CHAPTER 1
INTRODUCTION

1.1 Introduction

At the end of 2007, there were over 3 billion GSM (Global Service for Mobile communication) users all over the world. This would stand for half of the world people if there was one partnership per person, and nearly 7 billion text messages are sent every day (GSM Europe, 2007). Therefore, the most used data service is SMS (Short Message Services), which is used mainly for personal communication, but it has also been used in many applications where the other party is an information system like mobile commerce.

SMS messages are sent from a mobile device to a message centre and the message center stores it until it can be delivered to the destination. This means the message travels unencrypted in the operators network. SMS messaging routines do not facilitate end-to-end encryption. So, there is a demand to solve the confidentiality issues of messages traveling within the operator network to communicate or exchange confidential information in a secure environment.

Many applications and systems have been developed to solve SMS security issues using either symmetric or asymmetric key cryptography. In symmetric solutions the encryption is based on a shared secret password (secret key) between the sender and the recipient. Although it is simple and fast, there is a high possibility that the key will be exposed to the public, and it may have a problem on key distribution and thus decreases the reliability of the service.

Symmetric cryptography is considered quite fast but has a strict problem which is the process of transferring the keys to the recipient is insecure and vulnerable to the security risks (Choudhury et al., 2002). The key exchange process should rely on a very high level
of trust between the communications partners. Therefore, if the secret key (shared password) is intercepted by an attacker, he can decrypt all the messages sent using that key. Although asymmetric cryptography (public key cryptography) is very slow, it solves the problem in symmetric cryptography by using two different keys: a public key for encryption, and a private key for decryption. The private key is not transferred and kept securely (Claret, 2006). Public key cryptography does not only solve the problem of key distribution but also simplifies the process of key management, and provides the security services including confidentiality, data integrity, authentication, and non-repudiation (Chadhoury et al., 2002).

Although asymmetric key cryptography is efficient in complex and huge networks but long execution time prevents the public key cryptography from being commonly deployed onto low-power wireless device (Shim and Lee, 2005).

Neal Koblitz and Victor Miller (1985) separately introduced the Elliptic Curve Cryptosystem (ECC) as another alternative Public Key Cryptosystem (PKC). Its security depends on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP) that presents the most secure public-key system. The elliptic curve cryptography provides the same level of cryptographic security as DSA (Digital Signature Algorithm) or RSA (Rivest Shamir Adleman) using smaller key sizes, which result in smaller system parameters, smaller public-key certificates, bandwidth savings, faster implementations, lower power requirements, and smaller hardware processors (Jurisic and Menezes, 2005).

The vital component in the public key cryptography is the certificate, which is a data structure that binds a public key to an entity in an authentic way and signed by a trusted third party among other information. The management of certificates during their life cycle in an administrative domain requires an infrastructure, namely Public Key Infrastructure (PKI) (Dankers et al., 2002).
1.2 Problem Statement

Currently the short message services not providing end-to-end security, it provide only point-to-point of some security services, encrypt the message during the transmission between the mobile phone and operator network. Besides, the protocol used to transmit the message through the operator network is unencrypted protocol, the result of which is that anyone who has access to the operator network can have the ability to read and/or modify the messages where mobile users cannot communicate securely, and exchange secret and confidential information using SMS.

Due to the limitation of mobile device especially that related to the processing power prevents the public key cryptography from being widely deployed onto the mobile devices. Nowadays the existing solutions to solve SMS security issues are using either symmetric or asymmetric cryptography. Although symmetric cryptography is simple and quit fast, it has a problem of transferring the key (shared password) to recipient. The asymmetric cryptography (public key cryptography) solutions are based on small key size to achieve an acceptable performance on the mobile device, which decrease the security level in those systems.

In this evolving situation, a Public Key Infrastructure (PKI) using the elliptic curve cryptography became a proven solution for mobile phone SMS since it provides the security services required for information (confidentiality, integrity, authentication, non-repudiation), and solves the processing power limitations in the mobile phone.

1.3 Research Objectives

   Due to problems encountered in traditional messaging service, and the limitations of mobile device, the study seeks to realize the following objectives:

   • To study the current implementations of SMS security.
To enhance the security level of SMS by implementing an end-to-end secure application that is capable of sending a private and confidential SMS

To apply the public key infrastructure in the phone using the Elliptic Curve Cryptosystem (ECC).

1.4 Research Scope

The focus of this research is to apply the public key infrastructure in the mobile phone by implementing the elliptic curve cryptography in the mobile phone to provide end-to-end secure SMS messaging. The system provides the four security services i.e. Confidentiality, Integrity, Authentication, and non-repudiation. The software encrypts the SMS using the Elliptic Curve Integrated Encryption Schema (ECIES) to provide confidentiality. Moreover, it uses the Elliptic Curve Digital Signature Algorithm (ECDSA) for signature generation and verification to provide integrity. The system provides the authentication and non-repudiation throughout the Certificate that is issued and signed by the Certificate Authority (CA).

1.5 Thesis Outline

The thesis consists of six chapters all together including the introduction chapter. Each chapter will discuss a certain domain in this thesis research process. Below are the chapter’s numbers, titles and summaries:

- Chapter 1: Introduction
  
  This chapter presents an overview of the research topic. It discusses the main objectives, scope, schedule, and thesis outline of the project.

- Chapter 2: Literature Review
This chapter provides a brief review of some previous related studies in addition to particular techniques that are relevant to the present research. The recent research and technology of mobile messaging are discussed in a consistent way, and some related studies and available suggested solutions for the problem under discussion are reviewed.

- Chapter 3: Research Methodology
  This chapter provides a thorough and detailed view of the approach and technique used in this research to implement the framework.

- Chapter 4: ECSMS Analysis and Design
  The complete project analysis and design are presented in flow charts and diagrams such as use cases, activity, sequence, and class diagrams.

- Chapter 5: ECSMS Implementation and Testing
  This chapter highlights the implementation phase includes NetBeans IDE, algorithms, as well as the pseudo code for the main classes and core interfaces are shown to give better illustration for primary function and the flow of the application. Besides, this chapter will provide testing for the system, which will focus on the usage and performance of the application.

- Chapter 6: Discussion and Conclusion
  This chapter summarizes the work accomplished and discusses possibilities and recommendations for the future.
CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

The secure transport of messages was the concern of many early civilizations. Since then, different methods have been developed to ensure that only the sender and the addressee would be able to read a message, while it would be illegible or without significant meaning to a third party.

Today, this practice continues with more intense interest. Wireless, and optical communication networks are able to transport unbelievable amounts of data and thus privacy of information and security of the network are the greatest concern because the transported information may be very sensitive and/or confidential. Since the transmission media in the mobile communication systems are air interface, security is one of the most important requirements of mobile communication systems to ensure that the service is properly used (Al-Fayoumi et al., 2007).

2.2 Cryptography

The fundamental task in cryptography is to allow users to communicate securely over an insecure channel in such way that secures their transmissions’ privacy and authenticity (Gutmann et al., 2006).

According to Tipton and Krause (2004), cryptography is the science of secure and secret communications which uses the encryption algorithm; transforming the information form plain text into a cipher text, and decryption algorithm; reveres the encryption information to restore the original plain text. The cryptography system or a cryptosystem is formed from the encryption algorithm and a decryption algorithm plus the description on the format of messages and keys (Mao, 2004).
Modern cryptography can be classified into two main types: symmetric and asymmetric cryptography (Mogollon, 2007). What follows is a brief account of both of them.

2.2.1 Symmetric Key Cryptography

Symmetric key cryptography is also known as secret key cryptography (Burnett and Paine, 2001). In a secret key cryptosystems, encryption and decryption use the same key, which implies that the entity that encrypts a message has to share the encryption key with the entity that will receive and decrypt the encrypted message (Mao, 2004). So, everyone can encrypt messages, but only the owner of the private key can read them (Khalifa et al., 2004).

Although symmetric key technique is simple and easy to implement, but it has explicit disadvantages which are as follows (Kapoor and Abraham, 2008):

- Both parties have to agree on a shared secret key.
- If there are $n$ users communicating, then each user must share $n-1$ keys with others.
- Sharing a secret key caused that the authentication cannot be proved.
- Has a problem of managing the symmetric keys.

Numerous symmetric key algorithms were developed. The main generally used secret key algorithms include (Choudhury et al., 2002):

- Data Encryption Standard (DES).
- Triple−DES (3DES).
- International Data Encryption Algorithm (IDEA).
- ARCFOURE (RC4).
- Carlisle Adams and Stafford Tavares (CAST)−128.
- Advanced Encryption Standard (AES).
2.2.2 Asymmetric Key Cryptography

Asymmetric key cryptography is most commonly referred to as public key cryptography. Diffie and Hellman (1976) introduced the concept of public key cryptography via presenting a new idea in cryptography to build a scheme to provide a secret communication. They suggested that encryption and decryption could be done with a pair of different keys rather than with the same key. The decryption key would still have to be kept secret, but the encryption key could be made public without compromising the security of the decryption key.

