possessed), the client agent is not really needed. Any mail encrypting and sending program that is able to encrypt a mail regardless of the encrypting key belonging to the recipient is usable.

When the client is launched it looks just like a simple mail compiling/sending program. And it can be used for one or both the functionalities.

User puts in its mail, along with his username (mail address) and password (for the mail address) into the client agent just like he was composing a regular mail. Until now, no extra work is done. Only extra work a user needs to do to send a secured mail is the next step. It is just one click, pushing the request key button on the client agent.

When the user clicks request key on the client agent, the client connects to the key manager using SRP. His password never leaves his computer. Only the username is transmitted out in the open, and then, the algorithm produces a symmetric key for communication between the two parties, at the same time providing mutual authentication. (There can also be a certificate provided by the key manager after initial contact by the agent, but it was not found necessary in the proof of concept.)

Then information of the domain(s) the mail is to be sent is transmitted to the key manager, and the key manager supplies the public key(s) to of the domain(s).

Then the client agent indicates which recipients can receive the encrypted message. A timestamp and some text padding are also included in the mail encryption. The encrypted message is displayed in the body of the mail agent.

After this, the user has two options. He can send the mail using the client agent by clicking send, or he may copy/paste the encrypted mail to any third party mailing client, or web mail client, and send from there.

Either way has no advantage over the other. And both ways, the mail is successfully encrypted for any domains their recipients are from. (It may be noted that for multiple domains, copy/paste sending involves multiple such practices, as the mail has multiple encryptions. Alternatively they may all be sent in one body, and the mail gateway would only decode the appropriate one.)

After these steps, the mail is successfully created and sent on its way.

3.4.2 Mail Transmission

After the previous steps, the mail leaves the MUA, and is at the hands of the MTAs (SMTP servers) now.

It will travel through the MTAs until it reaches its destination domain. Even if it is transmitted out in the open in all those transactions, it is encrypted throughout this process and so secured. As it is encrypted, any tampering would be noticed, and even if intercepted, it would not be possible for the assailants to reach the content of the mail.

It should be noted that the destination domain might be no other than the originating domain. It does not make any difference, whether the mail travels through multiple MTAs or just one.

The mail reaches its destination securely encrypted, or a failure report is sent back as any normal mail transaction.

After the mail's arrival to its destination domain, it is stored until its recipient requests for it.

3.4.3 Receiving Mail

The gateway is the portal that the users come to, when they want to access their mails. The gateway poses as the mail server (POP3/IMAP4 server) to the users. The gateway enacts on the mail just once, when it passes through it, from the MTA to MUA.

Until a user requests its mails, the mails are stored in the mail server. This scheme is applicable to all mail systems, whether web based or not.

The users in this system see only the gateway as their mail server for incoming mail. So any request for mails lands to the gateway's ports.

When a user decides to request the mail, he connects to the gateway instead of the mail server (POP3/IMAP4 server) itself.

The mail gateway may be contacted using any protocol desired. For the gateway uses handlers for each protocol and any add-ons may be put in to accommodate any protocols.

The gateway acts like a communication relay until the mail passes through. When the mail is passing through, it is intercepted. The included encrypted body is decrypted, and then re-encrypted for the recipient as if the mail was encrypted for the recipient all along. As only the encrypted part is decrypted, any extra characters, or extra content added by free mail servers, or any signature automatically added by the MUAs used are not issues.

It should be noted that as the gateway has the capability to decrypt the passing mail, it also has the capability to subject it to virus, malware and content checking procedures. Those procedures are built in like add-ons, so that any third party software and/or modules can be used for these functionalities, providing an increase in number of options and security.

As the mail is passed a log is also recorded.

When the client receives the mail, it is trivial to retrieve the contents of the mail, as it is encrypted by his own public key.

This marks the end of the mail’s journey. But it is by no means an exhaustive scenario the mail gateway may be used. There may be other uses of the gateway.

3.4.4 Other Mail Scenarios

The e-mail that will be sent can be sent in other ways than using the client secured from end-to-end.

For mail creation, simply ignoring the facilities the gateway provides is enough. Classical systems can create any kind of mail messages. It is not needed to provide any details to ordinary mail creation here.

As for the mail transportation to the final mail server, the gateway is not involved in any part of it, so the transportation is not needed to be detailed here.

The most common scenario different than the regular scenario depicted earlier is when the gateway users receive mails other than the ones encrypted for the gateway. As the gateway and its users may receive any number of mails from any sources, it is very likely that the mails may be encrypted by different sources (not for the gateway) or not encrypted at all. We have explored how the gateway will behave when the mail is encrypted for itself. Now let’s discuss how the gateway will behave on other scenarios.

If the mail is from a source that did not encrypt the mail the creation and delivery processes are nothing different from ordinary. The mail is created as usual and sent through any number of mail servers (SMTP servers) until it reaches the mail server that has the gateway installed.

When the client connects to the gateway for its mails, the log on and mail requesting procedures are all the same as if it is a secured mail. The only difference is when the mail is passing through; it is noticed as not encrypted. The mail is still subject to virus, malware and content checking, and is still logged before it is passed through. No other steps are necessary.

So, even though the gateway is not used as it is intended, it still provides additional security to the domain but its full security capabilities are not utilized.

If the mail is from a source that did encrypt it, but it was not encrypted for the gateway, instead it is encrypted for the end-user himself, all the procedures for it is the same with an unencrypted mail. It is treated as if it is a plaintext message by the mail gateway, except for the virus, malware or content checking functionalities, which are rendered useless. They are still applied, but its purpose is mostly to catch any mis-configurations of the mail header, or to filter out known malicious content senders.

3.4.5 Key Management

In all the steps that have been mentioned in the previous chapters, there was a component that was mentioned implicitly, the key management.

In this system the key manager is responsible of all the operations that are to do with keys. Creating, storing, transferring and verifying keys are all handled by this part of the system.

We will delve a little deeper on how those functions are handled in this part.

When a client fires up its client agent, there is no knowledge of any keys or passwords in the client's system. The user puts in his username, and his password, along with the target domain (the recipient of the message implicitly gives this information.). A password handler takes in the password, and it is never transmitted. Actually, the password is used to compute some verifiers before the transmission begins and is discarded afterwards (detailed in the SRP algorithm).

When the connection is first established, it is with the key manager. The client agent transmits the username and the derived parameters from the password; never transmits the password itself, any data from which it is possible to replicate the password or its verifiers.

The key manager on the other hand, does not even know the password of the users. A secured key store is used (and protected by a password itself), and the salts and verifiers for passwords indexed by the usernames are used.

When the connection is verified, a common secret is established and verified between the client agent and the key manager (symmetric session key).

This part is not crucial for security purposes itself, as then the information transmitted through that secure channel are only public keys (But assuming the public keys of the users are not used by any entity other than the gateway, security principles dictate no extra information should be supplied and this practice accomplishes this task). But the same mechanism is used to transmit personal private keys to the users when they demand it. Also they are used for any input or editing on the key manager. But the main use here is for the gateway to identify the sender of the mail, and authenticate the sender. Which is crucial to prevent forging mail.

Besides the passwords of the users, the key manager also handles the certificates of the mail gateways.

The primary certificate it is responsible is of its own. The gateway's own certificate, its private key, is stored in a password protected keystore. When the gateway demands it, it is extracted by the key manager and passed t the gateway for decryption to take place.

A crucial detail here is that the gateway should keep its old private keys, indexed by the times they are changed, to ensure that the changes to the gateway keys do not affect the mails that are on transit (or being kept in the mail server encrypted).

Also, the key manager is responsible of supplying a number of public keys. These are of the other secure gateways that the local gateway needs to make a connection. The key manager fetches public keys and passes them to the users that want to encrypt messages to those destinations.

