East West University

Undergraduate Thesis Report

On

A Novel and Secured Email Method Based on Digital Signature and Secure/Multipurpose Internet Mail Extensions (S/MIME)

By

Shifatul Islam Shifat	**(2013-1- 80-019)**
Md. Solayman Khan	**(2013-1- 80-022)**
Marium Begum	**(2013-1- 80-023)**

Submitted to the

Department of Electrical and Electronic Engineering

Faculties of Science and Engineering

East West University

Dhaka, Bangladesh

In partial fulfillment of the requirement for the degree of

Bachelor of Science in Electrical and Electronic Engineering

(B.Sc. in EEE)

Spring 2017

East West University

Department of EEE

Thesis Title

A novel and secured email method based on Digital Signature and Secure/Multipurpose Internet Mail Extensions (S/MIME)

By

Shifatul Islam Shifat	**SID: 2013-1- 80-019**
Md. Solayman khan	**SID: 2013-1- 80-022**
Marium Begum	**SID: 2013-1- 80-023**

Submitted to the

Department of Electrical and Electronic Engineering

Faculty of Science and Engineering

East West University

Dhaka, Bangladesh

In partial fulfillment of the requirement for the degree of

Bachelor of Science in Electrical and Electronic Engineering

(B.Sc. in EEE)

Spring 2017

Approved by

-------------------------------------	--
Thesis Advisor	Department Chairperson
Fakir Mashuque Alamgir	Dr. Muhammed Mazharul Islam

Abstract

Email as one of the most popular application among internet of things, needs more attention in front of potential dangerous attacks. Digital Signature is known as one of the most prominent applications of public key cryptography to resist in front of attacks. To save secure email against fake, S/MIME is one of the best choices for digital signature application. However, S/MIME uses RSA with a weak hash function that may be broken by intruders. On the other hand, the speed of implementing S/MIME pushes some time complexity. Therefore, it needs a better scheme to refuse attacks. This thesis paper presents a new scheme of an efficient email application which uses a secure public-key encryption – digital signature. We have implemented our scheme with three most popular digital signature algorithms: RSA that currently uses by S/MIME (for comparison), ELGAMAL and Elliptic curve (ECDSA) with different hash functions in PHP as one of the best web languages. As our contribution, we discussed and analyzed the security aspects of proposed digital signature schemes to choose the best scheme which has features of low computational complexity and fast operating speed for using beside S/MIME.

Keywords: Digital Signature, S/MIME, Hash function, RSA, ELGAMAL, Elliptic curve.

Acknowledgment

This thesis work would not be possible without the courage and help of some important persons in our academic and personal life. First of all, we would like to pay our gratitude and thanks to our honorable thesis supervisor Fakir Mashuque Alamgir, Senior Lecturer, Department of Electrical and Electronic Engineering (EEE), East West University, for letting us to work under his guidance. We have reached to this stage with the help of his advices, constructive guidance, courageous instructions and we greatly appreciate for his valuable time.

We are also very much thankful to our honorable teacher, Dr. Anisul Haque, Dean of Faculty of Sciences and Engineering and Professor, Department of Electrical and Electronic Engineering (EEE), East West University, our honorable course advisor Dr. Khairul Alam, Professor, Department of Electrical and Electronic Engineering (EEE), East West University and our honorable department chairperson Dr. Muhammed Mazharul Islam, Assistant Professor and Chairperson, Department of Electrical and Electronic Engineering (EEE), East West University for giving us the opportunity to do thesis work in partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering.

We are also thankful to our other honorable faculty members and East West University Authority.

At last, we are very grateful and want to thank our parents and friends for their continuous support that was very courageous and helpful to overcome the difficulties during the whole work.

Authorization

We hereby declare that we are the sole author of this thesis. We authorize East West University to lend this project to other institution or individuals for the purpose of scholarly research.

Shifatul Islam Shifat	Md. Solayman Khan	Marium Begum
(SID: 2013-1-80-019)	(SID: 2013-1-80-022)	(SID: 2013-1-80-023)

We further authorize East West University to reproduce this thesis by photocopy or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Shifatul Islam Shifat	Md. Solayman Khan	Marium Begum
(SID: 2013-1-80-019)	(SID: 2013-1-80-022)	(SID: 2013-1-80-023)

Table of Content

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Digital Signature

The Digital Signature scheme is a modern era implementation of cryptographic system that provides data security which is passed over the insecure channel. In analog reality any important data or we can say message, is signed by specific entity that ensures the authenticity, integrity and validity of the signed paper. Another important property is non-repudiation, which means the entity who has signed the data or the paper cannot deny the signature or the actions later. But in the modern era, the digital signature system is used that ensures fundamental objectives of secure communications which are:

a. Confidentiality: Data should be secret and only be accessible by the authorized personnel. Even though the channel is insecure the attacker will not find the meaning of actual data or message.
b. Data integrity: It is ensured that the signed data has not been altered by adversary entity.
c. Entity authentication: It will provide the identity verification that the sender of the data has the authorization of the data and the receiver is convinced about the origin of the signed data.
d. Non-repudiation: The authorized entity who signed and sent the data will not be able to deny later that it is not signed by them. As the verification process is related to the authentication of the signer public and private key, digital signature prevents future denial issues.

1.2 Public Key Cryptography

Public key cryptography is also known as asymmetric key cryptography that is used in the digital signature scheme. The digital signature scheme seems a little bit impractical due to non-repudiation which is difficult to achieve using symmetric key cryptography and that is why a public key system is highly used scheme. In this system, a key pair is generated to proceed the encryption and decryption of the data. The key is not secret and shared over the channel is public-key (K_{pub} *or e*) and the key which is kept secret by the owner of the both key is called private-key (K_{priv} *or d*). The key pair is generated such a way that it should be impossible or too much time consuming problem to compute the private key from the known public-key. To make this happen usually discrete logarithm problem is used to generate the key pair and proceed the cryptographic system.

1.3 Digital Signature Procedure

Let's assume there is two entities 'A' and 'B'. 'A' needs to sign some important data 'm' and send over the insecure channel (internet) to 'B'. After receiving the signed data, 'B' does the verification process that whether the signed data originally come from the 'A' or not. It proves the authenticity of the message.

To sign a data or massage 'm':

'A' (entity) must have to generate a key pair (public key and private key) and find the message digest ($h= H (m)$) using Hash function and encrypt it using the private key. 'A' will send the signed and encrypted message, is over the insecure channel.

To verify the data 'm':

'B' will get the public key of 'A' and decrypt the signed, encrypted messages with it. 'B' will also generate the hash of message 'm' that is $h'=H (m)$.

If the encrypted digest that is found over the insecure channel is equal to the 'h′', that is $h=h'$, then it is verified that 'A' signed the data with his private key. Thus, authenticity is ensured and as only 'A' owns the private key, 'A' cannot deny the actions.

There are many schemes of digital signature and uses according to the necessity of the users. Entity 'A' can also send some encrypted message using the public key of 'B' which ensures that only 'B' can decrypt the message 'm' using the private key of 'B' and this is how confidentiality is achieved.

In summary, anyone with a known public key can cipher the message, but only the person who has the private key can decipher the message. In asymmetric cryptosystem, if it is used to encrypt a message with a public key, it will not be possible to decrypt the data or message with the same key. One must have the private key to decrypt the data. Also private key can be used to encrypt the message by the owner of it. It will work as signing the data, though anyone can decipher the data as the public key is known to all, it verifies that the data is signed by the authorized person and is not altered by an adversary group.

There are some different security schemes which are commonly used in public-key crypto-system.

1. RSA public-key encryption and signature schemes based on integer factorization problem.

2. The ElGamal public key encryption and signature schemes and their variants such as the Digital Signature Algorithm (DSA). This scheme is based on the discrete logarithm problem.

3. The Elliptic curve cryptographic schemes and elliptic curve discrete logarithm problem is used.

A signature scheme is a five-tuple (P, A, K, S, and V) where the following conditions are satisfied:

1. ‘P’ is a finite set of possible message.
2. ‘A’ is a finite set of possible signatures.
3. K, the key space is a finite set of possible keys.
4. For each k ϵ K, there is a signing algorithm sig_k ϵ S and a corresponding verification algorithm ver_k ϵ V.

Each sig_k: P, A and ver_k P*A – {true, false} are functions such that the following equation is satisfied for every message X ϵ P and for every signature Y ϵ A:

$$\text{Ver k(x)} = \begin{cases} true, & if\ y = sig_k(x) \\ false, & if\ y \neq sig_k(x) \end{cases}$$

A pair (x, y) with xϵ P and yϵ A is called a signed message. This algorithm is applied in all three schemes that are focus in this theses, meaning RSA, ElGamal, and ECDSA.

Stinson is represented a randomized mechanism for ElGamal. RSA was proposed shortly after discovery of public key cryptography [9] and is so popular for its speed and simplicity and finally the Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm (DSA). It is the most widely standardized elliptic curve based digital signature scheme, appearing in ANSI X9.62, FIPS 186-2, IEEE 1363-2000 and ISO/IEC 15946-2 standards as well as several draft standards [9].

1.4 RSA Encryption

This encryption system is introduced in 1977 and named after the inventors Ron Rivest, Adi Shamir, and Leonard Adleman.

The basic RSA scheme is,

$$m^{e\,d} \equiv m\ (mod\ n)$$

And we find the private key (d) and public key (e) using the equation,

$$e\ d \equiv 1\ mod\ \phi(n)$$

$$or, \quad e\ d \equiv 1\ mod\ (p-1)(q-1)$$

Where, p and q are two large prime of same bit length, n= p q, and prime factorization of n which is $\phi(n) = (p\text{-}1)\ (q\text{-}1)$. To generate the key pair generation, we have to select:

(a) An arbitrary integer e with $1 < e < \phi$ and $\gcd(e, \phi) = 1$

(b) Compute the integer d satisfying $1 < d < \phi$ and the equation, $ed \equiv 1 \ (mod\ \varphi)$

So, Steps for RSA encryption and decryption:

Encryption: RSA public key (n, e), plaintext $m \in [0, n-1]$, Cipher text c.