Public key cryptography permits users to communicate securely using a pair of different keys, public key for encryption and private key for decryption. That pair of keys are not independent but mathematically related, where the data that is encrypted with the public key can only be decrypted with the corresponding private key; on the other hand data encrypted with the private key can only be decrypted with the corresponding public key (Kartalopoulos, 2006).

Diffie and Hellman (1976) also introduced the concept of digital signatures; users can verify that the data have been exchanged have not been modified by performing an authentication operation using the same pair keys. First the sender A encrypts the data with his private key, then encrypts the result with the receiver public key, on the other side the receiver B first decrypts the message with his private key then decrypts the result with A's public key.

2.2.3 The Advantages of Public Key Cryptography

In symmetric cryptography, encryption is quite fast but has a severe problem. The major problem with symmetric cryptography is that the process of transferring keys to the recipient is prone to security risks. In other words, in order to share a secret with others,
they have to know your key. This implies a very high level of trust between people sharing secrets. Therefore, if a dishonest person has your key or if your key is intercepted by a spy, they can decrypt all the messages you send using that key. Although asymmetric encryption is very slow, it solves the trust problem inherent in symmetric encryption by using two different keys: a public key for encrypting messages, and a private key for decrypting messages (Claret, 2006). This private key is not transferred to anyone and is stored securely by the holder of the key and thus public key cryptography eliminates the need for transferring the private key (Choudhury et al., 2002). This makes it possible to communicate in secrecy with people you do not fully trust (Claret, 2006).

Public key cryptography does not only solve the problem of key distribution but also makes the process of key management a lot simpler, and addresses two other cryptography issues. Firstly, authentication is allows someone in the electronic world to confirm data and identities. Secondly, non-repudiation is prevents people from going back on their electronic word, where they use the digital signature to implement these features (Burnett and Paine, 2001).

### 2.3 Public Key Cryptosystems

Many public key cryptographic systems have been developed since the creation of public key cryptography in 1976. Nowadays, only the systems that rely on the following mathematical problems are considered secure and efficient: the Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP), and the Elliptic Curve Discrete Logarithm Problem (ECDLP) (Certicom, 2000).
2.3.1 The Integer Factorization Problem (IFP)

The IFP is the following: “given a number \( n \) that is the product of two large prime numbers \( p \) and \( q \), find \( p \) and \( q \)”. It is easy to find a large prime numbers, but factoring the product of two prime numbers is computationally hard if the primes are warily selected. Based on the difficulty of this problem, RSA public key was developed (Certicom, 2000).

2.3.1.1 RSA Public Key Cryptosystem

Diffie and Hellman (1976) did not identify a method with full public key encryption/decryption properties that would also enable digital signatures. However, they did introduce a specific method based on number theory for establishing secret keys by parties who did not previously share a secret. The method is called Diffie-Hellman key agreement. The security of the method is related to longstanding problem in discrete logarithms.

Rivest, Shamir, and Adleman (RSA) (1978), introduced the first applied scheme which is the most popular public key scheme. The security of the RSA public key scheme is based on the intractability of factoring the integer modulus which is the product of two large and distinct prime numbers. In the RSA public key cryptosystem, each user distributes his encryption key (public key) in the public directory, while keeping the secret key (private key) secret. The encryption \( E \) and decryption \( D \) procedures have the following properties:

1. Decrypting the encrypted message \( M \) produce \( M \)
   \[ D (E (M)) = M \]
2. Both \( E \) and \( D \) are easy to compute.
3. Knowing \( E \) the user can not compute \( D \). So the user can decrypt messages encrypted with \( E \).
4. Decrypting the message \( M \) and then encrypting it, the result is \( M \)
\[ E(D(M)) = M. \]

Two basic facts and one conjecture in number theory prepare the way for today’s RSA public-key cryptosystem:

1. Prime generation is easy: To generate a random prime, one can simply generate random numbers of a given size and test them for primarily until a prime is found.
2. Multiplication is easy: Given \( p \) and \( q \), it's easy to find their product, \( n = pq \).
3. Factoring is hard: given such an \( n = p*q \), it appears to be quite hard to recover the prime factors \( p \) and \( q \). It has been estimated recently that recovering the prime factors of a 1024-bit number would take a year on a machine costing US $10 million.

(a) RAS Encryption, Decryption, and Digital Signature Algorithms

RSA public key cryptosystem was developed on the basis of the difficulty of the IFP, which is the difficulty of solving the integer modulus \( n \) with the help of the public key \( e \) and the integer decrypted message \( c \). which means that solving \( e^{th} \) roots mod a composite modulus \( n \), where the modulus \( n \) and the public key \( e \) are defined to insure that for every integer \( c \in (0,1,...,n-1) \) there is just one \( m \in (0,1,...,n-1) \) where \( m^e = c \mod n \). RSA can be used for encryption and digital signature methods. The RSA scheme is as follows presuming that finding the prime factors of \( n = p * q \) is computationally hard (Aboud et al., 2008):

- Key generation algorithm

  In order for entity A to generate the keys he has to do the following:

  1. Determines two random and secret large prime numbers \( p \) and \( q \).
  2. Computes the modulus \( n = p*q \).
3. Determines $\theta (n) = (p-1)(q-1)$.

4. Chooses random integer $e, 1 < e < n$ where $\gcd(e, \theta) = 1$.

5. Calculates the decrypted key $d, 1 < d < \theta (n)$ where $e^d \equiv 1 \mod \theta (n)$.

6. Defines the public and private key, where the pair $(d, \theta)$ is the private key and the pair $(n, e)$ is the public key.

- Public key encryption and decryption algorithm

Assuming that entity B wants to encrypt message $m$ and send it to entity A. The encryption and decryption processes are as follow:

- Encryption: in order to encrypt the message $m$ entity B do the following:
  - Gets the public key $(n, e)$ for entity A.
  - Describes the message $m$ as an integer in the interval $[0…n-1]$.
  - Calculates $c \equiv E(M) \equiv m^e \mod n$.
  - Send(s) entity A the encrypted message $c$.

- Decryption: In order for entity A to decrypt the decrypted message $c$ and recover the message $m$, entity A should do the following:
  - Gets the decrypted message $c$ from entity B.
  - Computes $D(c) \equiv (c^d \mod n)$ to recapture the message $m$.

- The digital signature algorithm

In order for user B to send a sign message to user A, user B first computes his signature $S$ for the message $M$ using his private key $D_B$: $S = D_B(M)$. Then he encrypts $S$ using user A public key $E_A$, and sends the result $S$ to user A. On the other hand, user A decrypts the signature with his private key $D_A$ to obtain $S$. User A knows who is the sender of the signature, which can be attached to $S$ as plain text. Then he extracts the message with the encryption key of the sender, in this case $B$:
\[ M = E_B (S). \]

So user B cannot deny having sent user A this message, since he is the only one who can create the signature \( S = D_B (M) \), and user A can proof that user B signed this message \( E_B(S) = M \) (Rivest et al., 1978).

(b) Attacks on RSA

After Rivest, Shamir, and Adleman (1978) introduced the RSA cryptosystem the IFP has received some attention towards breaking it up. The Continued Fraction, Quadratic Sieve (QS), and Number Field Sieve (NFS) algorithms are used to solve the RSA system, where quadratic sieve was used to factor 129-decimal digit (429-bit) number, and the number field sieve algorithm was used to factor the 155 decimal digits equal to 513 bits number. On the other hand, a special purpose factoring algorithm was used a 54 digits (150-bit) prime number to factor a 127-digit (422 bits) number namely the Elliptic Curve Factoring Method (ECFM). So RSA cryptosystem consider secure when used a 1024-bit or larger modules (Certicom, 2000).

2.3.2 The Discrete Logarithm Problem (DLP)

The DLP is the following: If \( p \) is a prime number and \( \mathbb{Z}_p \) indicates the set of integers \( \{0, 1, 2 \ldots p-1\} \). There will be a generator \( a \) of \( \mathbb{Z}_p \); a non-zero element \( \in \mathbb{Z}_p \) such that each non-zero element in \( \mathbb{Z}_p \) can be written as a power of \( a \). And a non-zero element \( \beta \in \mathbb{Z}_p \), in such that finding a unique integer \( l, 0 \leq l \leq p-2 \), where \( \beta \equiv a^l \pmod{p} \). The integer \( l \) is called the discrete logarithm of \( \beta \) to the base \( a \) (Certicom, 2000).

Based on the difficulty of DLP, many cryptographic systems and protocols have been proposed which include the Diffie-Hellman key agreement scheme, the ElGamal
encryption and signature schemes, the U.S. government’s DSA, the Schnorr signature scheme, and the Nyberg-Rueppel signature scheme.