Finally the key manager is responsible of is transmitting its public key to the gateways that demand it. This part can be handled in a number of ways.

The first and simplest is the key manager to publish it itself. This one is very straightforward and easy to implement (and administrate). In this scheme, every gateway is responsible of its own key and its administration. Performance wise, as it mandates multiple connections if the mail is addressed to multiple domains, it can be slower compared to the other approach.

The second approach is to use a separate server, used as a directory for the public keys, indexed by the domains. This approach is the approach Diffie&Hellman originally intended the public key infrastructure was implemented. But the sheer number of the users makes this unmanageable as it is. If domains are to be used instead of the users, suddenly the number of keys are much more manageable. Also this may bring a performance increase as the directory can be configured solely for this purpose and a single connection would be enough even for mails with multiple recipients from multiple domains.

This last approach requires the gateways to check in their public keys after each key manipulation. And by assigning access control and different groups to key clusters, different networks for secure communications can be established.

As a final word, the key manager provides local key protection by local encryption and password protection, and at the same time makes the keys available to the parties concerned.

3.5 Implementation

For the implementation of proof of concept of this study, Java 1.6 is used as the programming language of choice. Several workstations running Windows XP SP2 with Java runtime environments and Hmail Server version 5.3 [44] are used as the model's gateways and servers. In this implementation, gateways and the mail servers are all present in the same machine.

As the client, any mail client can be used to receive the mails passing through the secure mail gateway. Microsoft Outlook 2003, Microsoft Outlook 2007, Yahoo Mail and Google Gmail are used and found compatible for both sending and receiving mails (for sending mails, the encrypted mail should be obtained by other means). Also, the included mail client agent can be used for sending mails, which can also be used to manufacture the encrypted mails.

For a typical configuration, key manager is listening to port 3000, forms secure communications through ports 3001-3999. The SMTP daemon listens to the port 25. The POP3 of the mail server listens to the port 111, reachable only locally. The gateway listens to the port 110, the usual POP3 port.

From outside, only ports 25 (SMTP), 3000 (Key exchange server) and port 110 (POP3) are available. Also it should be noted that ports 3001-3999 are opened dynamically by the key exchange server for secure communications (SRP).

Three executables comprise the proof of concept: the client, the key manager and the gateway.

3.5.1 The Client

The client was needed to be implemented as a full mail composing, encrypting and sending software only because of the limitations the current systems impose. Most of today’s systems require that the encryption certificate (public key) to be owned by the mail’s recipient, which is not the case in this model. However a key retrieving client is needed in any case.

The depiction in Figure 4 only takes into account this key retrieving portion of the software. There are two different modes of operation for this portion.

The usual usage is the destination key retrieval, where the key manager is contacted, and after authentication, the key for the destination is retrieved and loaded, in preparation for mail encryption.

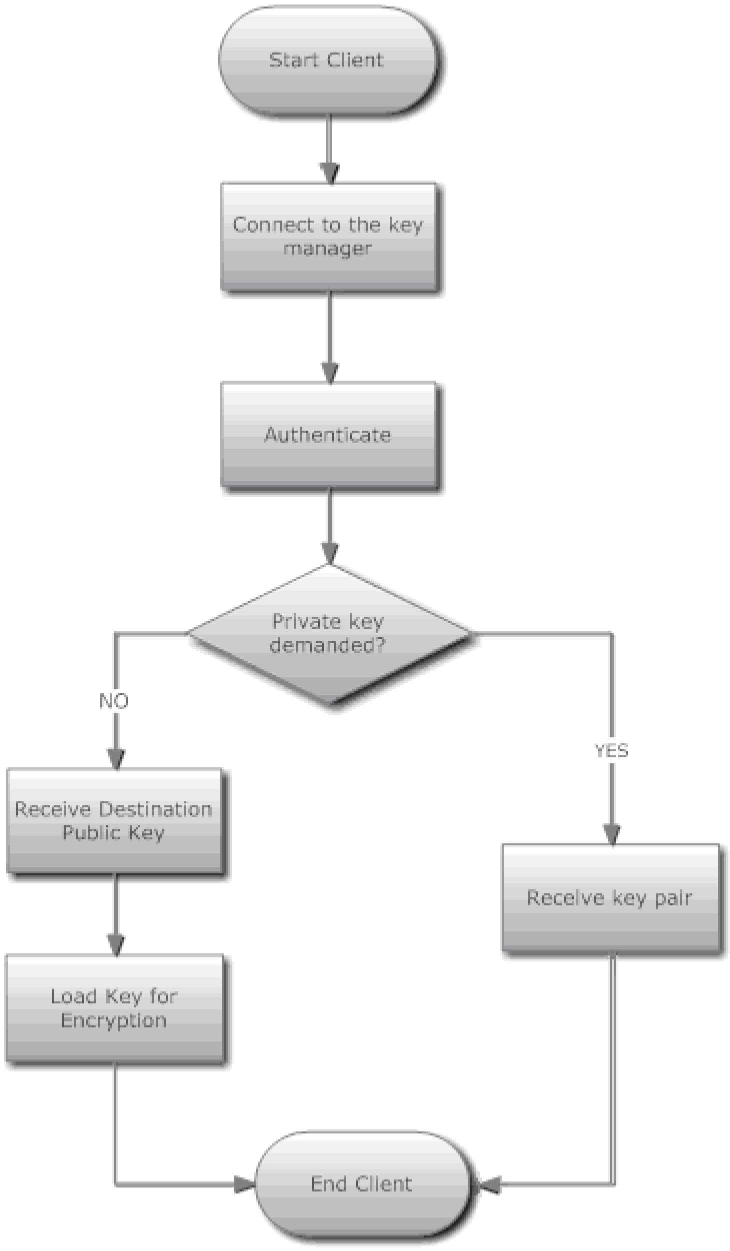

Figure 4 - The Client

The other usage, the private key retrieval is invoked to get or renew a private key for the user. This portion effectively takes care of the key-distribution problem, as the only key needed by the user, the user's own private key, is retrieved in this way. And the client software makes sure the transaction is secure, since SRP is used in authentication and transaction.

3.5.2 The Key Manager

The key manager is part of the gateway software, which is responsible of gathering the various keys of different destination gateways. The users contact their key managers for retrieving their own private keys or the public keys of their mails' recipients.

Since this is the portion of the software that is responsible of issuing and supplying the correct keys, no other part of the software has any controls. They only make the demand, and receive the answer.

When the key manager initializes, it initializes the keystore, then opens up a designated port to listen to incoming client requests. After each request, the client and the key managers handling thread jumps to a different port.

The key manager also has SRP capability for security, to be able to interface with the client software and authenticate the users. Also the keys that the key manager has are kept in a password protected, encrypted keystore for increased security.

Any time a private key is requested, the key pair is manufactured on the fly, and both the public and private keys are flushed to the client. Then the public key is stored for future reference, while the private key is destroyed.