1. Compute $c = m^e \ mod\ n$

Decryption: RSA private key d, cipher text c.

1. Compute $m = c^d \ mod\ n$

Thus we get the plaintext m using the private key. Anyone who knows the public key can encrypt the plaintext into c, send it over the insecure channel and only the person who has the private key d, can decipher the text. This is the basic of RSA encryption system.

Now if we want to use the RSA theory in the digital signature system, we need a scheme in which the theory will be useful to sign the message and also verify the signature. To do that, we need a signature algorithm and a verification algorithm to proceed the digital signature process.

RSA signature scheme:

Basic RSA signature generation

RSA public key (n, e), RSA private key d, message m, signature s

1. Compute $h = H(m)$ where H is a hash function.
2. Compute $s = h^d \ mod\ n$

Basic RSA signature verification: Acceptance or rejection of the signature. [16]

RSA public key (n, e), message m, signature s.

1. Using hash function find the message digest, $h = H(m)$
2. Compute $h' = s^e \ mod\ n$
3. If $h = h'$, then signature is valid. If $h \neq h'$, the signature is not valid.

1.5 Digital Signature Algorithm (DSA)

In 1991, Digital Signature Algorithm (DSA) was proposed by the U.S. National Institute of Standards and Technology (NIST) and was specified in a U.S. Government Federal Information Processing Standard (FIPS 186) called the Digital Signature Standard (DSS). In this scheme, discrete logarithm and ElGamal encryption are used.

With discrete logarithm systems, a key pair is associated with a set of public domain parameters (p, q, and g) where ‘p’ is a prime, ‘q’ is a prime divisor of (p −1), ‘g’ is a base generator and $g \in [1, p-1]$ has order of ‘q’.

An integer ‘x’ is called the private key selected uniformly at random from the interval [1, q−1]. The corresponding public key is $y = g^{x} mod\ p$ [16]. The problem of determining ‘x’ from giving domain parameters (p, q, and g) and which is the discrete logarithm problem (DLP). Also in ElGamal public key encryption scheme, ‘k’ is called an ephemeral key that is randomly selected by the sender and ‘m’ is the plaintext or the message.

In DSA scheme, an entity ‘A’ with private key x signs a message digest h (where $h= H(m)$) by selecting random ephemeral key ‘k’ from the interval [1, q−1]. The signature pair (r, s) is sent over the insecure channel and the recipient will verify the signature pair (r, s) through a verification algorithm whether it is valid or invalid. Here is the brief summary of the algorithm.

Steps for DSA signature generation:

DL domain parameters (p, q and g), private key x, message ‘m’, Signature (r, s).

1. Selection of ephemeral key ‘k’ where k ∈R [1, q −1].
2. Computation of $T = g^{k}\ mod\ p$
3. Compute $r = T$ mod q. If $r = 0$ then go to step 1.
4. Compute the hash fingerprint of plaintext m which is $h = H(m)$
5. Compute the signature $s = k^{-1}(h + xr)\ mod\ q$. If $s = 0$ then go to step 1.
6. Generated signature pair *(r, s)* [16]

Steps for DSA signature verification: Acceptance or rejection of the signature.

DL domain parameters (p, q, g), public key y, message m, signature (r, s).

1. Verification whether ‘r’ and‘s’ are integers in the interval [1, q −1] or not. If not, reject the signature.
2. Compute the hash fingerprint of plaintext m which is $h = H(m)$.
3. Compute $u1 = h\, s^{-1}\ mod\ q$ and $u2 = r\, s^{-1}\ mod\ q$
4. Compute$T' = g^{u1}\, y^{u2}\ mod\ p$
5. Compute $r' = T' mod\ q$
6. If $r = r'$then, the signature is valid and accept the signature [16].

In these process, all the prime numbers are very large, for example, p can be 1024-bit length and q can be 160 bit. The hashed message digest h will be 160 bit for SHA-1 or 256 bit for SHA-2 hash function. For securing the algorithm, the ephemeral key, ‘k’ should not be re-used.

1.6 ElGamal Cryptosystem

ElGamal is a public-key cryptosystem technique was designed by Dr. Taher ElGamal in 1985. In cryptography, the ElGamal encryption system is an asymmetric key encryption algorithm. The ElGamal Cryptosystem is based on the discrete logarithm problem. The discrete algorithm fits into the family of public key cryptographic algorithms. Therefore, it makes use of a key separated into a public and a private party. The knowledge of the private part makes the decryption easy and this is a fundamental aspect of this system. It is virtually impossible to decrypt the message in acceptable time, if the private key is unknown.

1.6.1 The ElGamal Public Key Encryption Algorithm

The ElGamal Algorithm provides an alternative to the RSA for public key encryption. The basic requirement for a cryptographic system is at least one key for symmetric algorithms and two keys for asymmetric algorithms and key generation steps are similar to this. With ElGamal, only the receiver needs to create a key in advance and publish it.

Let’s assume two entities Alice and Bob. Key generation procedure of Alice will be:

Alice chooses

i) A large prime p_A

ii) A primitive element α_A Mod p_A,

iii) A (possibly random) integer d_A with $2 \le d_A \le p_A - 2$.

Alice computes

iv) $\beta_A \equiv \alpha_A{}^{dA} \ (mod\ p_A)$.

Alice’s public key is (p_A, α_A and β_A). Her private key is d_A.

Bob encrypts a short message ‘M’ ($M < p_A$) and sends it to Alice like this:

i) Bob chooses a random integer k (which he keeps secret).

ii) Bob computes $r \equiv \alpha_A{}^k \ (mod\ p_A)$ and $t \equiv \beta_A{}^k M \ (mod\ p_A)$, and then discards k.

Bob sends his encrypted message (r, t) to Alice.

When Alice receives the encrypted message (r, t), she decrypts (using her private key d_A) by computing tr^{-dA}.

$$\begin{aligned} \text{Note}\quad tr{-}dA &\equiv \beta_A{}^k\, M\, (\alpha_A{}^k)^{-dA} && (mod\ p_A) \\ &\equiv (\alpha_A{}^{dA})^k\, M\, (\alpha_A{}^k)^{-dA} && (mod\ p_A) \\ &\equiv M && (mod\ p_A) \end{aligned}$$

Even, if Eve (attacker) intercepts the cipher text (r, t), she cannot perform the calculation above because she doesn't know d_A.

$$\beta_A \equiv \alpha_A^{dA} \ (mod\ p_A), \quad so\ d_A \equiv L_{ogA}\ (\beta_A)$$

Eve can find d_A if she can compute a discrete log in the large prime modulus p_A, presumably a computation that is too difficult to be practical.

If 'M' is a longer message, so it is divided into blocks, he should choose a different 'k' for each block.

If he encrypts two messages (or blocks) M_1 and M_2, using the same 'k', producing cipher text:

$$(r_1, t_1) = (\alpha_A^k, \beta_A^k M_1), \qquad (r_2, t_2) = (\alpha_A^k, \beta_A^k M_2)$$

Then $t_2 t_1^{-1} \equiv M_2 M_1^{-1}$ (mod p), $M_2 \equiv t_2 t_1^{-1} M_1$ (mod p). If Eve intercepts both cipher text messages and discovers one plaintext message 'M_1', she can compute the other plaintext message M_2.

1.7 The Elliptic Curve Digital Signature Algorithm (ECDSA)

The Elliptic Curve Digital Signature Algorithm (ECDSA) was standardized in the US in 1998 by the American National Standard Institute (ANSI). Elliptic curves have several advantages over RSA and DSA. In particular, in the absence of strong attacks against elliptic curve cryptosystems (ECC), bit lengths in the range of 160–256 bit can be chosen which provide the security equivalent to 1024–3072-bit RSA and DL schemes. The shorter bit length of ECC often results in shorter processing time and in shorter signatures [17].

1.7.1 The Algorithm

The arithmetic to be performed for actually computing an ECDSA signature is entirely different from that used for DSA. The ECDSA standard is defined for elliptic curves. But the steps in the ECDSA standard are related to the DSA scheme [17].

Key generation:

Let's design an elliptic curve E with modulus p.

Now let's:

1. Take coefficients 'a' and 'b'
2. Pick an arbitrary point A to generate a cyclic group of prime order q
3. Choose a random integer d with 0 < d < q
4. Compute $B = dA$.

The keys are now:

Public Key, $k_{pub} = (p, a, b, q, A, B)$

Private Key, $k_{pr} = (d)$

1.7.2 Signature and Verification

An ECDSA signature consists of a pair of integers (r, s). Each value has the same bit length as q. It makes fairly compact signatures. Using the public and private key, the signature for a message x is computed as follows:

Signature Generation:

1. Choose an integer random key 'k_E' with $0 < k_E < q$
2. Compute $R = k_E A$
3. Compute $r = x_R$
4. Compute 's' $\equiv (h(x) + d \cdot r)\, k_E^{-1} \bmod q$

In step 3, the x-coordinate of the point R is assigned to the variable r. The message x has to be hashed using the function h in order to compute's'. The hash function output length must be at least as long as 'q'.

Signature Verification:

Compute auxiliary value $w \equiv s^{-1} \bmod q.$

2. Compute auxiliary value $u_1 \equiv w \cdot h(x) \bmod q.$

3. Compute auxiliary value $u_2 \equiv w \cdot r \bmod q.$

4. Compute $P = u_1 A + u_2 B.$

5. The verification conditions:

Now If, $x_P = r \bmod q$ ⇒ valid signature

Else ⇒ invalid signature

In the last step, the notation 'x_P' indicates the x-coordinate of the point 'P'. The verifier accepts a signature (r, s) only if the 'x_P' has the same value as the signature parameter 'r mod q'. Otherwise, the signature should be considered invalid.

The main thing is that the expression '$u_1A + u_2B$' is equal to '$k_E A$', if the correct signature and key is used. But this is exactly the condition that we check in the verification process by comparing the x-coordinates of $P = u_1A + u_2B$ and $R = k_EA$.