### 2.3.2.1 ElGamal Encryption and Signature Scheme

ElGamal (1985) produced one of the effective and useful public key cryptosystems, which depends on the complexity of DLP over finite fields. It is also the basis of many digital signature systems. The ElGama public key cryptosystem scheme works as follows:

- **Key setup (Liu et al., 2006)**

  Entity A generates the public key by:
  1. Generating a large random prime \( p \).
  2. Calculating a random multiplicative generator element \( \alpha \) of \( \mathbb{Z}_p \).
  3. Choosing random integer \( a \) as his private key from \( \mathbb{Z}_p \).
  4. Calculate his public key \( y \equiv \alpha^a \mod p \).
  5. The public key is \( (p, \alpha, y) \), and the private key is \( a \).

- **Public key encryption (Haraty et al., 2005)**

  If entity B wants to encrypt a message \( m \) to entity A, he does the following:
  1. Gets a copy of A’s public key \( (p, \alpha, y) \).
  2. Represents the message \( m \) as an integer in the range \( \{0, 1, \ldots, p-1\} \).
  3. Selects a random integer \( k \), where \( 2 \leq k \leq p-2 \).
  4. Calculates \( \gamma = \alpha^k \mod p \), and \( \delta \equiv m \cdot y^k \mod p \).
  5. Sends the encrypted message \( c = (\gamma, \delta) \) to entity A.

  In order for entity A to decrypt the message \( c \) from B, he does the following:

  Entity A uses his private key \( a \) to determine \( \gamma^{p^1-a} \mod p \), where \( \gamma^{p^1-a} = \gamma^a = \alpha^{ak} \), then recovers the message \( m \) by calculating \( \gamma^a \cdot \delta \mod p \).
Signature scheme

ElGamal’s signature scheme can be described as follows (Liu et al., 2006):

- Signature generation: if entity A want to send a sign message to entity B, he does the following:
  1. Obtains the hash value for the message $m$, $h(m)$.
  2. Generates a random secret integer $k$, with $\gcd(k, p-1) = 1$, where $1 \leq k \leq p-1$.
  3. Computes $r = \alpha^k \pmod{p}$, and $k^{-1} \pmod{(p-1)}$.
  4. Computes $s = k^{-1} \{h(m) - a^* r\} \pmod{(p-1)}$.
  5. Then entity A sends the signature $m$, $(r, s)$.

- Signature verification: in order for entity B to verify the signature $(m, (r, s))$, he does the follows:
  1. Obtains authentic copy of A public key $(p, \alpha, y)$, and verify that $1 \leq r \leq p-1$.
  2. Calculates the hash value for the message $m$, $h(m)$.
  3. Calculates $v_1 = y^r \cdot r^s \pmod{p}$, and $v_2 = \alpha^{h(m)} \pmod{p}$.
  4. Accepts the signature if $v_1 = v_2$.

2.3.2.2 Attacks on ElGamal

Attacking the ElGamal public key scheme means that solving the discrete logarithm problem. LaMacchia and Odlyzko (1990) solved a discrete logarithm problem modulo a 191-bit prime using the Gaussian integer method. Later on Weber and Denny (1998) used the number field sieve algorithm to solve a discrete logarithm problem modulo a 129-digit
special prime from. This specifies that to consider the discrete logarithm cryptosystem is secure, 1024 bits or larger moduli \( p \) should be used.

### 2.3.3 The Elliptic Curve Discrete Logarithm Problem (ECDLP)

The ECDLP is the following: given an elliptic curve \( E \) defined over \( F_q \) (where \( F_q \) denotes the finite field containing \( q \) elements and \( q \) is a prime number), a point \( P \in E(F_q) \) of order \( n \), and a point \( Q \in E(F_q) \). Find out the integer \( l \), \( 0 \leq l \leq n - l \), such that \( Q = lP \) (Certicom, 2000). Based on the difficulty of this problem, many cryptographic systems have been proposed including the elliptic curve analog of the DSA Elliptic Curve Digital Signature Algorithm (ECDSA), and the elliptic curve analog of the Diffie-Hellman key agreement scheme, the ElGamal encryption and signature schemes, the Schnorr signature scheme, and the Nyberg-Rueppel signature scheme.

#### 2.3.3.1 Elliptic Curve Cryptosystem

Although elliptic curve has been studied for several hundred years, it was not applied to the cryptography before Koblitz and Miller purposed it. In 1985, Neal Koblitz and Victor Miller separately introduced the ECC as another alternative public key cryptosystem. Its security depends on the difficulty of the ECDLP that admits highest security strength of all known public key cryptosystems. The security level is determined by the size and structure of the group of rational points lying on an elliptic curve defined over a finite field.

Elliptic curve cryptosystem is based on elliptic curve point multiplication and the apparent intractability of ECDLP: Given an elliptic curve \( E(F_q) \) defined over the finite field \( F_q \) and two points \( P, Q \in E(F_q) \), find an integer \( k \), \( 0 \leq k \leq r-1 \) such that \( Q = kP \), provided such an integer exists (Edoh, 2004).
Menezes (2001) gives some comparisons of the difficulty of the three hard mathematical problems which provide the basis for the security of public key cryptosystems used today: IFP, DLP, and ECDLP. He figured out that the ECDLP is computationally more difficult than the DLP or IFP. Besides, the expected time to solve the ECDLP with a key size 160-bit is approximately equal to the time required to solve the DLP and IFP 1024-bit key size.

(a) Elliptic Curve Mathematics Principle

1. Finite field

Fields are abstractions of familiar number systems (such as the rational numbers, the real numbers, and the complex numbers) and their fundamental characteristic. They consist of a set $F$ together with two operations, addition (denoted by $+$) and multiplication (denoted by $\cdot$), that satisfy the usual arithmetic properties:

- $(F, +)$ is an abelian group with (additive) identity denoted by 0.
- $(F \setminus \{0\}, \cdot)$ is an abelian group with (multiplicative) identity denoted by 1.
- The distributive law holds: $(a+b) \cdot c = a \cdot c + b \cdot c$ for all $a, b, c \in F$.

If the set $F$ is finite, then the field is said to be finite (Edoh, 2004).

A finite field contains of a finite set of elements $F$ together with addition and multiplication operations on $F$. The number of elements in the finite field is the order of that field. There is only one finite field of order $q$ if and only if $q$ is a prime power; this field is denoted by $F_q$. If $q = p^m$ where $p$ is a prime and $m$ is a positive integer, then $p$ is called the characteristic of $F_q$, denoted $\text{char}(F_q)$, and $m$ is called the extension degree of $F_q$ (Menezes, 2001).

Two kinds of fields those are especially amenable for the efficient implementation of elliptic curve systems are (Hankerson et al., 2004):
• Prime fields: Let $p$ be a prime number. The integers modulo $p$, consisting of the integers $\{0, 1, 2, \ldots, p-1\}$ with addition and multiplication performed modulo $p$, is a finite field of order $p$; denote by $F_p$ and call $p$ the modulus of $F_p$. For any integer $a$, $a \mod p$ shall denote the unique integer remainder $r$, $0 \leq r \leq p-1$, obtained upon dividing $a$ by $p$; this operation is called reduction modulo $p$.

• Binary fields: or characteristic-two finite fields are Finite fields of order $2^m$. Where a polynomial basis representation used to create it. The elements of $F_{2^m}$ are the binary polynomials (polynomials whose coefficients are in the field $F_{2} = \{0, 1\}$) of degree at most $m-1$:

$$F_{2^m} = \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots + a_2z^2 + a_1z + a_0 : a_i \in \{0,1\}\}.$$  

2. Elliptic Curve

An elliptic curve $E$ over a field $F$ is defined by a Weierstrass equation (Cilardo et al., 2006):

$$E: y^2 + a_1x y + a_3 y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in F$ and $\Delta \neq 0$, where $\Delta$ is the discriminant of $E$ and is defined as follows: $\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$ , and $d_2 = a_1^2 + 4a_2, d_4 = 2a_4 + a_1a_3, d_6 = a_3^2 + 4a_6, d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$. The sum of $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ (with $P \neq Q$) is given by $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 + a_1 \lambda - a_2 - x_1 - x_2, y_3 = - (\lambda + a_1)x_3 - \mu - a_3,$ with $\lambda = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)$ if $P \neq Q$, and $(3x_1^2 + 2a_2x_1 + a_4 - a_1y_1)/ (2y_1 + a_1x_1 + a_3)$ if $P = Q$, and $\mu = y_1 - \lambda x_1$.