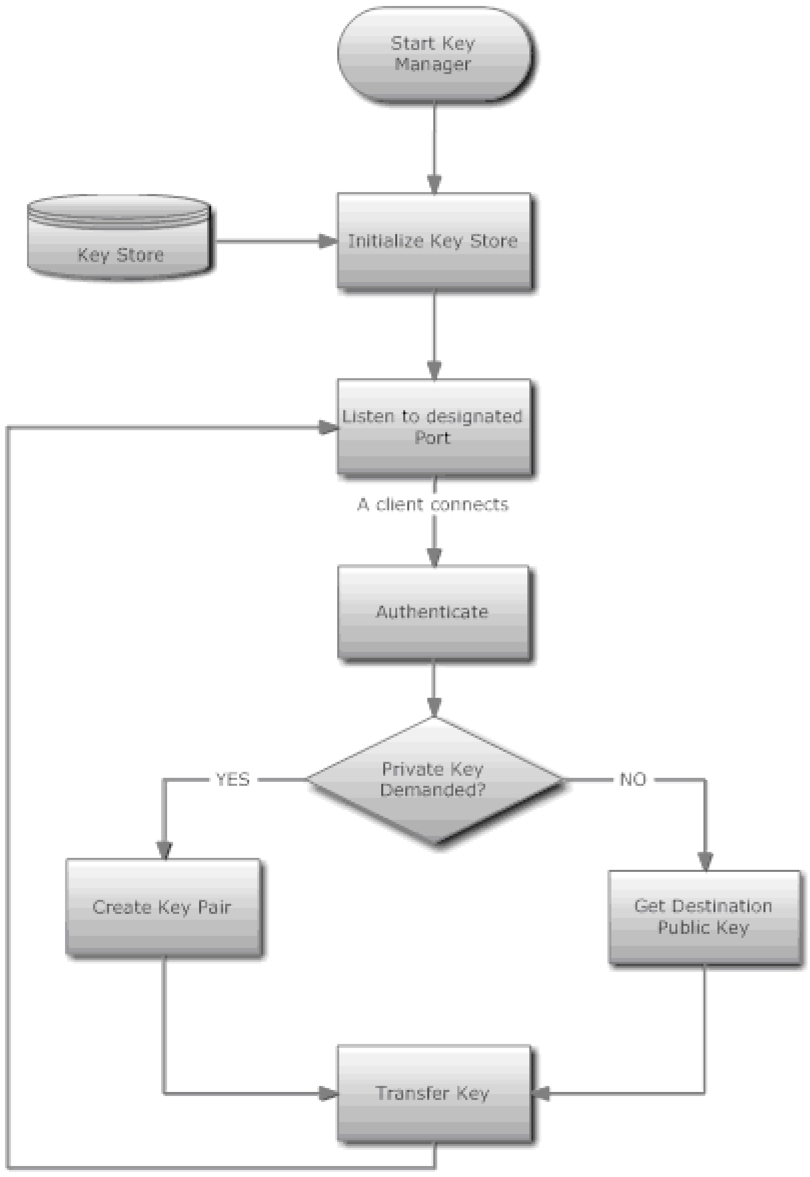

Figure 5 - The Key Manager

How the key manager retrieves the destination keys is left to the administrator. It may be through logging in as a client to the remote (destination's) key manager, which needs to be set up beforehand (requiring a trust relationship). Or it may be through a web directory where all the gateways are listed much like a phonebook. Alternatively, all gateways may have a web service publishing their public keys.

The method is not so important, as the software is modular, any of those methods, or a combination of them can be used. The proof of concept uses the first approach, logging in to a previously trusted gateway.

3.5.3 The Gateway

The gateway is the main portion of the developed software. Its main role is to act as a relay between the actual mail server (POP3/IMAP4 server) and the clients.

This portion makes use of the keystore heavily. It is used for decrypting the mail that is intended for the mail gateway. The gateway uses its own private key to decrypt the mails.

The most important detail here is, as the gateway's keys change, however frequent as the changes may be, previous keys are kept. So this brings about a crucial advantage.

One of the biggest problems of frequent key changes is what happens during and immediately after a key change. The mails that have been encrypted with an obsolete key may not be decrypted. Keeping the previous keys, and checking the mail date and using the appropriate key solve such problems.

Also keys need not be revoked, as key is demanded again for every mail. Moreover, change of the key at one place results in change of all the keys.

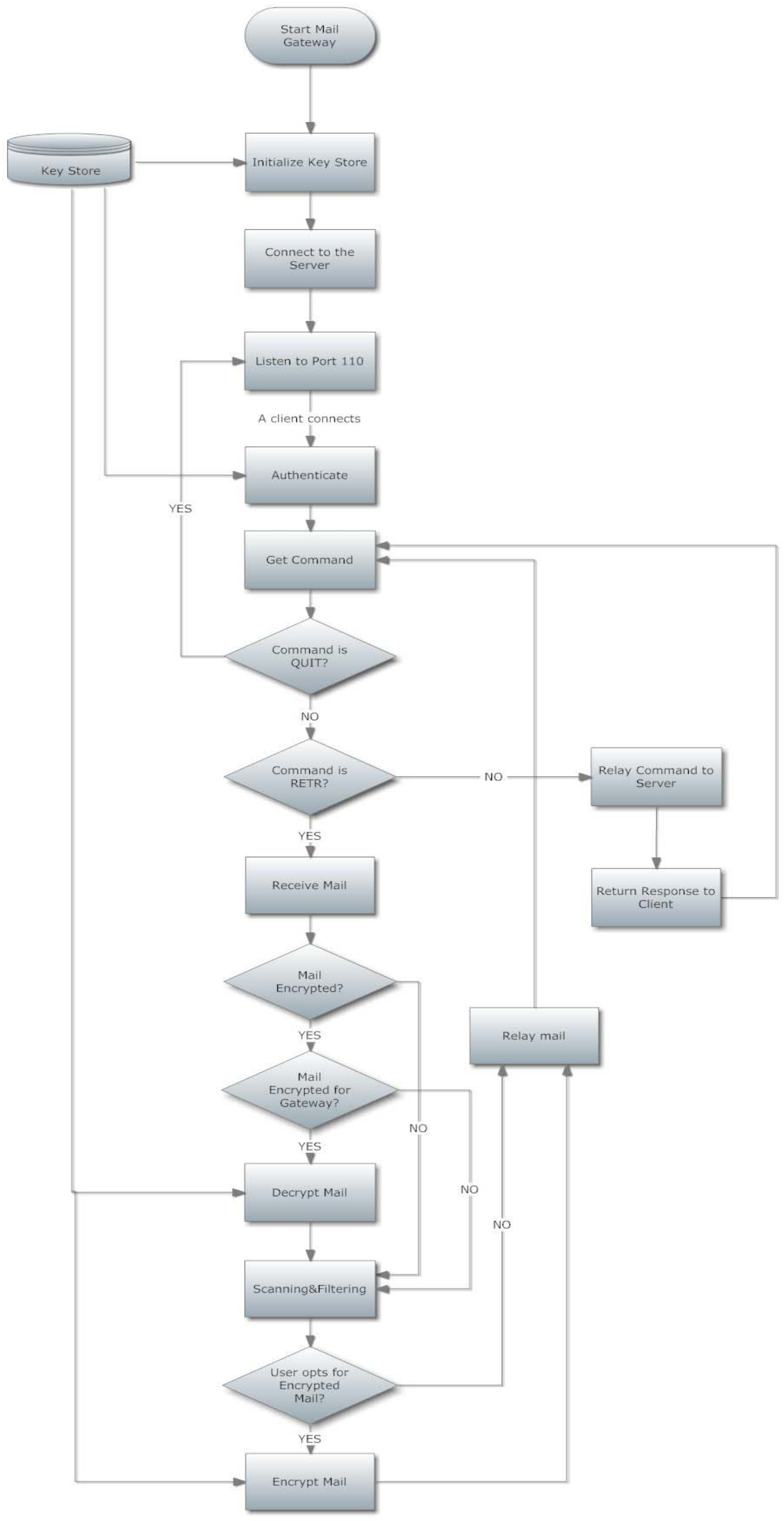

Figure 6 - The Gateway

One other advantage the gateway brings is its ability to scan and filter the incoming mail – even if they are encrypted. Figure 6 shows the procedure the incoming mails are handled. Unencrypted, encrypted for the gateway, or privately encrypted, the gateway is compatible with any kind of mail.