1.8 MIME (Multipurpose Internet Mail Extensions)

MIME (Multipurpose Internet Mail Extensions) is an extension of the original Internet e-mail protocol. To exchange different kinds of data files on the Internet such as audio, video, images, application programs, and other kinds, as well as the ASCII text, MIME lets people use the protocol.

MIME was designed to exchange different format of email to support non-ASCII characters, attachments other than text format, and message bodies which contain multiple parts. MIME describes the message content type and the type of encoding used with the help of headers. Through SMTP in MIME format all manually composed and automated emails are transmitted. Sometimes emails are referred to as SMTP/MIME email. The MIME standard defines the content types which are of prime importance in communication protocols like HTTP for the World Wide Web. The data are transmitted in the form of email messages through HTTP even though the data are not an email.

The features offered by MIME to email services are as follows:

i. Support for multiple attachments in a single message
ii. Support for non-ASCII characters
iii. Support for layouts, fonts and colors which are categorized as rich text.
iv. Support for attachments which may contain executable, audio, images and video files, etc.
v. Support for unlimited message length.

1.8.1 S/MIME (Secure/Multipurpose Internet Mail Extensions)

S/MIME (Secure/Multipurpose Internet Mail Extensions) is a standard for public key encryption and signing of MIME data. S/MIME is used for sending digitally signed and encrypted messages through email. It is a widely accepted method, or more precisely a protocol. S/MIME allows encrypting emails and digitally signing into emails. When S/MIME is used with an email message, it helps the people who receive that message to be certain that what they see in their inbox is the exact message that started with the sender. It also helps to find the specific sender and not from someone pretending to be the sender. To do this, S/MIME provides for cryptographic security services such as authentication, message integrity, and non-repudiation of origin (using digital signatures). It also helps enhance privacy and data security (using encryption) for electronic messaging.

S/MIME uses public key cryptography (an asymmetric system) to sign and encrypt e-mail. Basically, every participant has two keys: A private key, which is kept secret and a public key, which is available to everyone. Files or mails encrypted using someone’s private key can only be decrypted using his public key and vice versa.

1.9 Hash Function

A hash function is very important element in crypto-system. Normally a hash function is a mathematical function which converts or generates a unique output that represents corresponding an input to the function. It is a one-way function. From the hash output, it is impossible to generate the original input. In modern cryptography system or in digital signature scheme hash function is the most needed element which provides the confidentiality and integrity.

The output of the input is called as 'hash value' or 'message digest'. It is also known as the fingerprint and always of fixed value. The common features of cryptographic hash functions are:

1. Arbitrary inputs: The data or the input that we need to generate the fingerprint can be any length. It can be a large data file or PDF or mp3 file with arbitrary length, but always generates with a fixed length output. Popular hash functions generate values between 160 and 512 bits. Assuming arbitrary bit length message m, hash value or fingerprint 'h' (fixed bit length) and hash function *H (m)*, mathematically, we can represent as:

$$h = H(m)$$

2. Efficient: Hash function usually compressed the input such a way that fingerprint must be in very short compare to the input. So it must be fast enough to generate the digest. Also it cannot be too fast generation hash function.

3. Pre-image resistance: This property means that it should be computationally hard to reverse a hash function. Given a hash value 'h_1', it should be a difficult process to find any input value 'm_1'that produces h_1. Also the fixed fingerprint must be random which means the hash value will not express any idea of the original input and be completely different. If any bit is changed in original message, then the hash function must produce completely different fingerprint.

This property protects against an attacker who only has a hash value and is trying to find the input.

4. Second Pre-image resistance: It means given an input and its hash, it should be hard to find a different input with the same hash. Given one input 'm_1' and its hash value 'h_1', it should be very hard to find any different input 'm_2' that generates 'h_2' such that, $h_1=h_2$.

This property of hash function protects against an attacker or adversary group who has an input value and its fingerprint, and wants to substitute different input message with same fingerprint because in digital signature the hash fingerprint must be identical in verification process.

5. Collision resistance: This feature means it should be very hard and time consuming computation to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function. For a hash function H, it must be hard to find any two different inputs ‘m’ and ‘n’ such that $H(m) = H(n)$. Also it is understandable that, as the input is arbitrary length and hash function generates a fixed length digest so there will be a collision. But a hash function should be built such way that finding a collision should be very difficult.

1.9.1 Hash algorithm

We already know that, hash functions can process an arbitrary-length message and produce a fixed-length output. In practice, this is achieved by segmenting the input into a series of blocks of equal size. The hash function processes these blocks sequentially. It has some compressing function and known as Merkle–Damgard construction [17]. The hash value of the input message is then defined as the output of the last iteration of the compression function. The hash function generates a hash code by operating on two blocks of fixed-length binary data. Hashing algorithm is a process for using the hash functions, specifying how the message will be broken up and how the results from previous message blocks are chained together.

Currently the most popular cryptographic hash function belongs in the SHA-1/256/224/512 family. In this thesis, we want to analyze our implementation with some of these hash functions to realize the best candidate for each scheme.

1.9.2 Message Digest (MD)

MD4 is a message digest algorithm developed by Ronald Rivest. In practice, the most popular ones have been the hash functions of what is called the MD4 family. MD5, the SHA (secure hash algorithm) family and RIPEMD are all based on the principles of MD4. It uses 32-bit variables, and all operations are bitwise Boolean functions such as logical AND, OR, XOR and negation. All subsequent hash functions in the MD4 family are based on the same software-friendly principles. In 1991 Rivest proposed a strengthened version named MD5. Both hash functions compute a 32 character of 128-bit output and they possess a collision resistance of about 2^{64}. MD5 became extremely widely used in Internet security protocols, for computing checksums of files or for storing of password hashes. But soon weakness was seen. In 1996, a partial attack against MD5 by Hans Dobbertin led to more and more experts recommending SHA-1 as a replacement for the widely used MD5. In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

1.9.3 Secure Hash Algorithm (SHA)

Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. In 1995, SHA-0 was modified to SHA-1 by the US National Institute of Standards and Technology (NIST).The difference between the SHA-0 and SHA-1 algorithms lies in the schedule of the compression function to improve its cryptographic security. Both algorithms have an output length of 160 bit. SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols, including Secure Socket Layer (SSL) security. But due to increasing the security level some of the variants was introduced like SHA-224, SHA-256, SHA-384 and SHA-512, with message digest lengths of 224, 256, 384 and 512 bits, respectively.SHA-1 is also facing security issues with founding collision and SHA-2 is widely used nowadays.

1.9.4 RIPEND

The RIPEND is an acronym for RACE Integrity Primitives Evaluation Message Digest. This family was designed by open research community, and generally known as a family of European hash functions. The set includes RIPEND, RIPEMD-128, and RIPEMD-160. There also exist 256, and 320-bit versions of this algorithm. RIPEMD-160 is an improved version and the most widely used version in the family.

1.10 Open PGP

OpenPGP is a protocol for encrypting email communication using public key cryptography. It is one of the most widely used email encryption standard. It is basically dependent on the Pretty Good Privacy (PGP) software. The OpenPGP Working Group of the Internet Engineering Task Force (IETF) defined it as a Proposed Standard. It was created by Phil Zimmermann. OpenPGP is a non-proprietary protocol for encrypting email communication using public key cryptography. The OpenPGP protocol defines standard formats for encrypted messages, signatures, and certificates for exchanging public keys.

1.10.1 Confidentiality

PGP can be used to send messages very much confidentially. For this, PGP creates symmetric-key encryption and public-key encryption. The message is encrypted using a symmetric encryption algorithm, which requires a symmetric key. Each symmetric key is used only once and is also called a session key. The message and its session key are sent to the receiver. The session key needs to be sent to the receiver so that can decrypt the message, but to protect it during transmission; it is encrypted with receiver's public key. Only by using the private key, the receiver can decrypt the session key or the symmetric key.

1.10.2 Development

OpenPGP is still under active development. Many email clients provide OpenPGP-compliant email security. The current specification is RFC 4880 (November 2007), RFC 2440 [18]. RFC 4880 specifies required algorithms consisting of ElGamal encryption, SHA-1, DSA and Triple DES. Beyond these, many other algorithms are supported. The standard was extended to support encryption based on elliptic curve cryptography (ECDSA, ECDH) in 2012.

Chapter 2: Literature Survey

2.1 Protocol for the Secure E-mail in Mobile Environments

In the paper, titled as “SMEmail-a New Protocol for the Secure E-mail in Mobile Environments” [1], a new security application-layer protocol was introduced. It is called SMEmail. It provides several security attributes such as confidentiality, integrity, authentication, non-repudiation and forward secrecy of message confidentiality for the electronic mails [1]. SMEmail offers an elliptic curve-based public key solution that uses public keys for the secure key establishment of a symmetric encryption, and is so suitable for the resource restricted platforms such as mobile phones.

2.2 Internet Mail Extensions (S/MIME)

S/MIME provides a consistent way to send and receive secure MIME data. S/MIME provides the following cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin (using digital signatures), and data confidentiality (using encryption). This is based on the popular Internet MIME standard.

In the paper titled as “Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification” [2], the author said that S/MIME can be used by traditional mail user agents (MUAs) to add cryptographic security services to mail that is sent, and to interpret cryptographic security services in mail that is received. Further, by the support of cryptographic security services the S/MIME can be used in automated message transfer agents that do not require any human intervention, such as the signing of software-generated documents and the encryption of FAX messages sent over the Internet.

The author of the paper titled as “Towards Usefully Secure Email” [3], believes that, it is important to identify the tasks involved at which humans excel (and at which computers do not), and then design the system accordingly. To demonstrate this principle, the author focused on Attribute-Based, Usefully Secure Email (ABUSE).

It is very disappointing that most users cannot, or simply do not, secure their everyday internet communications. The attention from the author of the paper titled as “A Case (Study) For Usability in Secure Email Communication” [4] on available public-key cryptography techniques for digitally signing and encrypting email. The author’s first step was to use Secure/Multipurpose Internet Mail Extensions (S/MIME), a standard for signing and encrypting email, because it’s already supported by popular email clients such as Apple Mail, Outlook Express, and Mozilla's Thunderbird. The author also discussed about how Fingerprint verification can be important to secure email Communication. A fingerprint is a secure hash of the public key and is a smaller, digested form. Verifying that the exchanged key's fingerprint matches the original key's fingerprint is a much faster way to verify the key's authenticity.