Two main types of elliptic curves have been used in cryptography, which are based according to the base field over which the curve is defined:

• If $F_p$ is a field with characteristic $\neq 2, 3$. The general Weierstrass equation can simplified to: $y^2 = x^3 + ax + b$  

(1)
where \( a, b \in F_p \) and \( 4a^3 + 27b^2 \neq 0 \mod p \), where \( a_1 = a_2 = a_3 = 0, a_4 = a, \) and \( a_6 = b \) in the general Weierstrass equation, the sum of \( P = (x_1, y_1) \) and \( Q = (x_2, y_2) \) (with \( Q \neq -P \)) is given by \( P + Q = (x_3, y_3) \), where \( x_3 = \lambda^2 - x_1 - x_2 \) and \( y_3 = \lambda (x_1 - x_3) - y_1 \), and

\[
\lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ if } P \neq Q \text{ and, } \lambda = \frac{3x_1^2 + a}{2y_1} \text{ if } P = Q.
\]

- If \( F_p \) is a field of characteristic-two (Binary field) the equation for (non-supersingular) elliptic curves is given by:

\[
y^2 + xy = x^3 + ax^2 + b \tag{2}
\]

where \( b \neq 0 \), and the sum of \( P = (x_1, y_1) \) and \( Q = (x_2, y_2) \) (with \( Q \neq -P \)) is given by \( P + Q = (x_3, y_3) \), where: \( x_3 = \lambda^2 + \lambda + x_1 + x_2 \) and \( y_3 = \lambda (x_1 + x_3) + x_3 + y_1 \), with \( \lambda = \frac{y_1 - y_2}{x_1 - x_2} \) if \( P \neq Q \) and, \( \lambda = x_1 + (y_1/x_1) \) if \( P = Q \).

(b) ECC Encryption, Decryption, and Signature Algorithm

When two parties are going to use elliptic curve cryptography, there are certain parameters they should agree on, either they were selected by them or by a third party; those parameters are called the elliptic curve domain parameters, which identify the arithmetic operations implicated in the public key cryptographic scheme, the six domain parameters are the following (Mogollon, 2007):

\[ T = (q; FR; a, b; P; n; h) \]

in which,

1. \( q \): Identifies the implicit finite field \( F_q \). The field size is defined by the module, so, \( q = p \) or \( q = 2^m \); \( p > 3 \) should be a prime number.
2. \( FR \): Field Representation that (is) used for representing field elements in \( F_q \).
3. \( a, b \): The coefficients defining the elliptic curve \( E \), elements of \( F_q \).
4. \( P \): A point on an elliptic curve called the base point or generating point; it is defined by two field elements \( x_p \) and \( y_p \) in \( F_q \).
5. n: The order of the base point P, which is a large prime.

6. h: Called the cofactor, h = \#E(F_q) / n.

The public and private keys are related with a particular set of elliptic curve domain parameters (q; FR; a; b; P; n; h). To generate a key pair, entity A does the following (Mogollon, 2007):

- Chooses a random or pseudorandom integer \( d \) in the interval \([1, n-1]\).
- Calculates \( Q = d \times P \).
- Returns \((Q, d)\), where \(Q\) is the public key and \(d\) is the private key.

The simplest method to encrypt and decrypt a message \( m \) in the ECC is done as follows (Stallings, 2005):

- Each entity generates his domain parameters and key pairs, and each entity has a copy of the domain parameters and public key of the other entity.
- Entity A, who wants to send an encrypted message to entity B, has to convert the message \( M \), which is represented as an integer in binary into an octet string, and then into a field element point \( P_m \). Then, he selects a random positive integer \( k \) and produces the cipher text \( C_m \) consisting of the pair of points:

\[
C_m = (kP, P_m + kQ_B)
\]

where, \( Q_B \) is entity B public key

- To decrypt the cipher text, entity B multiplies the first point in the pair by his private key and then subtracts the result from the second point:

\[
kP * d_B \text{ where, } d_B \text{ is entity B private key.}
\]

\[
(P_m + kQ_B) - kPd_B \text{ since } Q_B = Pd_B \text{ then, } P_m + kPd_B - kPd_B = P_m.
\]

In 1992 Vanstone was the first who proposed the well-known signature scheme ECDSA, which is the elliptic curve analogue of DSA. In 1998, it was accepted as one of the International Standards Organization (ISO) standards. And it was accepted as a
standard in the American National Standards Institute (ANSI) in 1999, as well as accepted in 2000 as an Institute of Electrical and Electronics Engineers (IEEE) standard (Han and Yang, 2005).

ECDSA is the mainly commonly standardized elliptic curve-based signature system, showing in the ANSI, FIPS, IEEE and ISO standards as well as several draft standards, and is done as follows (Hankerson et al., 2004):

- **Signature generation**, which takes the domain parameters $D = (q, FR, a, b, P, n, h)$, private key $d$, and message $m$ as input, and produces the signature $(r, s)$:
  1. Choose $k$ in the interval $[1, n-1]$.
  2. Calculate $kP = (x_1, y_1)$ and convert $x_1$ into an integer $x_1^{-1}$.
  3. Determine $r = x_1^{-1} \mod n$.
  4. Calculate $e = H(m)$ where, $H$ denotes a cryptographic hash function.
  5. Compute $s = k^{-1} (e + d \cdot r) \mod n$.
  6. Return $(r, s)$.

- **Signature verification**, which takes the domain parameters $D = (q, FR, a, b, P, n, h)$, public key $Q$, message $m$, and signature as input. Then accept or reject the signature:
  1. Confirm that $r$ and $s$ are integers within the interval $[1, n-1]$. If the verification fails then return (“Reject the signature”).
  2. Calculate $e = H(m)$.
  3. Determine $w = s^{-1} \mod n$.
  4. Compute $u_1 = e \cdot w \mod n$ and $u_2 = r \cdot w \mod n$.
  5. Calculate $X = u_1P + u_2Q$.
  6. If $X = \infty$ then return (“Reject the signature”);
  7. Change the $x$-coordinate $x_1$ of $X$ to an integer $x_1^{-1}$; determine $v = x_1^{-1} \mod n$. 

21
8. If \( v = r \) then return (“Accept the signature”); Else return (“Reject the signature”).

(c) Attacks on ECC

Since 1985, the ECDLP has received significant concentration from mathematicians around the world. Pohlig and Hellman (1978) introduced an algorithm to reduce the determination of \( l \) to the determination of \( l \) modulo each of the prime factors of \( n \). Hence, in order to achieve the maximum possible security level, \( n \) should be prime. Van Oorschot and Wiener (1994) showed how the Pollard rho method can be parallelized so that if \( r \) processors are used, then the expected number of steps by each processor before a single discrete logarithm is obtained is \( \sqrt{\pi n / 2r} \).

Menezes et al. (1991) demonstrated how the ECDLP can be reduced to the DLP in extension fields of \( F_q \), where the index-calculus methods applied for a very special class of curves known as Supersingular curves. Nevertheless, supersingular curves are specifically prohibited in all standards of elliptic curve systems.

Many choices have to be made before implementing an ECC system. Among them is the elliptic curve underlying field (Cilardo et al., 2006). The efficient implementation of finite field arithmetic is an important requirement in elliptic curve systems because curve operations are performed using arithmetic operations in the underlying field (Hankerson et al., 2004).

When implementing an elliptic curve cryptosystem, three essential decisions must be into consideration (Certicom, 2000):

1. Choosing of the fundamental finite field \( F_q \).
2. Selection of the representation for the elements of \( F_q \).
3. Picking out of the elliptic curve \( E \) over \( F_q \).
2.3.3.2 ECC on Mobile Devices

The explosive growth in the use of mobile and wireless devices demands a new generation of PKC schemes that has to accommodate limitations on power and bandwidth, at the same time, to provide a suitable level of security for such devices. As ECC provides the same level of security with smaller key sizes, as compared to the customary cryptosystems, the perfect solution to provide more security without adding to the computational load is to use ECC instead of the traditional cryptosystems such as RSA (Dabholkar and Yow, 2004). Miao et al. (2005) developed a security solution for mobile communication using ECDLP, and one digital module of ECC was given. The solution had achieved fast arithmetic associated with ECC in finite field GF(p). The security of ECC with 160-bit key size is equal to RSA with 1000-bit key size and the arithmetic performance of addition and subtraction against the other field was 20%-40% faster than common.

Guajardo et al. (2001) used the 16-bit TI MSP430x33x family of microcontrollers to implement ECC over prime fields. Which demonstrate that it is still probable to implement ECCs in restricted embedded systems and get suitable performance. They decreased the number of intermediate variables by modifying the EC point addition and doubling formulae when on the other hand allowing for flexibility. These ideas are merged to attain an EC scalar point multiplication in 3.4 seconds not include stocking or precomputing any values while the processor clocked at 1 MHz.

Weimerskirch et al. (2001) used Motorola Dragonball CPU which offers 16-bit and 32-bit operations and runs at 16 MHz to implement EC over binary fields. They used Koblitz curves over $GF(2^{163})$ to achieve less than 0.9 sec execution time for ECDSA signature generation operation and less than 2.4 sec for signature verification operation. The authors point out that Koblitz curves over fields $GF(2^{163})$ provide about the same level of security
as RSA with a 1024-bit length, while at the same time providing acceptable performance which is not possible to achieve by using RSA-based systems since the integer multiplier in the Dragonball processor is very slow.

Based on the fact that elliptic curve systems is smaller in size and at the same time keeping the same level of security, the result is smaller key sizes, bandwidth savings, and faster implementations, which are particularly attractive for the low computational power devices such as smart cards, personal digital assistants, and wireless devices (Menezes, 2001).