As a general principle, the gateway relays every command between the client and the mail server (POP3/IMAP4 server), until it intercepts a mail. Then the mail is classified and dealt with accordingly, ultimately ending up being relayed to the mail client encrypted for the client.

3.6 Comparison with Other Systems

This proposed system, the secure mail gateway, is made up of all existing technologies. Yet it manages to improve the security of the electronic mail we so abundantly use. Let's see how it can improve the security.

The most important property of the secure mail gateway is that it can be made non-optional. At least for the destinations where a secure gateway is installed, this increases the basic security. It can be readily integrated with webmails, and with its client agent, it s supported for local mail usage. Even as it is right now, not optional, it is easier to use than other systems presented here. This is an advantage if the percentage of the users opting in to encrypting their communication increases because of this. All other security methods referred here in are heavily dependent on user choice, and much harder to set up and use.

Another advantage is the lack of any third parties involved. No dependence on web of trust or certificate authorities exists as in public key infrastructure methods. The secure mail gateway builds its own trust network, and the security is only dependent on the parties involved, not some other third party. Local gateways are the sole responsible parties for their domains' security. This permits the institutions to establish their own standard for security.

OpenPGP uses the “web of trust” model for key management, which is similar in concept to the secure mail gateway's key management especially if used in the directory configuration. The users can expand the trust just by installing the gateway.

S/MIME relies on the more classical, hierarchical certificate authority model. Both OpenPGP and S/MIME depend on third parties for acquiring authentication, which may result in security risks.

Also unlike many certificate methods, the secure mail gateway does not have any costs related to certificates. Institutes may choose to purchase such certificates to be used with the gateway, but the gateway manufactures its own keys and utilizes them without any dependence on any third parties. The only cost for having the secure gateway is the need for extra space and some overhead for its installation and operation.

Also key management is much more easier as the local gateway is the sole responsible for its own key, and with no third parties involved, and keys are broadcast mainly by the gateway itself, regular changes to the key used is possible and extreme easily to practice. This alone can increase the security. Instead of keys that are valid in the order of years or months as in other systems, weekly, daily or even hourly keys may be used.

Like most commercial systems, the secure gateway relies heavily on symmetric key and asymmetric encryption, using a public key like system (SRP) in initiating key exchange, symmetric keys for key transfers, and public-private keys for mails themselves.

Also, the secure gateway chain is not only an encryption method. It is a complete system, integrating anti-virus, spam filtering and content checking.

The modification needed for the use of secure gateway chain is none; therefore the system is independent of the client or server software used, unlike other systems, thus usable in any system. Even the web-mail systems like Gmail and hotmail are usable with this system.

CHAPTER IV

RESULTS AND DISCUSSION

For the implementation of this project, Java 1.6 is used as the programming language. Several workstations running Windows XP SP3 with Java runtime environments and Hmail Server version 5.13 are used as the model's gateways and servers. In this implementation, gateways and the mail servers are all present in the same machine.

Workstations with Pentium 4 2.66 GHz processors, 2 GB RAM are used in the performance measurements. Java programming language's management class is used to directly extract the CPU times of the threads. Event logs are used to determine the CPU time of the Hmail server.

As the client software, several options are considered. Microsoft Outlook 2003, Microsoft Outlook 2007, Yahoo Mail and Google Gmail are used and found compatible for both sending and receiving mails. Additionally, included mail client agent can be used for sending mail too.

For a typical configuration, key manager is listening to port 3000, forms secure communications through ports 3001-3999. The SMTP daemon listens to the port 25. The POP3 of the mail server listens to the port 111, reachable only locally. The gateway listens to the port 110, the usual POP3 port.

From outside, only ports 25 (SMTP), 3000 (Key exchange server) and port 110 (POP3) are available. Also it should be noted that ports 3001-3999 are opened dynamically by the key exchange server for secure communications (SRP).

4.1 Performance Results

It is clear that the secure mail gateway brings many advantages. But it must be mentioned that it does so at a price, as extra software and performance costs.

There are two aspects to the performance burden the Secure Mail Gateway entails. The first one is the key distribution aspect, which is not significant in regular usage, but may become a burden to the server if every mail demands a new key. The second aspect is the mail decryption/encryption costs.

The data presented in this chapter is the average values obtained by simulations conducted on the same systems the secure mail gateway is implemented (Workstations with Pentium 4 2.66 GHz processors, 2 GB RAM) after a clean deploy. From ten to a hundred iterations of each simulation is realized to obtain the average numbers presented.

4.1.1 Key Distribution Costs

Key distribution costs have two different parts involved in its realization. The first part is the creation of the key-pairs. The second part is the actual transmission of the keys.
The same systems are used for the simulation of these operations. But it should be noted that key pairs can be created beforehand to eliminate the key creation costs, or those costs can be realized when the server is underutilized.

The key transmission costs can be carried away from the main server, easing the burden on the mail server.

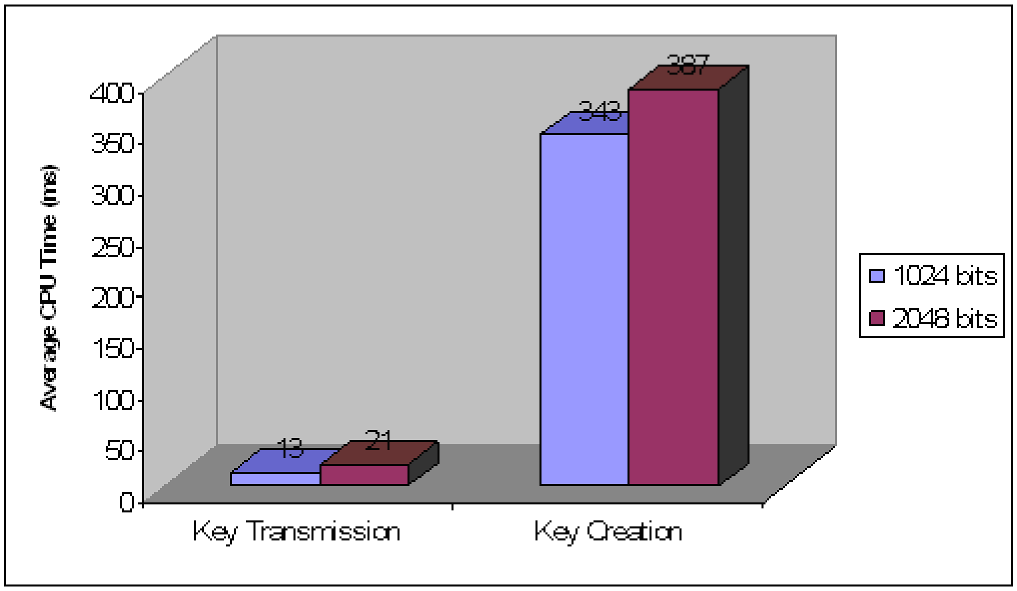

Figure 7 - Performance Cost of Key Creation and Transmission

As the simulation shows, compared to mail decryption/encryption, key transmission cost is negligible. Although key creation cost is comparable.

This is worth mentioning that only the key transmission is involved in sending mail, as the transmitted key is the target domain's public key. Key creation is only a factor when the user is receiving mails. And if a key pair is created every time a user demands its mails, it bring a burden of about transmitting 2 extra mails every time.

4.1.2 Decryption/Encryption Costs

A simulation is conducted to find the extra burden the system imposes on the server machine using a single gateway and enough users to saturate the mail demand as a model. This model is representative of the secure mail gateway as it is designed as a stand-alone system. If a domain has multiple servers, all servers can be relayed through a single secure mail gateway, or each may have its own gateway in which case user data should be shared (or distributed). However sharing user data does not affect the costs measured by this work, as the measured performance cost is CPU time. It is not expected to induce significant performance costs, as only user authentication is affected, but at this stage of the proof of concept, it was not possible to simulate such costs.