2.3 Domain Keys Identified Mail

Email protocols were designed in a day when internet usage was a cooperative and flexible thing. Email protocols were not designed to provide protection against falsification of a message address of origin, referred to today as "spoofing". The paper titled as "Domain Keys Identified Mail (DKIM): Using Digital Signatures for Domain Verification" [5] is discussed about the Domain Keys Identified Mail (DKIM) defines a mechanism for using digital signatures on email at the domain level, allowing the receiving domain to confirm that mail came from the domain it claims to. The use of Domain Keys Identified Mail (DKIM) signatures and signing practices gives sending domains one tool to help recipients identify legitimate messages from their domain, and a reliable identifier that can be used to combat spam.

2.4 Security Measurement

In all online based companies or every information based company, the security of information system is a great issue for quality and management assurance. To reduce the damage and disclosure ensuring information security is a must thing to do. The main theme of the paper "Security Metrics" [6] some models about IT management. The author instructed with the idea of four security management models. The External audit and the Internal auditory model function ensure that the information system of the organizations is being assessed the risk of security and following a strategy to deal with it. Internal and external, both auditory system model collects the measurement of activity of the management. Capability maturity model measures the improvement the process of security and monitor the stages of development. The risk analysis model is about the cost and security relation based model and Defect elimination is about the security parameter measurement model.

This model is explained with two approaches that are Investigative and Automatic, with the comparison. These approaches are also used to detect the security defects like corruption and intrusion according to the author.

2.5 Elliptic Curve Cryptography

In the paper, "Implementation of Elliptic Curve Digital Signature Algorithm" [7], the researchers focused on the cryptosystem called The Elliptic Curve Digital Signature Algorithm

(ECDSA) which was accepted in 1999 as an ANSI standard. Basically it is now most popular and secure cryptosystem that is even used as an alternative for RSA.

The digital Signature cryptography scheme should provide data authenticity, data integrity, should be un-forgeable in case of attack situation and Elliptic Curve Digital Signature Algorithm (ECDSA) is ideal for the required schemes. With same key size ECDSA provide stronger security than RSA and DSA. In this journal, researchers give very basic knowledge elaborately and also the implementation advantages of ECDSA. They also give a brief and clear comparison about ECDSA, RSA and DSA (Digital Signature Algorithm). As RSA takes sub-exponential time and ECC take full exponential to solve, the ECC curve based system is safer and un-forgeable. So, it requires less key size and takes less time to respond with strong security.

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce. The paper "Digital Signature Standard (DSS)" (FIPS 186-4) [8] focused about the standard of all this cryptographic scheme that are DSA (Digital Signature Algorithm), RSA based DSA and ECC based DSA.

A digital signature standard is a suite of algorithm to generate digital signatory system to provide the data authentications. This signature system verifies the modification of signatures and secure the web system from any unauthorized access. ECC curve cryptography and RSA can be used in this DSA system and are elaborately explained tin this paper.

Elliptic curve cryptography is modern era challenges survived cryptosystem which is more dynamic and provide better securities. ECC is lower power consumed computation which makes this cryptosystem more useable for mobile devices as well as good for securing web servers. In the paper titled as "Speeding up Secure Web Transactions Using Elliptic Curve Cryptography" [9], the researchers mainly focused on the use of ECC with SSL (Secure Sockets Layer (SSL) protocol. They showed the result about the improvised performance of the ECC enhanced version of Open SSL on Apache web server.

In this paper, the comparative analysis shows that ECC has the best security handling HTTPS request per second over RSA (Ron Rivest, Adi Shamir, and Leonard Adleman) based security encryption. As the demand of quick responses and securing the privacy of customers, increasing day by day, the researchers offered ECC based SSL cryptosystem that will meet up the demands reducing process time with a short key size security.

In the paper named by "Research and Application of Digital Signature Based on Elliptic Curve Cryptosystem" [10], the researchers focused on the digital signature problem solving using elliptic curve cryptography. They also have analyzed the achievement of fast point multiplication algorithms for elliptic curve and the establishment of the ElGamal model. High efficient digital signature scheme has also been presented here.

In the paper titled as, "Implementation of Elliptic Curve Digital Signature Algorithm (ECDSA)" [11], the researchers mainly focused on the architecture of the Elliptic Curve Digital Signature Algorithm (ECDSA) in FPGA circuit. The architecture was basically designed to improve FPGA circuit by reaching high performance. The security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem. Some benefits of having smaller key size include faster computation time and reduction in processing power and storage space has also been shown here.

In the paper named by, "Elliptic Curve Cryptosystems", [12] the author discussed about Elliptic curve cryptosystems. These elliptic curve based public key cryptosystems use a multiplicative group of a finite field. These ECC (Elliptic Curve Cryptosystems) is more reliable, secure and difficult to break through as their difficulties to solve are higher than the classical discrete logarithm problem. In this paper the author explained some mathematical operations and approached a theorem about the probability that the order of a cyclic subgroup is non-smooth.

In the paper titled as "Use of Elliptic Curves in Cryptography" [13], the researcher mainly focused on the use of elliptic curves in cryptography and proposed an analogue of the Diffie-Hellmann key exchange protocol. The protocol appears to be immune from infeasible attacks. In this theory, as computational power grows, this disparity should get rapidly bigger.

2.6 Integrated Access Control and Intrusion Detection

Many traditional web security has limited control over an intrusion or spoofed attack to the servers. Some web only takes requests from the trusted users and provide access. These servers do not observe suspicious activities or build a modified protection system according to attack situations. In this paper named by "Integrated Access Control and Intrusion Detection for Web Servers" [14], a generic authorization framework is proposed to call Generic Authorization and Access Control Apache Integration (GAA-API). It detects the intrusion situations and provide security policies to modify the system to protect the resource. The GAA-API is designed in response effectively to the suspected intrusion in real-time before it causes damage. The major advantages of this system are explained in this paper. The GAA-API ensures some routine observing actions like logging information, notifying the administrator, evaluation of the system and make decisions of whether a request should be accepted or rejected. Moreover, it also provides a policy evaluation and can be used by different applications.

In the paper titled as “A Survey and Vital Analysis of Various States of the Art Solutions for Web Application Security” [15], the researchers mainly focused on the security vulnerabilities and dangerous security flaw. They have worked on protecting the confidentiality, integrity and to avail the resources in cyberspace. They have surveyed in the area of web application security, the characteristics of some of the best known techniques of the art solutions for web application security and their weakness.

Chapter 3: Implementation of various methods

3.1 Implementation procedure

In this chapter, the implementation of three popular digital signature scheme is presented: ElGamal, RSA (that currently used by S/MIME) and ECDSA as a security check part of an email application test and compare the performance and characteristics of these schemes.

Most of high level web applications are written in PHP. So, in this thesis paper PHP language is handled to implement these schemes. To analyze performance results simulation tools available on Netbears software are used.

The PHP application includes a Test Suite, along with a class encapsulating NIST published safe curves at various key lengths. NIST Curve class is used as an encapsulation of static methods that contains all NIST published elliptic curves that are approved as secure for elliptic curve cryptography [13] [14].

The key generation process generates the public and private keys in pairs in figure 01. This step in S/MIME in one of the responsibilities of CA. However, in our thesis, we just want to implement different schemes with a variety of hashes to compare their effectiveness as a key part of an email security structure.

If required, the keys can be viewed after generation. This application uses MD5, SHA-1, and SHA-256 and SHA-512 hash algorithms for the digital signature. SHA-256 and SHA-512 hash functions are more secure than the other ones. However, SHA-512 needs more computational resources and in this thesis, we have tried to analyze that, whether it is good to use this secure function or it is not economical.