2.3.4 Comparison of ECC and RSA

The primary advantage that elliptic curve systems have over systems based on the multiplicative group of a finite field and also over systems based on the intractability of integer factorization is the absence of a sub-exponential-time algorithm that could find discrete logarithms in these groups. Thus, to preserve the same level of security the key length required in the RSA generated key pair will be so much bigger than in the ECC generated key pair, as seen in Table 2.1 below (Kapoor and Abraham, 2008).
Table 2.1: Comparison of strength of RSA and ECC (Kapoor and Abraham, 2008).

<table>
<thead>
<tr>
<th>Time to break (in MIPS-years)</th>
<th>RSA key-size (in bits)</th>
<th>ECC key-size (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>512</td>
<td>106</td>
</tr>
<tr>
<td>$10^8$</td>
<td>768</td>
<td>132</td>
</tr>
<tr>
<td>$10^{11}$</td>
<td>1024</td>
<td>160</td>
</tr>
<tr>
<td>$10^{20}$</td>
<td>2048</td>
<td>210</td>
</tr>
<tr>
<td>$10^{78}$</td>
<td>21000</td>
<td>600</td>
</tr>
</tbody>
</table>

Menezes et al. (1997) mentioned that to reach sensible security, a 1024-bit modulus would have to be used in RSA systems, while at the same time 160-bit modulus should be enough for ECC. Certicom, a Canadian company, has been worded on the ECC system since the early ’80s. Table 2.2 as follows showed one of their results for fast implementations of ECC compared to RSA (Kapoor and Abraham, 2008):

Table 2.2: Comparison of RSA and ECC (Kapoor and Abraham, 2008).

<table>
<thead>
<tr>
<th>Function</th>
<th>ECC 163-bit (in ms)</th>
<th>RSA 1024-bit (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key generation</td>
<td>3.8</td>
<td>4708.3</td>
</tr>
<tr>
<td>Signature Generation</td>
<td>3.0(ECDSA)</td>
<td>228.4</td>
</tr>
<tr>
<td>Signature Verification</td>
<td>10.7(ECDSA)</td>
<td>12.7</td>
</tr>
</tbody>
</table>
Elliptic curve cryptosystems present a security level more than any well-known public-key system. An elliptic curve system with a 160-bit small key size provides security strength as DSA or RSA with 1024-bit key size. While on the other hand ECC smaller key sizes result in smaller system parameters, smaller public-key certificates, bandwidth savings, faster implementations, and lower power requirements (Jurisic and Menezes, 2005).

2.4 Public Key Infrastructure (PKI)

Key management is defined as the set of actions and methods that maintain the creating and the observation of keys material during their life cycle (Arslan and Alagöz, 2006). The objective of key management is to present secure processes for managing cryptographic keying material to be used in symmetric or asymmetric cryptographic systems (Fumy and Landrock, 1993). Menezes et al. (1997) defined key management as the set of procedures and activities saving the setting up and protection of keying materials between the authorized parties.

Key management includes procedures and processes standing up for:

1. Establishing the connection between the domain and the users systems.
2. Generation, delivery, and installation of user's keys.
3. Administrate the use of keying material.
4. Updating, invalidation, and destruction the user's keys.
5. Storage, backup/recovery, and archival of keying material.

The security in the mobile communication systems is a serious issue, and in view of the fact that the mobile applications have special constraints and weaknesses, therefore those applications need a special concern. The most important problem is designing protocols for authentication and key management where any mistakes at this phase will weaken the
security of the whole communication process, and big possibility the following sessions as well (Boyd and Mathuria, 2000).

PKI is known as a set of procedures to provide the security services to protect the users communications in administrative domain through certificate authorities and digital certificate (Bao, 2000; Hunt, 2001; Hazari, 2002). In the PKI, digital certificates are an important component which operates as digital passports to combine the user’s digital signature and other information to their public key certificate (Hunt, 2001).

The function of PKI is to guarantee the authenticity of the combining between a public key and a specific identity using the public key certificate; which is a digitally signed data structures contains the signature of the PKI administrator to insure the authenticity of the PKI user public key. The effectiveness and trustworthiness of the PKI is based on a methods and procedures for saving and recovering the digital certificates (Wörlfl, 2005).

2.4.1 Components of the PKI

According to Hunt (2001) a PKI consists of:


2. Certificate Authorities (CA): CA is the core component of a PKI, which issue the digital certificate for users, responsible for the status of the certificates it issues.

3. Registration Authority (RA): RA offers the interface between the user and the CA and submits the certificate request to the CA after authenticates the identity of the users.

Dankers et al. (2002) represented the basic PKI needed to manage the certificates during their lifecycle. (The main steps of certificate life cycle management in Figure 2.1 are as follows):

Figure 2.1: Basic concepts of PKI (Figure adapted from Dankers et al., 2002).

1. Registration and key pair generation: End entities must be registered before it can use the service supported by PKI. Throughout the registration and according to the policy of the KPI the identity of the end-entity is established and verified. The registration process depends on who will generates the key pair as is shown below:
   - If the end-entity generates the key pair, after that the end-entity passed the public key to the CA.
   - If the CA generates the key pair, at that moment CA securely passed the private key to the end-entity.
2. Certificate generation and distribution: the digital certificate is issued and distributed to the end-entity as soon as the end entity has been verified and the key pair generated, at the same time the certificate stored at the certificate repository.

3. Certificate expiration: When the certificates are expired, the end-entity informs the CA to renew it.

4. Certificate revocation: when the private key has been disclosed, the CA has to revoke the corresponding certificate.

5. Certificate retrieval: where the end entities can reacquire certificates from the certificate repository.

6. Certificate validation: in order for end entities to validate certificates, the end entity need to get the CRL from the CRL repository or may make use of On-line Certificate Status check Protocols (OCSP).

2.5 Global System for Mobile Communications (GSM)

The GSM is a European Telecommunications Standards Institute (ETSI) standard. In 1990, the specification of stage one of the GSM was published, and now is the most usually used cellular phone system in the world. Beside the basic service that GSM supported i.e. telephony, GSM allows data to be transported as a service. GSM standard is taken as a Second Generation (2G) cellular system, and was designed to have strong subscriber authentication and Over The Air (OTA) transmission encryption (Croft and Olivier, 2005).

2.5.1 GSM Architecture

Croft and Olivier (2005) represented the GSM structure, and the main components of its architecture as shown in Figure 2.2 are as follows:
- The Mobile Station (MS): is the mobile device.
- The Base Transceiver Station (BTS): is a radio tower which the Mobile Station communicates with.
- The Base Station Controller (BSC): connects multiple BTSs with the network’s backbone.
- The Mobile Switching Centre (MSC): presents the switching utilities of the network. This has an interface to one or more BSCs and to external networks. It has databases to control and manage the network, the following are usually considered to be part of the MSC:
  - Home Location Register (HLR): contains the profiles, and location information data for all registered users within the network operator.
  - Visitor Location Register (VLR): is responsible for specific location areas, which stores data about the users who locate in those areas.
  - Authentication Centre (AuC): is responsible for authenticate the MS to the network.
  - Equipment Identity Register (EIR): registers equipment data.
2.5.2 Short Message Services in GSM

Short Message Services (SMS) is one of the services in the GSM, and Universal Mobile Telecommunications System (UMTS). SMS is a universal text messaging system which offers a mechanism to transmit short messages “to” and “from” mobile devices with a maximum of 160 characters message’s length. SMS requires low bandwidth and it is a
low cost service compared with others which makes it suitable for a set of applications (Garza-Saldana and Diaz-Perez, 2008).

In order for a mobile user to send and receive messages or communicate with the services provider, subscription to the service is needed. All SMS messages are sent from the MS to SMSC, which stores them until they can be delivered to the destination. Transmission of the short messages between SMSC and MS is via Signalling System Number 7 (SS7) within the GSM Mobile Application Part (MAP) framework, which identifies methods and mechanisms for communication in wireless networks between the components (Zhao et al., 2008).

2.5.3 Security on Short Message Services

GSM MAP is unencrypted protocol used to transmit the SMS between SMSC and phone via the SS7, the result of which is that anyone who has access to the SS7 can have the ability to read and/or modify SMS messages (Zhao et al., 2008).

Current messaging systems provide only point-to-point of some security services, encrypt the traffic between the MS and the BTS using the stream cipher (A5/1 or A5/2), which is a weak and broken algorithm, and is the only encryption involved during transmission (Toorani and Shirazi, 2008).