Furthermore, the costs at the clients' machines would be imperceptible, as there would be few mails that need to be encrypted/decrypted, and at worst no worse than that of conventional encryption/decryption.

However, it is entirely different situation with the mail servers. These machines need to process from thousands to millions of mail per day. And this proposed structure brings the burden of an extra encryption and decryption operation on the server machine.

This simulation used RSA algorithm with SHA-1 and Rijndael (with 256 bits of key) and used several key lengths for the domain and user keys. 512 bits of key length is considered too short for sensitive data and thus not utilized in this analysis. Only key lengths of 1024 and 2048 bits are used.

Several different mail sizes are employed in this analysis to capture the difference that will be observed with different mail sizes. It may give further insight as the average mail size is increasing with time. A Radicati group research found the average mail size to be 75 KB in 2009, whereas it was about 30 KB in 2002.

It should be noted that Java is used along with bouncy castle encryption libraries, and other implementations may provide improvements over the performance observed in these simulations.

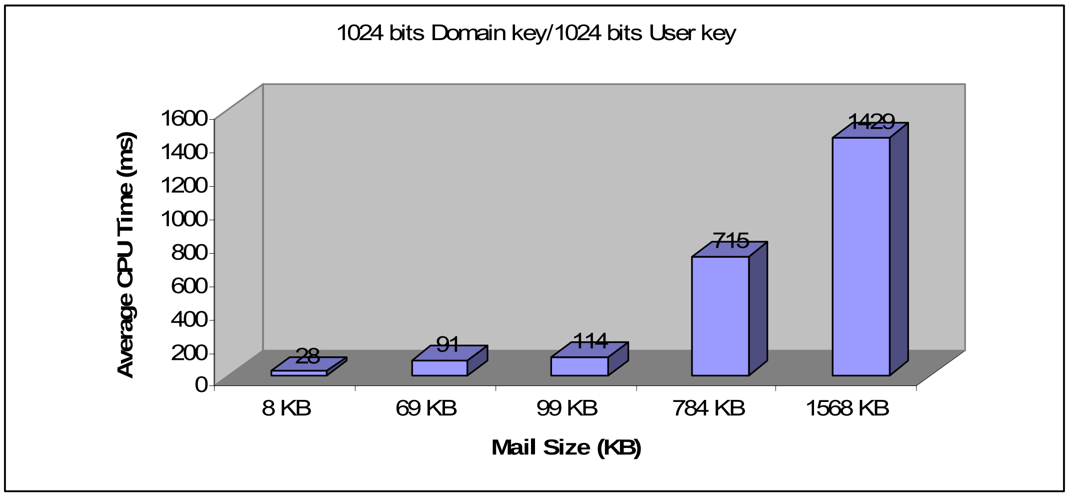

Figure 8 - 1024/1024 bits Domain/User Key Size

The results show that there is a direct correlation between the mail size to be encrypted/decrypted and the CPU time needed. A brief time constant can be observed in all mail sizes. The rest depends almost directly on the mail size.

This observation means that even on a weak system like the one used in this experiment, and with non-optimized implementation, this system can still push through over 700000 mails easily each day assuming an average mail size of 100KBs.

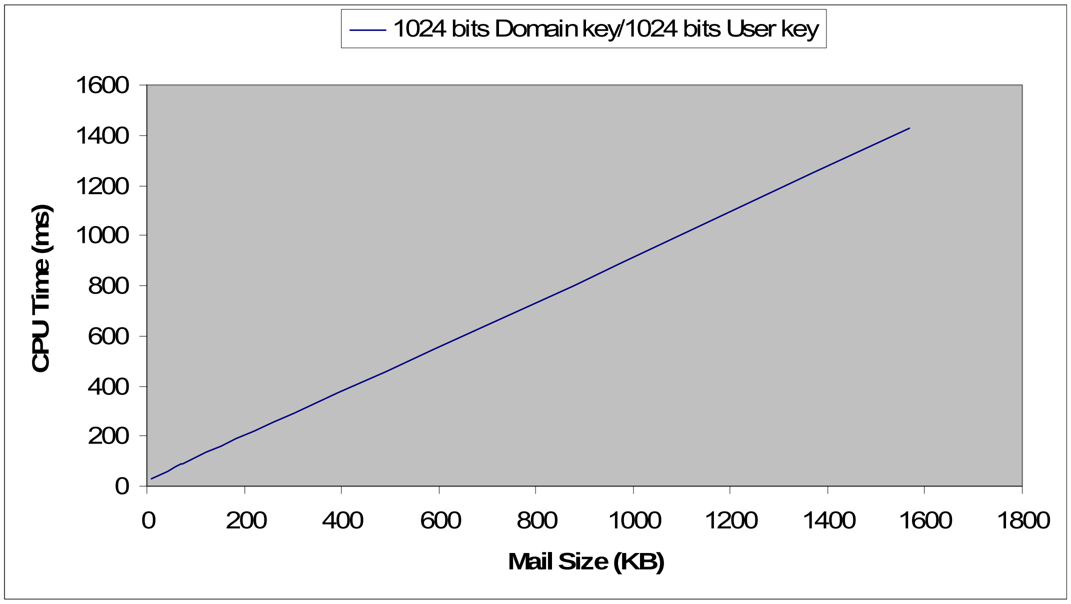

Figure 9 – CPU Time vs. Mail Size (1024/1024)

What if we increase the security? How does the performance affected?

It is logical to increase the domain key size to increase the security over the world-wide-web.

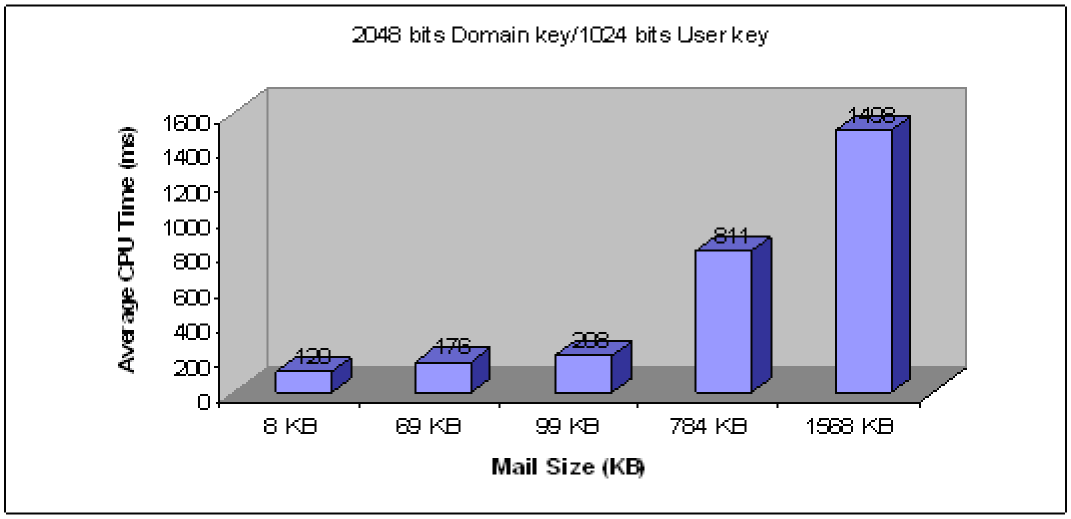

Figure 10 - 2048/1024 bits Domain/User Key Size

The performance impact of the size of the domain key is much bigger than that of the user key operation. This is mainly because the domain key is used for decryption (private key operation) and the user key is used for re-encryption (public key operation).