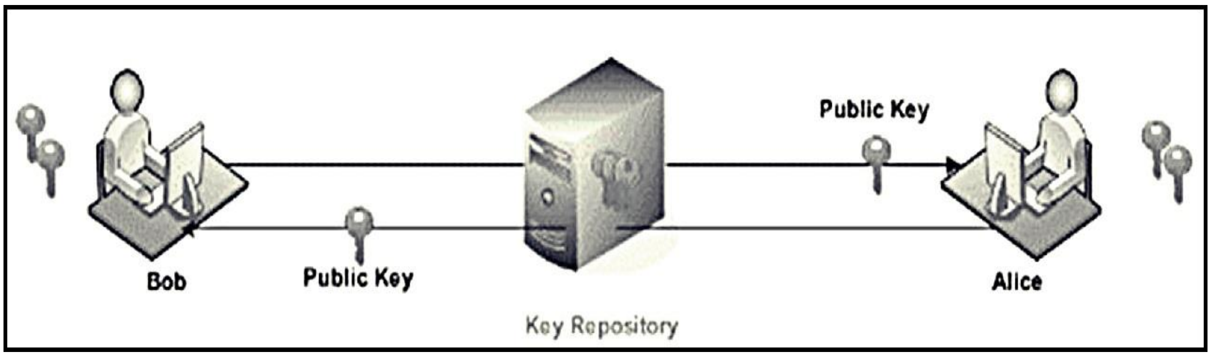
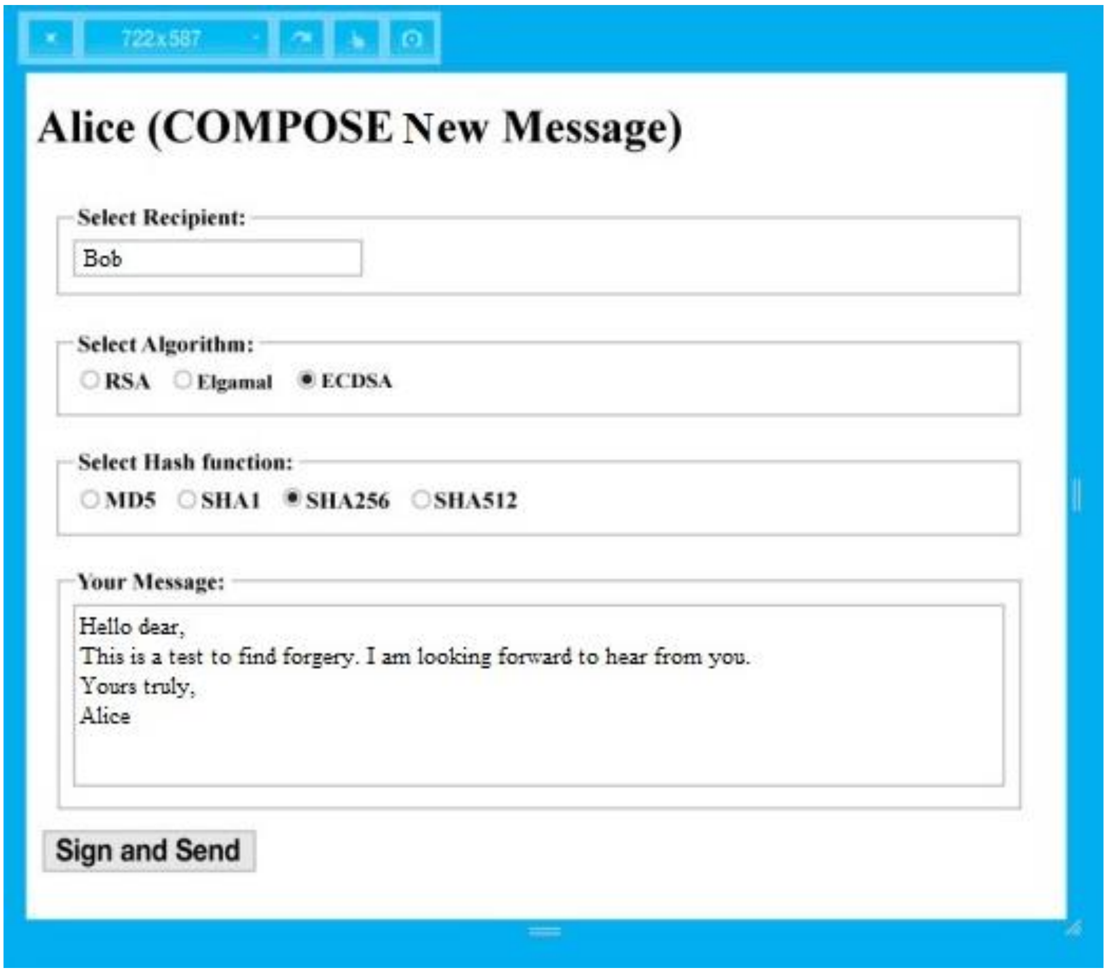


Figure 01: Key Generation Phase

After key generation, the Signature Generation module is processed when Alice (Sender) wants to send a message to Bob (Receiver) which is supposed to be signed by the sender and sent to the recipient in figure 02.



Alice (COMPOSE New Message)

Select Recipient: Bob

Select Algorithm: ○ RSA ○ Elgamal ● ECDSA

Select Hash function: ○ MD5 ○ SHA1 ● SHA256 ○ SHA512

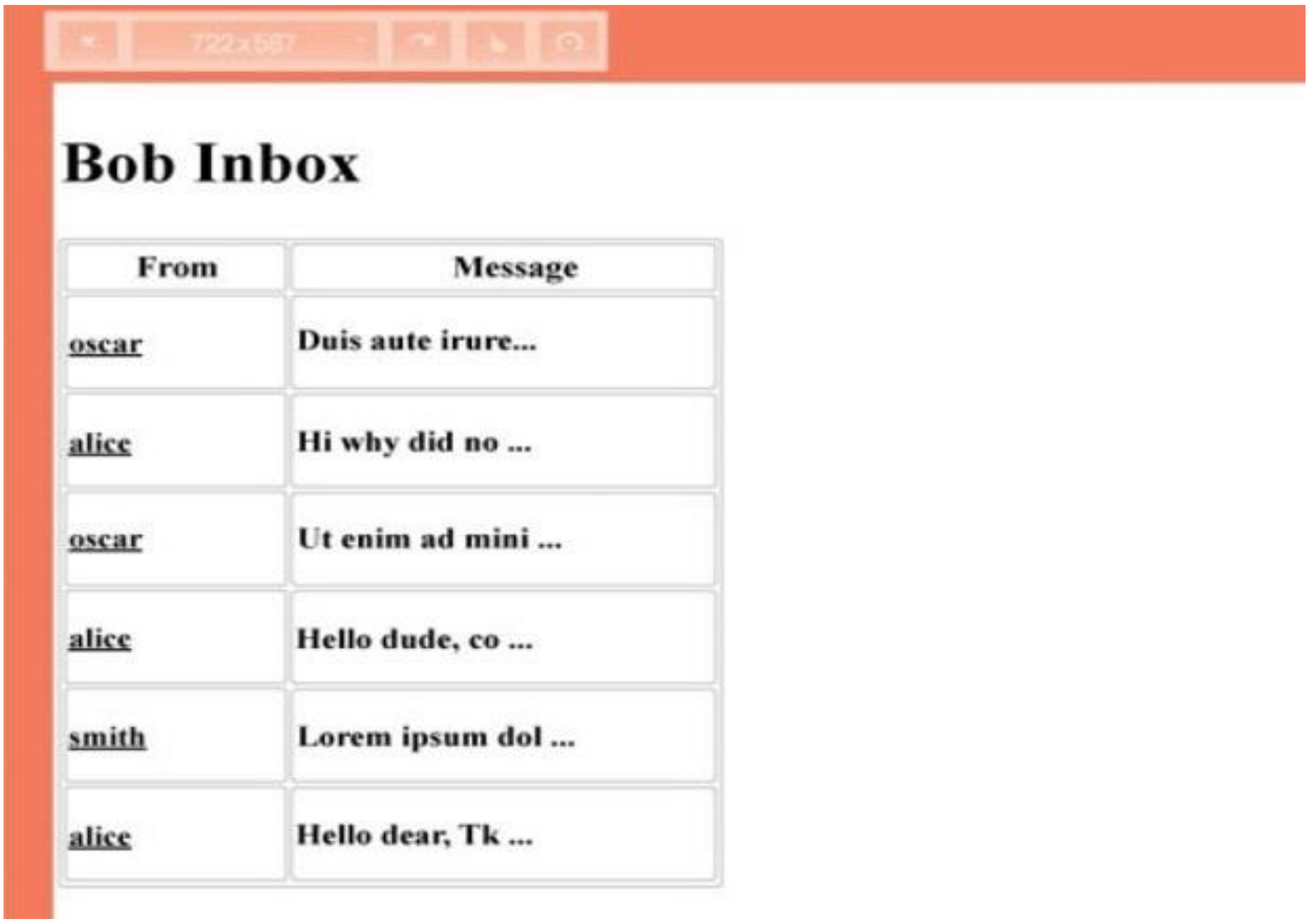
Your Message:

Hello dear,
This is a test to find forgery. I am looking forward to hear from you.
Yours truly,
Alice

Figure 02: Signature Generation

Further, at the receiving time, the recipient (Bob inbox) verifies the signature by execution of the signature verification process to authenticate that if the message has been sent by the authentic sender and to validate that the message has been tampered or not. The figure 04 and 05 shows the execution of the developed application.

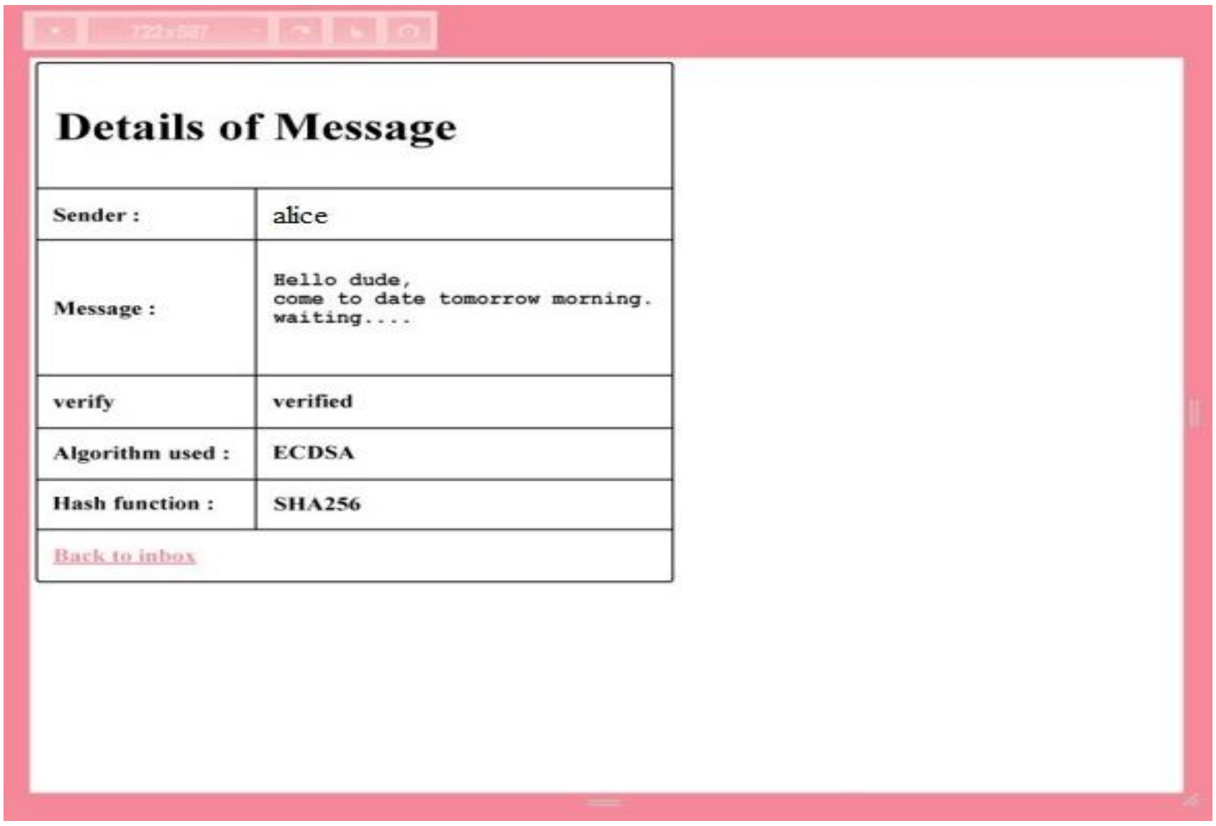
If Bob checks his inbox, he can see that recent Alice's message (figure 3). Now if this message is sent with no forgery, he can find a verify status in his details of message in figure-04.