GSM network uses A3/A8 authentication algorithm to authenticate the Subscriber Interface Module (SIM) card to the AuC based on a shared secret key between SIM card and the AuC, which is stored in the SIM card during the manufacture, and the AuC. Therefore, there is no end-to-end security from service provider to mobile users and from MS to MS, which is a security weakness for messaging service in mobile networks (Zhao et al., 2008).
Hassinen and Markovski (2003) proposed application for sending secure SMS messages using cryptographic methods based on theory of quasigroups, which is characterized with a structure called Latin square. A Latin square is an $n \times n$ matrix where each row and column is a permutation of elements of a set. The decryption and the encryption are table look-up methods. The Latin square is like a shared key between two entities which is considered as a symmetric key cryptography.

Croft and Olivier (2005) provided a solution to secure the SMS message using one-time pads based on shared information between the communicating entities and the GSM network, where the keys generated based on this shared information uses hashing techniques. During the transmission, the MSC acts as a mediator between the two entities, where it receives the SMS from the sender and decrypts it. It then locates the recipient, and encrypts the SMS message for the recipient and delivers it. This mechanism does not guarantee end-to-end security between the two mobile phones because there is a dependency on the network infrastructure where a decryption takes place within the mobile network.

Hassinen (2005) developed SafeSMS which is an application meant for end-to-end encryption, which can send, receive, and store encrypted text messages. Encryption is based on a shared secret password (Symmetric or Secret key encryption) between the sender and the recipient. Although it is simple and fast, there is a high possibility that the key will be exposed to the public, and it may have a problem on key distribution and thus decreases the reliability on service.

Chikomo et al. (2006) designed a secure SMS protocol to protect SMS communication in the remote mobile banking system using symmetric cryptography. The key used for encryption, and decryption is a one-time password just known by the server and the user. The server uses a database to store the one-time password, and indexes it by the account
identifier and the sequence number. The authors state that the protocol offers confidentiality, integrity, authentication, and non-repudiation to mobile banking service using SMS. On the other hand, this protocol does not provide end-to-end security.

Ratshinanga et al. (2004) proposed a secure SMS protocol using public and symmetric key cryptography and password authentication. The protocol is divided into two parts, the handshake which is the key exchange and authentication part of the protocol, and transaction which is the transactions of an encrypted SMS. The proposed protocol use(s) 1024-bit RSA for the public key algorithm and 128-bit AES/CTR for symmetric key and block cipher mode algorithm and Secure Hash Algorithm Version 1.0 (SHA-1) for the hash algorithm.

As mobile communication become so important in the daily use more than before, the initiative of using PKI in mobile was adapted by Tadashi Kaje. He applied the PKI encryption to mobile SMS, and provided a new way to communicate securely using mobile SMS (Anuar et al., 2008).

Hassinen (2006) developed Java based PKI for SMS, which is an approach to solve the security issues of SMS messaging by developing an application using the Java 2 Micro Edition (J2ME) environment. In J2ME the Wireless Messaging API (Application Programming Interface) (WMA) provides tools for sending and receiving SMS messages. The proposed solution works on all Java enabled mobile phones that support the WMA and MIDP 2.0 (Mobile Information Device Profile). The application can send and receive SMS messages in binary format using 1024-bit RSA cryptosystem.

IPCS Group (2006) developed PKI for SMS call IPCryptSIM™. IPCryptSIM™ is a java based SIM card solution for mobile SMS, and works with every type of mobile phones to exchange private and secure SMS. It uses a 128-bit RSA algorithm for encryption.
Anuar et al. (2008) developed a solution for GSM Mobile SMS and MMS using PKI called m-PKI, which provide(s) PKI encryption to the mobile SMS and MMS. All m-PKI users have to possess a Java enabled phone. This application allows the user to send private and classified message via SMS. The application uses the RSA cryptosystem with 512-bit key size, which is chosen by the tradeoff between performance and security, while the key pair generation and distribution are performed by the CA.

2.6 Conclusion

At present, the mobile handheld device successfully replaces the traditional telephone to become the most popular wireless communication tools, and fulfills almost all the user’s requirements as an effective communication and information delivering service. Currently the messaging services are not providing end-to-end secure SMS, where the users can not communicate securely and exchange private messages. Many Applications have been developed to solve the SMS security issues using symmetric or asymmetric cryptography. Despite symmetric cryptography considered fast and simple implementation it has an austere problem which is the key distribution process. The low speed in practice obviates the public key cryptography (asymmetric cryptography) from being implemented onto mobile phones. Where the developed asymmetric solutions used small key size to get suitable performance, but at the same time using small key size illuminate the security strength of those solutions.

ECC came as a new alternative public key cryptosystem to provide security strength more than any known public key system using smaller key sizes. The smaller key sizes result in smaller system parameters, smaller public key certificates, faster implementations, lower power requirements. Therefore, ECC is the best choice to solve SMS security issues in the
mobile devices since it provides acceptable performance in low power mobile devices with a high security level.

2.7 Summary

People are relying on SMS as a daily communication tool to send unconventional information. There is a demand to communicate or exchange confidential information in a secure environment. Elliptic curve cryptography is a proven solution.

The elliptic curve cryptography security depends on the difficulty of the ECDLP that admits the delivery of the highest strength per bit of all previously known public key cryptosystems, and that the ECDLP is computationally more difficult than the DLP or IFP. The ECC fulfills the requirement for mobile communication which ensures the highest security and performance on the mobile phone.
CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter describes the research methods and software process models used for Elliptic Curve Short Message Service (ECSMS) application development and implementation. Research methodology is a set of procedures or methods used to conduct research. The procedures adopted in the research methodology as shown in Figure 3.1 are:

- Collection of articles from different publication journal such as IEEE, ScienceDirect, and ACM to explore and understand the public key and private key cryptography, and the advantages of public key cryptography over the private key cryptography;
- Studying and reviewing relative tools to investigating ECC and PKI features, strength, and standards;
- Designing the framework for mobile ECSMS;
- Enhancing SMS to achieve high security level and acceptable performance using ECC techniques,
- Framework analysis, design, system implementation, and testing;
- Concluding the research.
A software development methodology is a significant methods and actions used to investigate the complication and difficulties during the software development process to decrease them and administrate the requirements of the development process (Trussell, 1998). The software development methodology used during the ECSMS development process is Rational Unified Process (RUP).
3.2 Rational Unified Process (RUP)

The Rational Unified Process is a software engineering process. It provides a techniques and procedures to specifying jobs and responsibilities within a development team to guarantee that the software outcomes are qualify enough for end-users satisfaction, as well as within a specific timetable and financial plan. Rational unified process is a new software process model that is derived from Unified Modeling Language (UML). The UML is an industry standard language use to simplify the softwares necessities, architectures and designs (Rational, 1998).

The Rational Unified Process describes how to successfully deploy business confirmed techniques to software development for software development teams. These are called best practices as follows (Kruchten, 2003):

1. Iterative software development.
2. Administrate the changing requirements.
3. The architectures used are the component-based
4. Using the UML software modeling.
5. Always validate the quality of the software.
6. Keeping track the software changes and manage it.

RUP consists of four phases and nine disciplines as illustrated in Figure 3.2.
3.2.1 Inception Phase

The main goals of the Inception phase are to understand the initial requirements, define and come to agreement with respect to the high-level requirements, vision, and scope of the project, as well as to justify the project and obtain resources to continue work on it. The Inception phase is where you define the project scope and the business case for the system, which means to define who will interact with the system actors and the character of this interaction. This requires identifying the initial use cases for the software, and describes the key ones briefly (Ambler and Constantine, 2000).

The main outcomes in this phase are; the Project Plan which showed function, scope and objectives, assumptions, project deliverables, schedule, budget summary, roles and responsibilities. One of the most important deliverables of this plan is scope statement as shown in table 3.1 below.
Table 3.1: Scope Statement

<table>
<thead>
<tr>
<th>Project Title: ECSMS</th>
<th>Prepared by: Addy Al-Qura`an</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: 10-03-2009</td>
<td></td>
</tr>
</tbody>
</table>

**Project Justification:**
This project aims to implement end-to-end secure SMS using the ECC. The main important reason for developing this project is to insure the mobile users can communicate and exchange private message in the SMS using the Elliptic Curve Cryptography. This research provides solutions for mobile SMS security issues and the mobile phone power limitations through the Elliptic Curve Cryptography.

**Product Characteristics and Requirements:**
1.1 Encrypting and decrypting the SMS using the ECIES.
2.1 Using the ECDSA for signature generation and verification.
3.1 Application can be run on Java-enabled mobile phone.

**Summary of Project Deliverables**

**Project management-related deliverables:** business case, charter, team contract, scope statement, WBS, schedule, cost baseline, status reports, final project presentation, final project report, lessons-learned report and any other documents required to manage the project.

**Product-related deliverables:** research reports, design documents, software code, hardware.

**Project Success Criteria:**
1. Improvement over time
2. Fulfilling mentioned requirement
3. Good communication among stakeholder.