This can be observed in the results of the next two key sizes. When the domain key is increased in size, the performance is affected significantly. For the smallest mail size, when 1024 bits of keys are used for both the domain and user keys, it takes on average 28 milliseconds for the key operations. When the size of domain key is increased to 2048 bits; the average time increases to 120 milliseconds with a jump of 92 milliseconds. When the user key is increased to 2048 bits too, the added time is only 11 milliseconds for a total of 131 milliseconds. This is expected, as the more costly operation is the decryption.

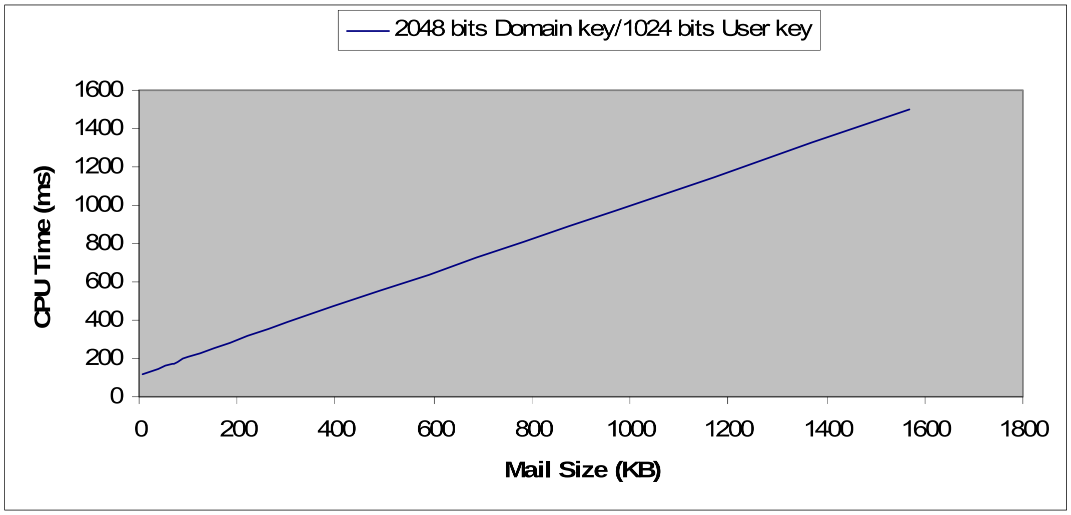

Figure 11 - Mail Size vs. CPU Time (2048/1024)

The mail throughput of the system goes down to just over 400000 mails per day for the system with 2048 bits of domain key and 1024 bits of user keys. The throughput does not change much when the user keys are set as 2048 bits, since the average time is 208 milliseconds versus 213 milliseconds, about 5 milliseconds of difference per mail, for a mail of about 100KBs.

This entails that the additional security offered by the larger keys is very affordable for the user keys, and needs consideration for the domain keys, where it will be most useful, as it will be used to protect the mail while it is traversing the internet.

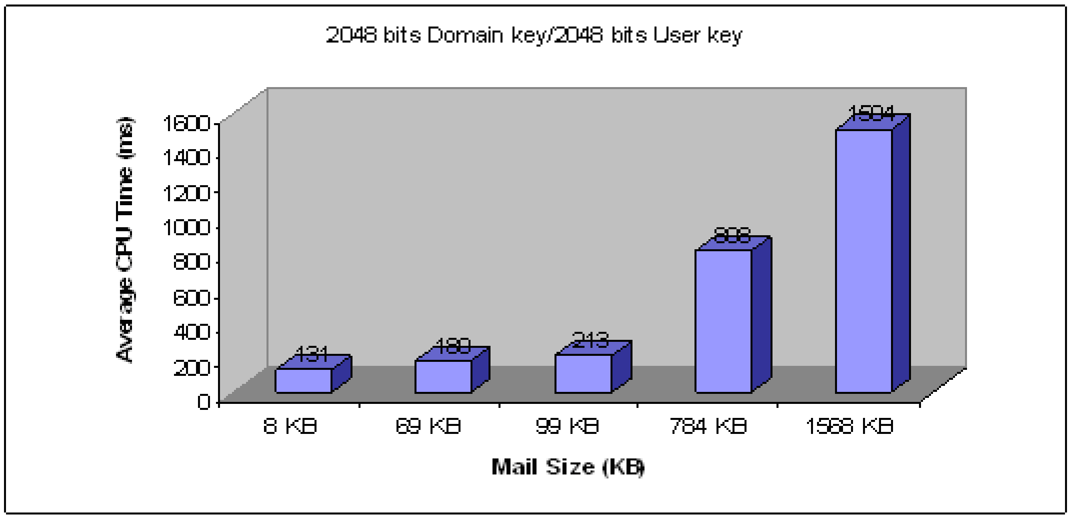

Figure 12 - 2048/2048 bits Domain/User Key Size

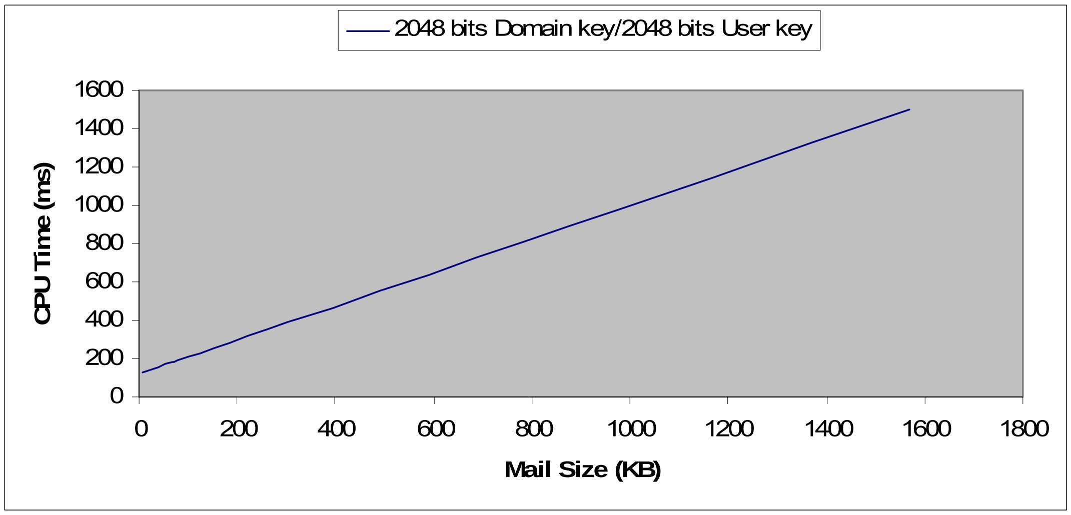

Figure 13 - Mail Size vs. CPU Time (2048/2048)

It should also be noted that, considering the user keys, the larger keys still mean a higher performance need, but it is of no concern as it will be imposed on the users' computers and will be distributed.