Bob Inbox

From	Message
oscar	Duis aute irure...
alice	Hi why did no ...
oscar	Ut enim ad mini ...
alice	Hello dude, co ...
smith	Lorem ipsum dol ...
alice	Hello dear, Tk ...

Figure 03: Inbox of Bob



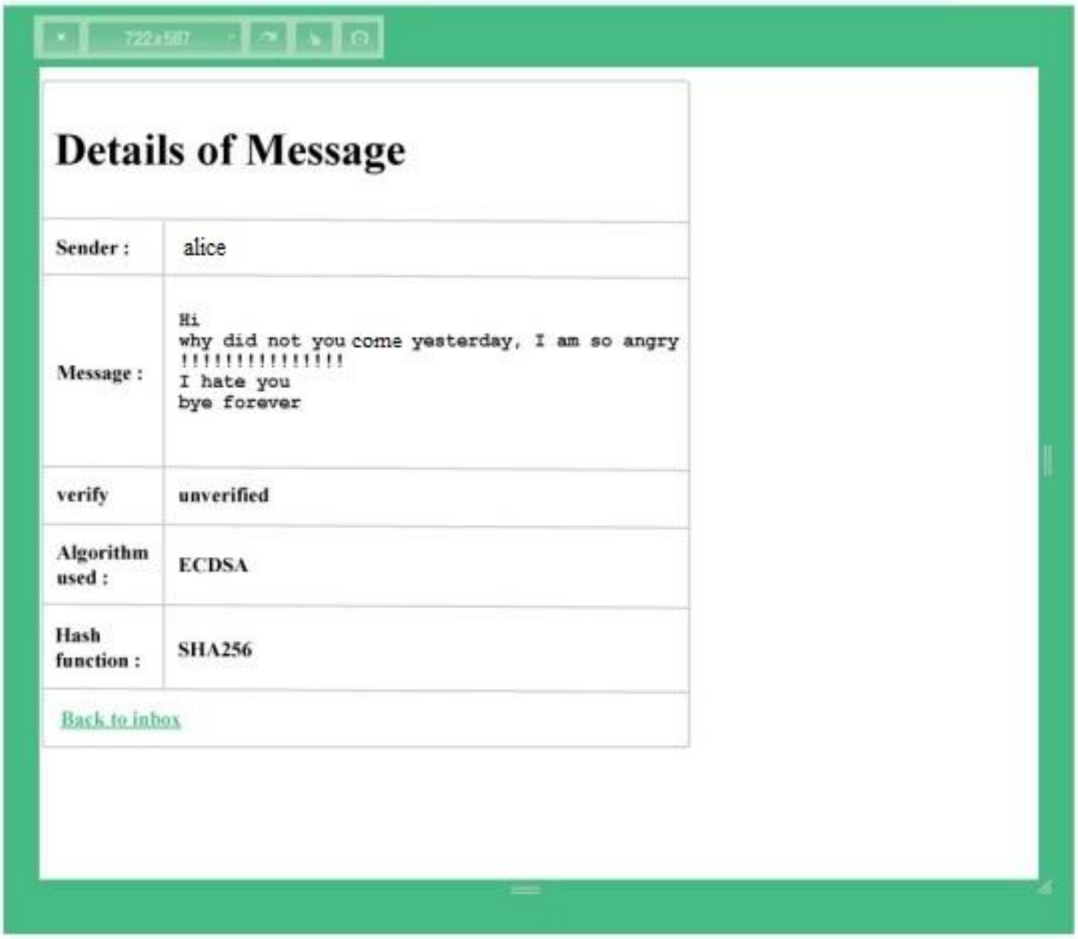
Details of Message

Sender :	alice
Message :	Hello dude, come to date tomorrow morning. waiting....
verify	verified
Algorithm used :	ECDSA
Hash function :	SHA256

Back to inbox

Figure 04: Details of Selected message

However, if Oscar (Attacker) tries to alter Alice's message in between, then Bob can find out that this message is not from Alice with an unverified flag in figure 05.



Details of Message

Sender :	alice
Message :	Hi why did not you come yesterday, I am so angry !!!!!!!!!!!!!!! I hate you bye forever
verify	unverified
Algorithm used :	ECDSA
Hash function :	SHA256

Back to inbox

Figure 05: Details of altered message by attacker (Oscar)

Chapter 4: Result and Analysis

4.1 Test Environment

To test efficiency of the secure communication, two nodes in the Local Area Network (LAN) are established. Two nodes running on two computers are set up. These two computers are running 64 bit Windows 8.1, with the hardware environment (Intel Core i3-2120 CPU 330 GHz RAM-4.00 GB). An application is running on the PHP platform.

4.2 Results

The test results of the developed web application for signature generation and signature verification using MD5, SHA-1, SHA-256 and SHA-512 are tabulated in microseconds as shown in tables 1 to 3.In table 1, we can see the time needed by the hash functions.

Table 1: Time needed for Hash function

Hash Function	Time Needed (microseconds)
MD5	0.000941
SHA-1	0.001264
SHA-256	0.002611
SHA-512	0.0004491

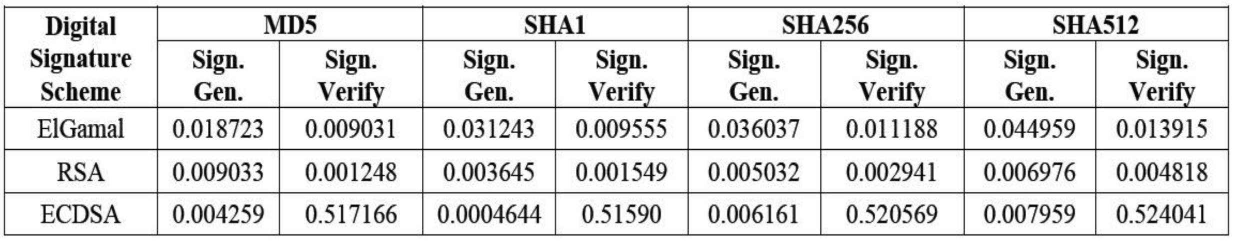
Table -2 shows the comparison of ECC with RSA, DSA (ElGamal) and DH in terms of key length and time to break on machine running 1 MIPS [7] [16]. Due to this table, we select key length 256 for elliptic curve and 3072 for our RSA and ElGamal schemes to analyze in a same security manner. We select a text with size 344 KB for all the experiments.

Table 2: Key comparison of symmetric, RSA/DSA/DH, ECC

Symmetric	RSA/DSA/DH (Bits)	ECC (Bits)
80	1024	160
112	2048	224
128	3072	256
192	8192	384
256	15360	512

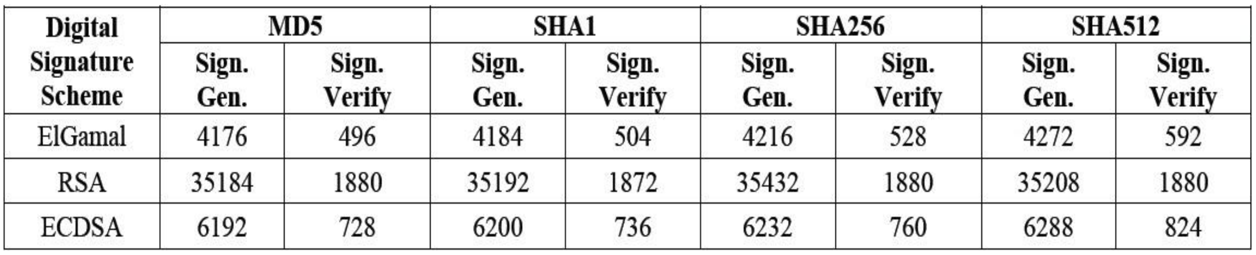
In table 3 and table 4, we have done some experiments for all the schemes and phases for testing both CPU and memory consumption.