The second outcome is the initial use-case model (10% -20%) complete. The ECSMS analysis and design chapter shows the use-case illustrations to the system with regard to inception phase of RUP.
3.2.2 The Elaboration Phase

After the system specifications have been well captured, the Elaboration Phase comes to focuses on detailed analysis of the problem domain and the definition of an architectural foundation for the project, and to define the supplementary specification which describes all nonfunctional requirements for the system. The main outcomes of this phase include, firstly, a Use-Case model (at least 80% complete), all use cases and actors have been identified, and most use-case descriptions have been developed to define the its scenarios, preconditions, and post-conditions as in section 2 chapter 4. Secondly, activity diagrams to represent the behaviors and the executions flow of the system functions and processes as in section 3 chapter 4.

3.2.3 The Construction Phase

The purpose of the construction phase is to develop the system to the stage that it can be deploy. That is, where all maintaining parts and application features are developed and included into the product and the whole features are tested. This is the phase where the bulk of the coding takes places as the first release where the software models include the class and sequence diagrams are sufficient to used as a guide to design and implement of the system flow and module integration as in section 4 and 5 chapter 4. The main outcome in this phase is the description of the current release, which is the result that comes out after the first implementation; where the application can encrypt and sign the SMS then send it, and receive the same SMS then decrypt it and verify the signature as in section 2 chapter 5.

3.2.4 The Transition Phase

The transition phase focuses on testing and validating the complete system to come out with the final production of the software; this includes operating the system and supporting
the users to work with it, and responding to error and features request, which requires to develop a new release. Normally, this phase includes several iterations to achieve the complete final product. The main outcome in this phase is to roll-out of the product out to the market.

3.3 Why Rational Unified Process

Rational Unified Process is an effective modeling using UML. It organizes the projects in terms of phases where each phase consists of more than one iteration. The main reasons for choosing RUP as a software process model are (Ambler, 2005):

- Improved governance by delivering high quality working software which meets the actual needs of users.
- Regular feedback to stakeholders, where they see and use the system to assure that they will get what they want.
- Implement the actual requirements: by using iterative process to integrate any changes in the requirements. Therefore, risks can be discovered and eliminated during integration.
- Modeling visualization: where UML gives standard means of system description

3.4 Software Development Environments

Most of the recent mobile devices are designed to fit applications written by third parity developers where users can run applications not provided by the manufacturer. According to Hassinen (2006), most of the mobile applications use the J2ME (Java 2 Micro Edition) platform, which was developed for small devices like mobile phones and it has better portability across device platforms. As this research solves the security issue of the mobile
SMS messaging and the J2ME has Wireless WMA, which provides tools for sending and receiving SMS messaging. J2ME becomes the best choice to build the ECSMS application.

3.5 Requirements

The following are the list of the requirements that have been used as the requirements statement of building the system:

- The application is able to authenticate each user by using the user names and password verification.
- If it is run for the first time, the application asks the user to create his account. After that, the application downloads the user's certificates. After authenticating the user's certificate, the application creates an entry for each user in the contact list with his name, number, public key, and certificate expire date.
- The application is able to send an encrypted and signed SMS using the elliptic curve cryptography for any registered user, and save the massage in the outbox.
- The application is able to receive encrypted and signed SMS. Then decrypt it and verify the signature, after that save it in the inbox.
- The application saves all the processing time in the statistics menu, which shows the average time of key generation, encryption, decryption, signing, and verifying.
- The application allows the user to update his configurations by changing his username and/or password.

3.5.1 Hardware Requirements

The application development will require:

- One PC with 1 GB RAM and 40GB HDD, which used as development PC.
J2ME enabled mobile phone for real implementation to test different features of the application and ensure their proper functionality. Nokia N80 will be used in testing and implementation of the application.

3.5.2 Software Requirements

The application development requires the following software packages:

a) NetBeans IDE

The NetBeans™ Integrated Development Environment (IDE) is an open source for software developers to create desktop, enterprise, web, and mobile applications with the Java language. The NetBeans IDE Mobility Pack is used to develop, test, and deploy the J2ME platform enable mobile devices applications for the Connected Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP). It is used to develop GUIs quickly with the Visual Mobile Designer (VMD); "Drag and drop components like wait screens, login screens, file browsers, an SMS composer, and splash screens". The IDE integrates tools for automatic code obfuscation with ProGuard 4.2, push registry emulation, WMA emulation for SMS, new deployment methods, and Wireless Messaging and Multimedia APIs; this makes it the best choice to build the ECSMS application (NetBeans, 2010).

b) The Legion of the Bouncy Castle Java Cryptographic APIs

The Legion of Bouncy Castle is an open-source Java implementation of cryptographic algorithms to provide the insufficiency of encryption libraries in the MIDP specification. Bouncy Castle for J2ME has a lightweight API that includes a simple cryptographic framework and a large number of algorithms (Nokia, 2003).

c) Nokia PC Suite
Nokia PC Suite is a software used to set up an interface between Nokia mobile devices and computers as it is used to send applications, files, and SMS, or being as a modem to connect the computer to the internet. It was used to install the application into the Nokia phone through cable (Nokia, 2009).

3.6 Conclusion

This chapter explains the methodology that is used in this research. Firstly, the research handles the elliptic curve cryptography and how it is used in the mobile phone messaging security as the most effective solution in terms of security and performance. Thus, the contribution of this research is to implement the elliptic curve cryptosystem in the mobile phone to provide end-to-end secure mobile SMS. Secondly, it addresses the Rational Unified Process (RUP) as a software process modeling used in this research, and shows how to apply its four iterative phases and their deliverables. Thirdly, it indicates the software development environments and requirements.
CHAPTER 4
ECSMS ANALYSIS AND DESIGN

4.1 Introduction

This chapter focuses on the ECSMS analysis and design for all features by presenting different diagrams that demonstrate the analysis and design of the application. Object Oriented Analysis will be used to analyze and design the system. This includes the listing of functional and non-functional requirements, the system deportment analysis (evaluating the inside interaction between the system functionality by the means of Use Case diagrams and use cases Templates), the activity diagrams (showing the whole process in details for complex use cases), and the class diagram and sequence diagram (capture the structure of the system).

4.2 Use Case Diagrams and Templates

4.2.1 Use Case Diagrams

The use case diagram is a set of scenarios that describe the system functionality as a set of tasks that should be done by the system and the interaction between a user and a system in order to complete those tasks. The two main components of a use case diagram are use cases and actors. An actor represents a user or another system that will interact with the system that it models. A use case is an external view of the system that represents some action the user might perform in order to complete a task. Figure 4.1 shows the use case diagram which captures all the functionalities of the system that the mobile phone user can perform.

First, the system will detect if the user runs the application for first time or not, if yes, the user will create his account by choosing a username and password. After that, the system automatically will open the certificate sent by the certificate authority and save...
all the contacts with their names, numbers, keys, and validation time in the contacts database. If it is not run for the first time, the system will validate the user password and username. After that, the user can choose to do so many operations that are identified in the diagram.

If the user selects the manipulate messaging, which is the heart of the system as the diagram showed, this use case is extended into three other use cases like sending message, viewing outbox, and manipulating inbox where he/she can send SMS and read new SMS. Besides, he/she can read, view details, and delete his/her sent and received messages.

![Use Cases Diagram](image-url)

**Figure 4.1: Use Cases Diagram.**
The other operations the user can perform are view statistics which shows the average time for key generation, encryption, decryption, signature generation, and signature verification. In addition, the change configuration gives the user the opportunity to change his/her username and/or password.

### 4.2.2 Use Cases Templates

Use case template is a documentary detailed for the use cases and help to draw the activity diagrams as is shown in the following section. It has seven fields: the name, the actors, preconditions which must be accomplished before starting a use case, post-conditions which must take place after the execution of the use case, main scenario, alternative scenario, trigger to activate the use case, and priority of the use case. Table 4.1 showed use case template for creating an account operation where the actor is the phone user, the name is Create account, and the precondition is the system run for the first time, where the priority is Essential.
Table 4.1: Use case Template for Create Account operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Create Account.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Phone user.</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
<td>Essential.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>The ECSMS run for the first time.</td>
</tr>
</tbody>
</table>
| **Main Scenario** | - From the setup page, the user chooses a username.  
|               | - The user chooses a password.  
|               | - The user confirms his password.  
|               | - The user clicks button “Create”.  
|               | - The application will display a message that the create account operation is successfully done. |
| **Alternative Scenario** | If the text field is empty in step 1 or 2, the system will display a message that the text field should not be empty.  
|               | If the text field in step 3 doesn’t match with the text field in step 2, the application will display a message that the two fields must be the same. |
| **Post condition** | The system will take the certificates sent by the CA and verify it then save all users details in the contact database. After that, it displays the main menu list. |
| **Other**     | None.           |

Table 4.2 shows the use case template for login operation, where the actor is the phone user, the priority is Essential, the name is Login, and the precondition is system loaded.
Table 4.2: Use case Template for Login process.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Login.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>Essential.</td>
</tr>
<tr>
<td>Trigger</td>
<td>None.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded.</td>
</tr>
<tr>
<td>Main Scenario</td>
<td>1. From the login page, the user enters username.</td>
</tr>
<tr>
<td></td>
<td>2. The user enters password.</td>
</tr>
<tr>
<td></td>
<td>3. The user clicks button “Login”.</td>
</tr>
<tr>
<td>Alternative Scenario</td>
<td>If the text field is empty in step 1 or 2, the system will display a message that the text field should not be empty.</td>
</tr>
<tr>
<td>Post condition</td>
<td>The system will display the main menu list if the user entered a correct username and password, otherwise it will display an error message</td>
</tr>
<tr>
<td>Other</td>
<td>None.</td>
</tr>
</tbody>
</table>

Table 4.3 shows the use case template for sending message where the actor is the phone user, the priority is high, the preconditions is the system are loaded, and login operation is correctly done, and the postconditions are the sent message saved in the outbox and the system updates the statistics database.
Table 4.3: Use case Template for Send Message process.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Send Message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>High.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Clicking on Send button.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded and login process correctly performed.</td>
</tr>
</tbody>
</table>

**Main Scenario**

1. From the main menu page the user selects manipulate messaging.
2. The user selects compose message from manipulate messaging page.
3. The user selects the receiver.
4. The user writes the message he/she want to send.
5. The system will check the receiver certificate validation.
6. The system will generate the signature.
7. The system will encrypt the message.
8. The user allows the application to send the message.