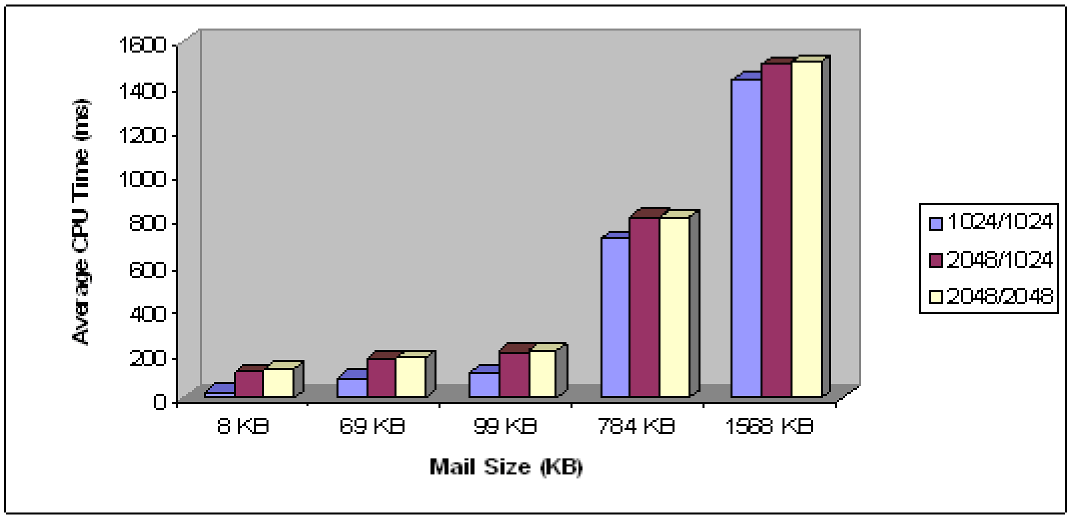

Figure 14 – General Comparison

If we consider the size of the mails in question, it can be noted that as the size of the mail increases, the size of the encrypting or decrypting keys become increasingly less relevant. The experiment shows that the time cost of the key size is just a constant. The time required to process the mail consists of this constant plus a linear function of the mail size, whose slope is not dependant on the key size.

However, it should be noted that only two distinct key sizes are used and data is not sufficient to make a definitive prediction of the effect of the key size on the relative slope of the mail size vs. CPU time function.

4.1.3 Overall Performance Costs

To grasp what the actual performance cost is to the overall mail system, we need a different perspective. Conducted simulations show that Hmail Server in its default configuration and when situated on the same computer system as the secure mail gateway on average processes and forwards a mail of 100 KBs in 182 milliseconds.

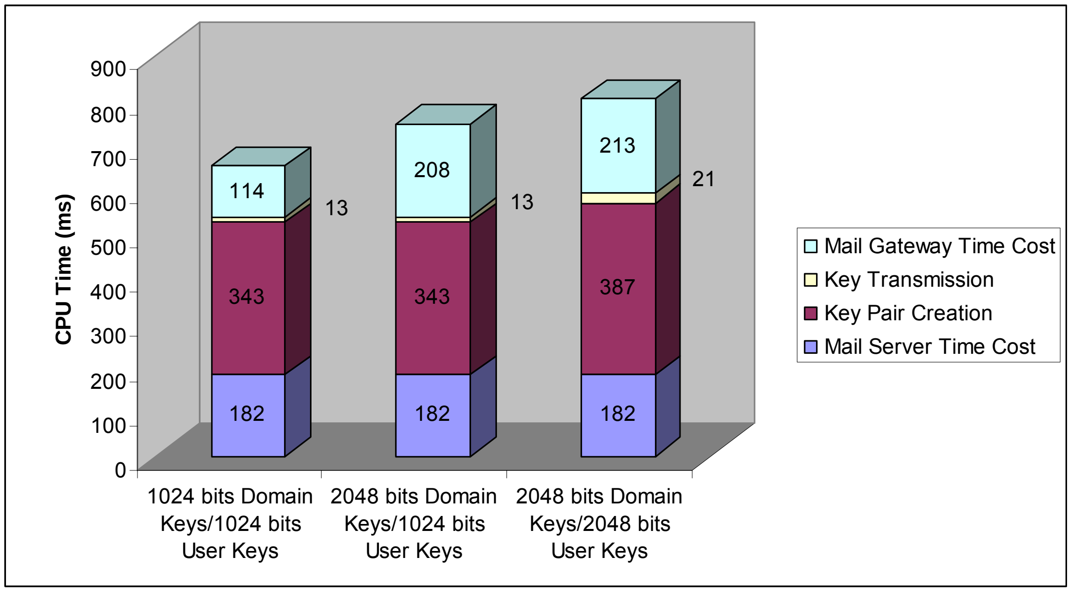

Figure 15 - Overall Average Costs for 100KB Mail Size

Assuming a single mail is demanded every time, and with it, a new key pair is generated and transmitted as well, costs are obtained as can be observed in Figure 15. It should be noted that, key creation and transmission costs become less significant as the number of mails demanded increase.

If we put this data into perspective, this means a 70%, 121% and 129% increase in performance costs for 1024 bits Domain Keys/1024 bits User Keys, 2048 bits Domain Keys/1024 bits User Keys and 2048 bits Domain Keys/2048 bits User Keys respectively if we exclude key creation costs. If key creation costs are included, these numbers become 258%, 310% and 341% respectively.

These numbers look like severe performance costs, but it should be taken into account that these costs can be carried away from the actual mail server to be assumed by a separate server, and they originate from non-optimized code.

Otherwise, if the gateway is to be located on the same server, number of mails per day should be taken as half if pre-creation of key pairs is adopted.

As a conclusion a significant performance cost is implied when the secure mail gateway is deployed to a mail server, but there exists methods to mitigate this performance cost. This cost is the trade-off of being able to use a smaller number of keys on the internet as will be discussed in the next part.

4.2 Advantage in Number of Keys

According to a compilation of sources [41], number of internet users is about 1.73 billion, where as number of e-mail users is 1.4 billion, by the end of 2009. Also, there are a total of 187 million top level domains.

Assuming these are the domains to be covered by our proposed system;

$1.73 \times 10^9 / 187 \times 10^6 \approx 9.25$ users per domain

So, instead of 9 user key pairs per domain, 1 key pair would be used on the internet corresponding to a reduction of 1/9 of the keys needed on the internet in the worst case.

Considering some of the domains do not have any e-mail users, and there are a lot of organizations with hundreds or thousands of e-mail users, we may expect a reduction in key numbers much more than 1/9 actually.

But taking into account the few most widely used internet mail domains (Gmail, Yahoo, Hotmail, AOL) constituting more than half of the e-mail users worldwide, the reduction in the number of keys would be much more dramatic. The reduction would be in the order of millions in the best case.

Only thing a user needs to use this program is his username and password, where as the only thing the administrator needs to do, is to reconfigure the IP addresses of the server, and populate the key store. So, this means that, a dramatic decrease in the number of keys (and their administration costs) can be obtained if the price is paid in performance and extra software.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Overall, the system is an easily installed, easily used, robust system that manages to increase the security of the electronic mail as it is today, through its design as well as its innovations.

This system has its upsides as well as its downsides as any system. They can be summarized as below:

Upsides:

- System provides end-to-end protection.
- The existing infrastructure is used. No modification is needed to any existing software. The existing mail infrastructure may not even know the mail is encrypted if it is so decided by the user. Even the web-mails of today may be used with the system.
- The number of keys needed in the internet is drastically reduced. Instead of using a separate key for all users of a domain, only one key is enough. This makes a directory approach (like a phonebook) possible as it is first intended by Diffie&Hellman when they introduced the public key cryptography.

- The above property also eliminates the need for any third parties for authentication.
- In the intranet, the users already have usernames and passwords. The system uses this to securely generate keys for mail and to transfer keys, by means of SRP.
- Can be made such that the user does not even know there is encryption with minimal software modification.
- Allows for frequent key renewals, as they are much fewer and centralized, increasing the overall security. Otherwise, at least as secure as any public key system.
- Extremely easy to install and use, increasing the availability of the software and consequently the overall security.
- Includes anti-virus, malware detection and content checking "slots" which can be used with third party software to increase the domain security even when the encryption is not used.
- The above property is applicable even when the electronic mail is encrypted, as the gateway can decrypt the mail. This increases the security, also eliminates one of the drawbacks of mail encryption.
- The security level may be set differently for inter-domain and intra-domain communications by the use of different keys.