Table 3: Experimental results of CPU usage in microseconds



Digital Signature Scheme	MD5		SHA1		SHA256		SHA512	
	Sign. Gen.	Sign. Verify	Sign. Gen.	Sign. Verify	Sign. Gen.	Sign. Verify	Sign. Gen.	Sign. Verify
ElGamal	0.018723	0.009031	0.031243	0.009555	0.036037	0.011188	0.044959	0.013915
RSA	0.009033	0.001248	0.003645	0.001549	0.005032	0.002941	0.006976	0.004818
ECDSA	0.004259	0.517166	0.0004644	0.51590	0.006161	0.520569	0.007959	0.524041

Table 4: Experimental results of Memory usage in Bytes



Digital Signature Scheme	MD5		SHA1		SHA256		SHA512	
	Sign. Gen.	Sign. Verify	Sign. Gen.	Sign. Verify	Sign. Gen.	Sign. Verify	Sign. Gen.	Sign. Verify
ElGamal	4176	496	4184	504	4216	528	4272	592
RSA	35184	1880	35192	1872	35432	1880	35208	1880
ECDSA	6192	728	6200	736	6232	760	6288	824

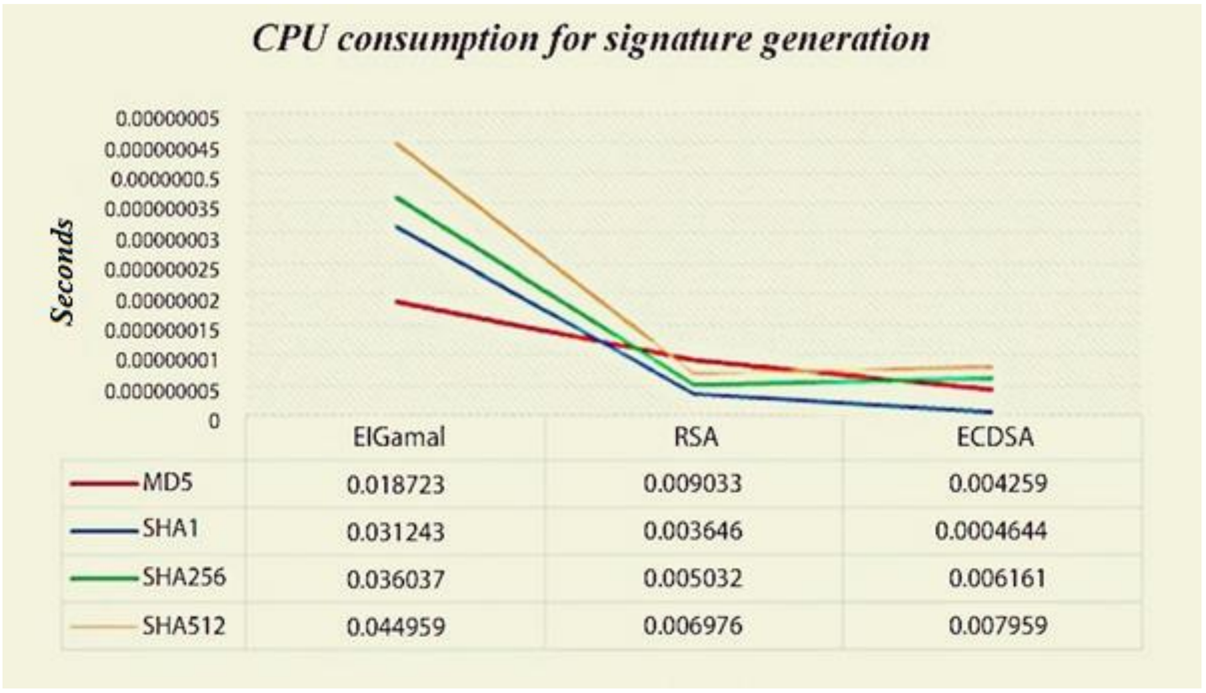
4.3 Discussion

It is generally thought that the security of the 256 bits ECC is equal to the 3072 bits RSA which is used by S/MIME and ElGamal cryptography. The arithmetic speed is faster, short scale of secret key, fast realizing speed & high security are the main advantages of ECC.

It is especially caught on in the situation such as weak calculating capacity, limited integrated circuit space, limited broadband and the high speed realizing case [17]. However, in this thesis, we have done an implementation with some popular digital signature schemes for web application to recognize the detail difference between them.

It can be clearly seen in figure 06, that the amount of CPU consumption in RSA for both signature generation and verification is better than the other scheme.

After that ECDSA and ElGamal schemes are competed. Though, ECDSA is faster in signature generation, but in verification ElGamal is better between ECDSA and ElGamal, as we can see in figure 6 and 7.



	ElGamal	RSA	ECDSA
MD5	0.018723	0.009033	0.004259
SHA1	0.031243	0.003646	0.0004644
SHA256	0.036037	0.005032	0.006161
SHA512	0.044959	0.006976	0.007959

Figure 6: CPU usage of various signature schemes during signature generation

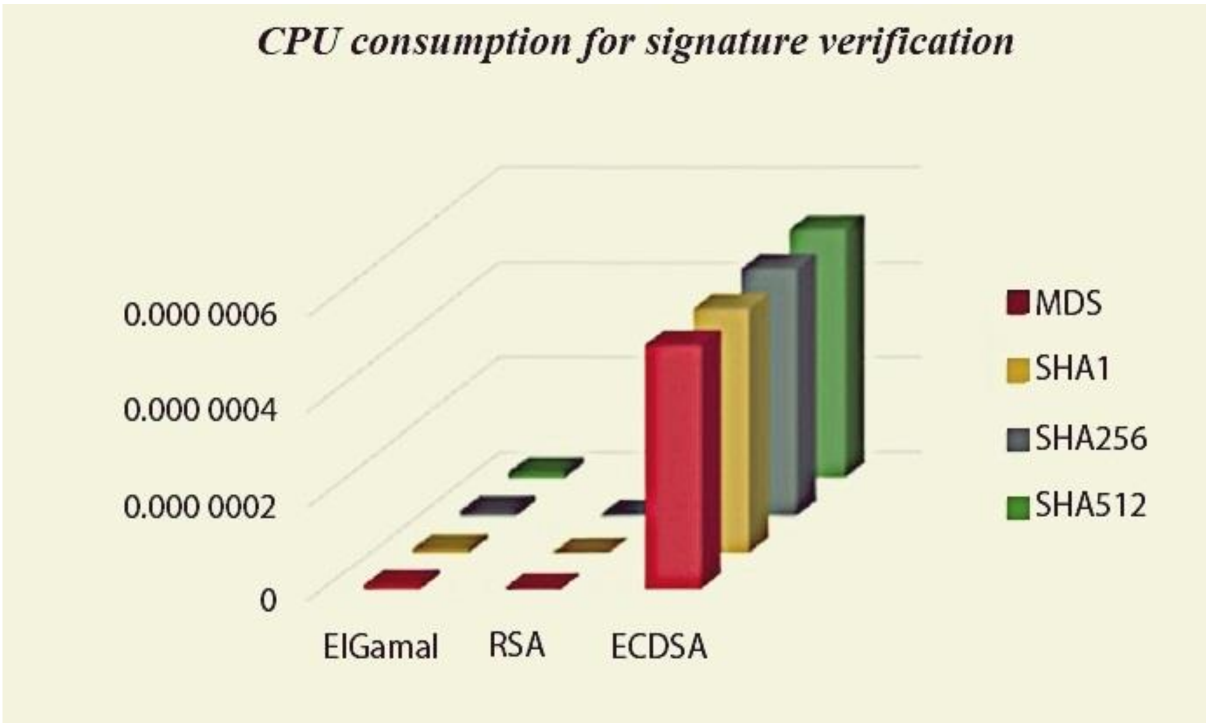
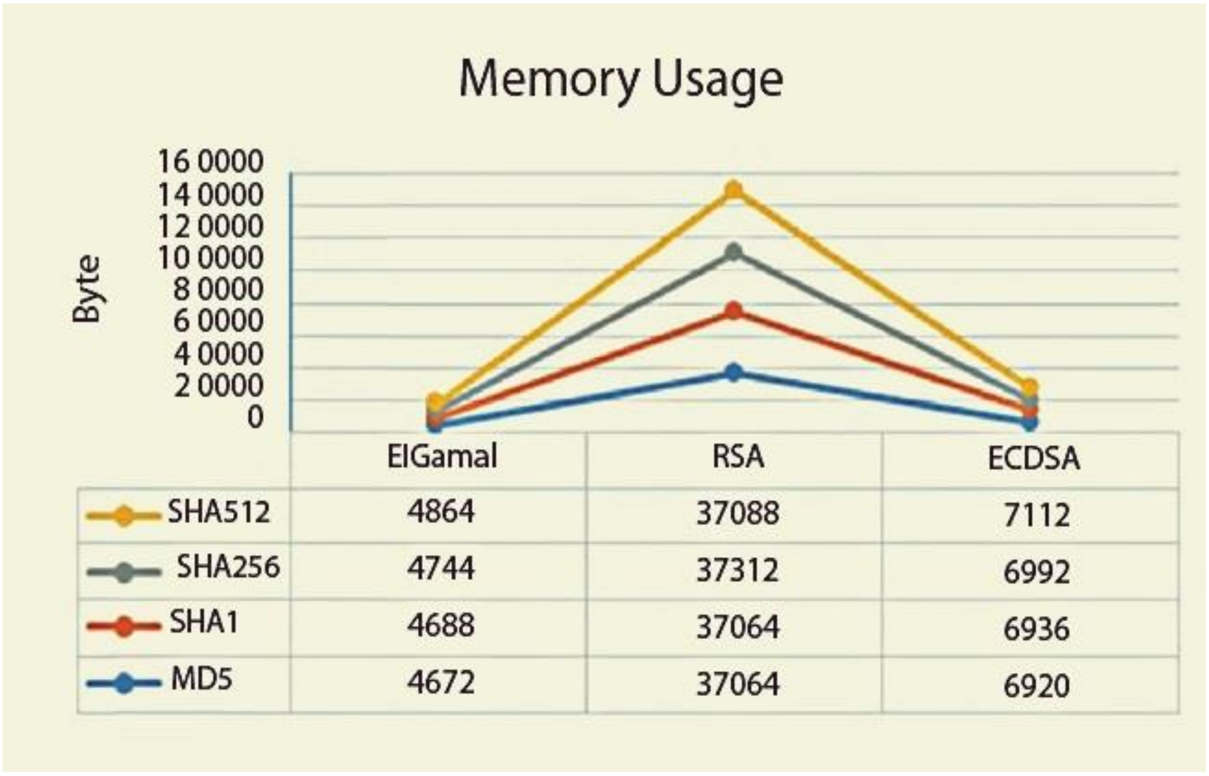


Figure 7: CPU usage of various signature schemes during signature verification

At the same time, when we want to consider another resource, meaning memory consumption, due to our result of analysis, we can see that this time RSA is the worst scheme with a sharp distance with the other two. Now ElGamal performance is the best and ECDSA has the second grade, as we can see in figure 08.

In this figure -08, memory usage of both signature generation and verification are added from table-4 and shown for diffirent hash functions, they are cumulative at each point to avoid overlaping as the values are relatively close to each other.