Downsides:

- Needs future work for full potential.
- Performance costs cut the throughput of a mail server the secure mail gateway is deployed by half.
- Can not be used with full functionality at the same time with personal keys. Though it should be noted the system is still able to provide basic mail functionalities when there is personal encryption.

- In this state, the proof of concept does not support IMAP, or mail attachments.
- Also in this state of the proof of concept, needs user discretion for encryption.
- Uses asymmetric keys for mails, so needs larger keys and more computation power. But computation is done on the client at least half of the time, mitigating the problem.
- Decryption on the gateway may be seen as vulnerability. This is a necessary evil, but standardizing and signing the gateway code may mitigate the problem.

5.2 Future Work

The most important work that is yet to be done is integrating the client agent's functionalities into regular mail clients. Web mail clients are great candidates for such integration, as they are online all the time.

At the present, the web mail applications mostly do not support mail signing or encryption. State of the art web systems support some add-ons or external programs that may be used for such purposes, but they are cumbersome. Also current local MUAs need a separate specific certificate to encrypt the mail for the recipient, not allowing a generic certificate to sign the said electronic mail.

The proof of concept of this study includes a mail sender software to overcome this problem. As a future work, enabling the MUAs with such capability (with user discretion) may be a necessity.

The other future work is the implementing more mail retrieval options than POP3 alone, such as IMAP and implementing newer security protocols such as TLS and SSL3.0.

Also the current implementation does not have multiple recipient domain functionality built-in. Such functionality may be considered as future work.

Optimizing the performance of the existing proof of concept, mitigating the main drawback of deploying the secure mail gateway is a very important future work.

The last but not least of the future work is the setting up of key/certificate directories. Such directories, working much like DNS servers, would provide the public keys to anybody that needs to send mails to domains' users that facilitate secure gateways. These directories would eliminate the need for the process of certificate signing by third parties.

Multiple servers, directories, working in synch can improve the internet's security a lot, at least the communication security using electronic mails. This may not be seen as a future work, as this will come naturally as the usage of this system becomes more widespread.

Integrating enough secure gateways to be useful is also a future work. Even though the secure gateway in no way needs wide usage to be useful (even a single one helps to increase security), as the number of secure gateways increase, its usefulness also increases.

The system may also be merged with "*domainkeys*", which feature domains signing mail, to increase the accountability.

The most important of all the future work is the signing of the gateway software, as this will prevent tampering with the gateway software itself, preventing the compromising of the mail as it is decrypted in the gateway even though the server may be compromised.

REFERENCES

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

[2] Tanenbaum A. S. “Computer Networks (Fourth Edition)” pp: 541 – 649, 2003.

[3] Postel J. B., “Simple Mail Transfer Protocol”, RFC 821, Information Sciences Institute, University of Southern California, 1982.

[4] Crocker D., “Standard for the Format of ARPA Internet Text Messages”, RFC 822, Department of Electrical Engineering, University of Delaware, 1982.

[5] Borenstein N., Freed N., “MIME (Multipurpose Internet Mail Extensions)”, Network Working Group, RFC 1521, 1993.

[6] Terry Gray, “Comparing Two Approaches to Remote Mailbox Access: IMAP vs. POP”, 1995.

[7] Myers J, Rose M. Network Working Group RFC: 1939, “Post Office Protocol – Version 3”, 1996.

[8] Crispin M., Network Working Group RFC: 1730, “Internet Message Access Protocol – Version 4”, 1994.

[9] Crispin M., Network Working Group RFC: 1733, "Distributed Electronic Mail Models in IMAP 4", 1994.

[10] Everette T., "Examination of IMAP to facilitate user-friendly multilevel email management", 2000.

[11] Levitt L., Livengood D., Macfarlane A., "IMAP servers - What differentiates standards-based messaging systems?", Proceedings JENC8

[12] W. Diffie and M. E. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, IT-26 (6):644–654, November 1976.

[13] D. Kahn, "The Codebreakers, The Story of Secret Writing", New York: Macmillan, 1967.

[14] Leeuw, Karl de, "The History of Information Security", Informatics Institute, University of Amsterdam, 2007

[15] 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.

[16] Federal Information Processing Standards Publication 197. "ADVANCED ENCRYPTION STANDARD (AES)", 2001 November 26.

[17] Rivest R., Baldwin R., Network Working Group, RFC 2040, "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms", 1996.

[18] Diffie W., Hellman M. E., “New Directions in Cryptography”, Stanford University, 1976.

[19] R.L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, Communications of the ACM, 21 (2):120-126, February 1978.

[20] Fischer A.M., “Public Key/Signature Cryptosystem with Enhanced Digital Signature Certification”, US Patent 4,868,877, 1989.

[21] "Secure Hash Standard", United States of American, National Institute of Science and Technology, Federal Information Processing Standard (FIPS) 180-1, April 1993.

[22] Rivest, R., Network working Group, "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[23] Jones, P., Network working Group, “US Secure Hash Algorithm 1”, RFC 3174, September 2001.

[24] U. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, Advances in Cryptology - Crypto '94, Springer-Verlag (1994), 271-281.

[25] An RSA Laboratories Technical Note – Public Key Cryptographic Standards # 7 (PKCS #7), “What is Diffie-Hellman?”.

[26] Tracy M., Jansen W., Scarfone K., Butterfield J., “Guidelines on Electronic Mail Security”, NIST Special Publications 800-45, February 2007.

[27] T. Wu, "The Secure Remote Password Protocol", In Proceedings of the 1998 ISOC Network and Distributed System Security Symposium, San Diego, CA, pp. 97-111.

[28] Atkins D., Stallings W., Zimmermann P., Network working Group, “PGP Message Exchange Formats”, RFC 1991, August 1996.

[29] Callas J., Donnerhacke L, Finney H., Thayer R., Network Working Group, “OpenPGP Message Format”, RFC 2440, November 1998.

[30] Elkins M., Del Torto D., Levien R., Roessler T., Network working Group, “MIME Security with OpenPGP”, RFC 3156, August 2001.

[31] Housley R., Ford W.,Polk W., Solo D., Network working Group, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”,RFC 2459, January 1999.

[32] R. Bellman, “On a Routing Problem”, Quarterly of Applied Mathematics, 16 (1), pp.87-90, 1958.

[33] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.1: The Bellman-Ford algorithm, pp.588–592., pp.614–615.

[34] T. Hansen, D. Crocker, P. Hallam-Baker, Network working Group, “DomainKeys Identified Mail (DKIM) Service Overview”,RFC 5585, June 2009.

[35] B. Pankey, Sans Institute, “Solving Healthcare’s e-Mail Security Problem”, February 2003.

[36] E. Rescorla, Network Working Group, “Diffie-Hellman Key Agreement Method”, RFC 2631, June 1999.

[37] M. Crispin, Network Working Group, “INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1”, RFC 3501, March 2003.

[38] T. Wu, Stanford University, Network Working Group, “Telnet Authentication: SRP”, RFC 2944, September 2000.

[39] T. Wu, Stanford University, Network Working Group, “The SRP Authentication and Key Exchange System”, RFC 2944, September 2000.

[40] D. Taylor, T. Wu, N. Mavrogiannopoulos, T. Perrin, Network Working Group, “Using the Secure Remote Password (SRP) Protocol for TLS Authentication”, RFC 5054, November 2007

[41] “Internet 2009 in numbers / Royal Pingdom”, http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers/ , visited on February 2010.

[42] J. Klensin, Network Working Group, “Simple Mail Transfer Protocol”, RFC 2821, April 2001.

[43] P. Resnick, Network Working Group, “Internet Message Format”, RFC 2822, April 2001.

[44] Hmail Server 5.3, obtained from http://www.hmailserver.com/ , visited on February 2010.