	ElGamal	RSA	ECDSA
SHA512	4864	37088	7112
SHA256	4744	37312	6992
SHA1	4688	37064	6936
MD5	4672	37064	6920

Figure 8: Memory usage of various signature schemes

Another factor that we can consider for comparing among these schemes is the length of public key where ECDSA has the shortest length which causes lowest overload in network traffics [8]. It is shown in table 5.

4.4 Thesis Findings

Overall, if we want to decide the best choice of digital signature schemes for web application, we need to summarize the key points:

1. RSA and ElGamal take sub exponential time and ECDSA takes full exponential time.

2. ECDSA offers same level of security with smaller key sizes but data size for RSA is smaller than ECDSA and still faster than ECC with same strength.

3. ECC key size is relatively smaller than RSA and ElGamal key size, thus encrypted message in ECC is smaller.

4. At the average, computational power is smaller for ECDSA.

5. ECDSA provides effective and compact implementations for cryptographic operations requiring smaller chips and due to smaller chips, less heat generation and less power consumption will be the consequences. Therefore, ECDSA has easier hardware implementations.

6. One key point is trust because RSA has been around longer than EC, and people feel they understand it and trust it more. Also, it is easier to implement.

7. And the last important key is length of public key which is shortest in ECDSA and therefore takes much shorter time to proceed.

So, RSA is faster and easier to implement in front of complicated mathematic rules in ECC but ECC is more secure and the key length is much shorter. However, the security level in ECC is quite high and infeasible to break up to know. Email application according to their security needs, could attend any of these schemes which can be implemented and verified in quite details.

Chapter-5: Conclusion

This thesis paper presents a security suggestion in email application using S/MIME. So, we have implemented three popular digital signature schemes with different hash functions and measured their performance in terms of time and memory.

Our experiments show that, ECDSA, which is one of the variants of Elliptic Curve Cryptography (ECC) proposed as an alternative to established public key systems which can be a good choice for this reason.

The main point for the attractiveness of ECDSA is the fact that there is no sub exponential algorithm known to solve the elliptic curve discrete logarithm problem on a properly chosen elliptic curve. Hence, it takes full exponential time to solve while the best algorithm known for solving the underlying integer factorization for RSA which is used by S/MIME and discrete logarithm problem in ElGamal, where both took sub exponential time.

The key generated by the implementation is highly secured and it consumes lesser bandwidth because of small key size used by the elliptic curves 256 in front of 3072.

There, it seems that if email application wants to be more secure, ECDSA is the best choice in unreliable environment like wireless transmission. Combination of S/MIME and elliptic curve would guarantee the security in authentication process.

Reference

[1] M. Toorani, “SMEmail - a new protocol for the secure e-mail in mobile environments," in Proceedings of the IEEE Australian Telecommunications Networks and Applications Conference (ATNAC'08), pp. 39{44, 2008.

[2] RFC 5751: Secure/Multipurpose Internet Mail Extensions(s/mime) Version 3.2 Message Specification, [https;//tools.ietf.org.html/rfc5751]

[3] C. Morris and S. Smith, “Towards usefully Secure E-mail,” IEEE technology and Society Magazine, 26(1), 25-34(2007).

[4] A. Kapadia, “A Case (Study) For Usability in Secure Email Communication”, IEEE Security and Privacy, 5(2), 80-84, 2007

[5] B. Leica and J. Fento, “Domain Keys Identified Mail (DKIM): Using Digital Signatures for Domain Verification”, CEAS 2007, Fourth Conference on E-mail and Anti-Spam August 2-3, 2007, Mountain View, California USA.

[6] J. Bayou, “Information security metrics-an audit based approach. “Computer System Security and Privacy Advisory Board (CSSPAB) Workshop on Security Metrics.2000

[7] A. Khalique, K. Singh and S. Sood, “Implementation of Elliptic Curve Digital Signature Algorithm” International Journal of Computer Applications 2.2 (2010): pp. -21-27.

[8] “Digital Signature Standard (DSS)” Technical Report FIPS 186-4, National Institute of Standards and Technology, July 2013.

[9] V. Gupta, D. Stebila and S. Fung, S. C. Shantz, N. Gura, & H. Eberle, “Speeding up Secure Web Transactions Using Elliptic Curve Cryptography” In proceedings of the 11th Annual Network and distributed system security symposium. The internet Society NDSS.2004, 231-239.

[10] Y. C. Wang, H. X. Fang and Y. Xia, “Research and Application of Digital Signature Based on Elliptic Curve Cryptosystem” Advances in Biomedical Engineering 8 (2012):236.

[11] A. Abidi, B. Boullegue and F. Kahri, “Implementation of Elliptic Curve Digital Signature Algorithm (ECDSA)”. Computer & Information Technology (GSCIT), 2014 Global Summit on IEEE2014.

[12] N. Koblitz, “Elliptic Curve Cryptosystems”, Mathematics of Computation vol.48, pp.203-209, 1987.

[13] V. S. Miller, “Use of Elliptic Curves in Cryptography”. Advances in cryptology - CRYPTO’85 Proceedings. Springer Berlin/ Heidelberg 1986.

[14] T. Ryutov, C. Neuman, K. D. Kim and Z. Li, “Integrated Access Control and Intrusion Detection for Web Servers”, IEEE Transactions on Parallel and Distributed Systems 14(9) (2003) 841-850.

[15] A. Thankachan, R. Ramakrishnan and M. Kalaiarasi, “A Survey and Vital Analysis of Various State of the Art Solutions for Web Application Security” Information Communication and embedded Systems (ICICES), 2014 International Conference on IEEE, 2014

[16] D. Hankerson, A. J. Meekness and S. Vanstone, “Guide to elliptic curve cryptography”. ISBN 0-387-95273-X.

[17] C. Paar and J. Pelzl, “Understanding Cryptography”. DOI 10.1007/978-3-642-04101-3.

[18] J. Callas, “OpenPGP Message Format”, IETF Proposed Standard RFC 4880. [Online]. Available: https://tools.ietf.org/html/rfc4880

APPENDIX-I

Source Code written in JAVA

•Sign the message

Next we have to write our message and then sign it. The message and the signature can be separate files but in our example we add them to a List of byte[] and write them as Object to the file.

Message.java

```java
package com.mkyong.sender;


import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.nio.file.Files;

import java.security.InvalidKeyException;

import java.security.KeyFactory;

import java.security.PrivateKey;

import java.security.Signature;

import java.security.spec.PKCS8EncodedKeySpec;

import java.util.ArrayList;

import java.util.List;


import javax.swing.JOptionPane;


public class Message {
        private List<byte[]> list;



        //The constructor of Message class builds the list that will be written
to the file.

        //The list consists of the message and the signature.

        public Message(String data, String keyFile) throws InvalidKeyException,
Exception {

                list = new ArrayList<byte[]>();

                list.add(data.getBytes());

                list.add(sign(data, keyFile));

        }



        //The method that signs the data using the private key that is stored in
keyFile path

        public byte[] sign(String data, String keyFile) throws
InvalidKeyException, Exception{

                Signature rsa = Signature.getInstance("SHA1withRSA");

                rsa.initSign(getPrivate(keyFile));

                rsa.update(data.getBytes());

                return rsa.sign();

        }



        //Method to retrieve the Private Key from a file

        public PrivateKey getPrivate(String filename) throws Exception {

                byte[] keyBytes = Files.readAllBytes(new
File(filename).toPath());

                PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keyBytes);

                KeyFactory kf = KeyFactory.getInstance("RSA");

                return kf.generatePrivate(spec);

        }
        //Method to write the List of byte[] to a file

        private void writeToFile(String filename) throws FileNotFoundException,
IOException {

                File f = new File(filename);

                f.getParentFile().mkdirs();

                ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(filename));

            out.writeObject(list);

                out.close();

                System.out.println("Your file is ready.");

        }



        public static void main(String[] args) throws InvalidKeyException,
IOException, Exception{

                String data = JOptionPane.showInputDialog("Type your message
here");



                new Message(data,
"MyKeys/privateKey").writeToFile("MyData/SignedData.txt");

        }
```

•Verify the Signature

The receiver has the file *(he knows it is a List of 2 byte arrays; the message and the signature)* and wants to verify that the message comes from the expected source with a pre-shared Public Key.

VerifyMessage.java

```
package com.mkyong.receiver;
```

```
import java.io.File;

import java.io.FileInputStream;

import java.io.ObjectInputStream;

import java.nio.file.Files;

import java.security.KeyFactory;

import java.security.PublicKey;

import java.security.Signature;

import java.security.spec.X509EncodedKeySpec;

import java.util.List;


public class VerifyMessage {

        private List<byte[]> list;


        @SuppressWarnings("unchecked")

        //The constructor of VerifyMessage class retrieves the byte arrays from
the File

        //and prints the message only if the signature is verified.

        public VerifyMessage(String filename, String keyFile) throws Exception {

                ObjectInputStream in = new ObjectInputStream(new
FileInputStream(filename));

            this.list = (List<byte[]>) in.readObject();

            in.close();


            System.out.println(verifySignature(list.get(0), list.get(1), keyFile)
? "VERIFIED MESSAGE" +

              "\n----------------\n" + new String(list.get(0)) : "Could not
verify the signature.");

        }
        //Method for signature verification that initializes with the Public Key,

        //updates the data to be verified and then verifies them using the
signature

        private boolean verifySignature(byte[] data, byte[] signature, String
keyFile) throws Exception {

                Signature sig = Signature.getInstance("SHA1withRSA");

                sig.initVerify(getPublic(keyFile));

                sig.update(data);



                return sig.verify(signature);

        }



        //Method to retrieve the Public Key from a file

        public PublicKey getPublic(String filename) throws Exception {

                byte[] keyBytes = Files.readAllBytes(new
File(filename).toPath());

                X509EncodedKeySpec spec = new X509EncodedKeySpec(keyBytes);

                KeyFactory kf = KeyFactory.getInstance("RSA");

                return kf.generatePublic(spec);

        }



        public static void main(String[] args) throws Exception{

                new VerifyMessage("MyData/SignedData.txt", "MyKeys/publicKey");

        }

}
```