**Alternative Scenario**

The receiver certificate is not valid or can't generate the signature or can't encrypt the message then the system will display an error message.

The user clears the message body and exit.

**Post condition**

The system will save the sent message in the outbox database.
The system will update the statistics database.

**Other**

None.
Table 4.4 shows the use case template for reading new messages, where the actor is the phone user, the priority is high, the preconditions are the system is loaded, login operation is correctly done, and has new message notification, where the postconditions are the received message saved in the inbox and the system updates the statistics database.

Table 4.4: Use case Template for Read New Message process.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Read New Message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>High.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Clicking on Open button.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded, login process correctly performed, and have a new message notification.</td>
</tr>
</tbody>
</table>
| Main Scenario | 1. From the main menu page the user selects Manipulate Messaging.  
2. From the Manipulate Messaging page the user selects Manipulate Inbox.  
3. The user selects the message that has "new" notification.  
4. The user selects Open from the option menu.  
5. The system will check the sender certificate validation.  
6. The system will decrypt the message.  
7. The system will verify the signature |
| Alternative Scenario | The sender certificate is not valid or can't decrypt the message or can't verify the signature then the system will display an error message. |
| Post condition | The system will remove the new message notification.  
The system will update the statistics database. |
Table 4.5 shows the use case template for viewing the inbox, where the actor is the phone user, and the preconditions are the system is loaded, and login operation is correctly done.

Table 4.5: Use case Template for View Inbox operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Inbox.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Phone user.</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
<td>High.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>Clicking on Inbox menu.</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>The ECSMS application is loaded and login process is correctly performed.</td>
</tr>
</tbody>
</table>
| **Main Scenario**     | 1. From the main menu page the user selects Manipulate Messaging.  
                         2. From the Manipulate Message page the user selects Manipulate Inbox.  
                         3. The user selects the View Inbox form the Manipulate Inbox.  
                         4. The user selects the message he/she wants to view, and then selects Open from the Option menu. |
| **Alternative Scenario** | In step number 4, the user can view the message details as long as the user can delete the message. |
| **Post condition**    | None.                              |

Table 4.6 shows the use case template for viewing outbox, where the actor is the phone user, and the preconditions are the system is loaded, and login operation is correctly done.
Table 4.6: Use case Template for View Outbox operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Outbox.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>High.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Clicking on Outbox menu.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded and login process is correctly performed.</td>
</tr>
</tbody>
</table>

**Main Scenario**

1. From the main menu page the user selects Manipulate Messaging.
2. The user selects the View Outbox form the Manipulate Messaging.
3. The user selects the message he/she wants to view, and then selects Open from the Option menu.

**Alternative Scenario**

In step number 3, the user can view the message details as long as the user can delete the message.

**Post condition**

None.

Table 4.7 shows the use case template for deleting message, where the actor is the phone user, the preconditions are the system is loaded, login operation is correctly done, where the user can delete a message either from inbox or outbox.
Table 4.7: Use case Template for Delete Message operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Delete Message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>High.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Clicking on Delete button.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded and login process is correctly performed.</td>
</tr>
</tbody>
</table>
| Main Scenario | 1. From the main menu page the user selects Manipulate Messaging.  
  2. From the Manipulate Messaging page the user selects Manipulate Inbox.  
  3. The user selects View Inbox from the Manipulate Inbox page.  
  4. The user selects the message he/she wants to delete. |
| Alternative Scenario | In step 2, the user can select View Outbox, and then he/she can select the message to be deleted.  
In step number 4, the user can view the message details as long as the user can Open the message. |
| Post condition | The system will delete the message from the database. |
| Other         | None.           |

Table 4.8 shows the use case template for viewing the message details, where the actor is the phone user, the preconditions are the system is loaded, login operation is correctly done, where) the user can views message details either form inbox or outbox.
Table 4.8: Use case Template for View Message Details operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Message Details.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor</strong></td>
<td>Phone user.</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
<td>High.</td>
</tr>
<tr>
<td><strong>Trigger</strong></td>
<td>Clicking on Details button.</td>
</tr>
<tr>
<td><strong>Precondition</strong></td>
<td>The ECSMS application is loaded and login process is correctly performed.</td>
</tr>
</tbody>
</table>
| **Main Scenario**   | 1. From the main menu page the user selects Manipulate Messaging.  
                      2. From the Manipulate Message page the user selects Manipulate Inbox.  
                      3. The user selects View Inbox from the Manipulate Inbox page.  
                      4. The user selects the message whose details he/she wants to view. |
| **Alternative Scenario** | In step 2, the user can select View Outbox, then he/she can select the message whose details to be viewed.  
                      In step number 4, the user can Open the message as long as the user can delete the message. |
| **Post condition**  |                       |
| **Other**           | None.                 |

Table 4.9 shows the use case template for viewing the statistics, where the actor is the phone user, the preconditions are the system is loaded and login operation is correctly done.
Table 4.9: Use case Template for View Statistics operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>View Statistics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>High.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Clicking on Statistics menu.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded and login process is correctly performed.</td>
</tr>
<tr>
<td>Main Scenario</td>
<td>From the main menu page the user selects Statistics.</td>
</tr>
<tr>
<td>Alternative Scenario</td>
<td></td>
</tr>
<tr>
<td>Post condition</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>None.</td>
</tr>
</tbody>
</table>

Table 4.10 shows Viewing Statistics use case template, where the actor is the phone user, the preconditions are the system is loaded and login operation is correctly done, and the Postcondition is the system will update the user database with the new username and password.

Moreover, table 4.11 shows the use case template for generating pair keys, where the actor is the phone user, the preconditions are the system successfully loaded, and login operation is correctly done, and the Postcondition is the system will update the statistics database.
Table 4.10: Use case Template for Change Configurations operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Change Configurations.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>High.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Clicking on Change button.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded and login process correctly performed.</td>
</tr>
</tbody>
</table>
| Main Scenario       | 1. From the main menu page the user selects Settings.  
                      | 2. From the Settings page the user selects Change Login.  
                      | 3. The user enters the current username.  
                      | 4. The user enters the current password.  
                      | 5. The user enters the new username  
                      | 6. The user enters the new password.  
                      | 7. The user will confirm the new password. |
| Alternative Scenario| If the text field step 3 or 4 doesn’t match with the current user configurations, the system will display an error message.  
                      | If the text field is empty in step 5 or 6, the system will display a message that the text field should not be empty.  
                      | If the text field in step 6 doesn’t match with the text field in step 7, the application will display a message that the two fields must be same. |
| Post condition      | The system will update the user password and username. |
| Other               | None.                                       |
Table 4.11: Use case Template for Generate Keys operation.

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Generate Keys.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor</td>
<td>Phone user.</td>
</tr>
<tr>
<td>Priority</td>
<td>High.</td>
</tr>
<tr>
<td>Trigger</td>
<td>Clicking on Generate button.</td>
</tr>
<tr>
<td>Precondition</td>
<td>The ECSMS application is loaded and login process is correctly performed.</td>
</tr>
</tbody>
</table>
| Main Scenario         | 1. From the main menu page the user selects Keys.  
                          2. From the Keys page the user selects Generate. |
| Alternative Scenario  |                |
| Post condition        | The system will update the Statistics. |
| Other                 | None.          |

4.3 Activity Diagrams

Activity diagram is one of the UML behavior diagram tools used to represent the overall behavior of the system. It models the actions flow executions and conditions that trigger those actions, and can model a specific use case at a more detailed level. Activity diagram graphically is composed of nodes and edges. The edges connect the nodes in sequential order. Nodes represent either Actions, Activities, or control nodes.

Figure 4.2 showed the activity diagram for the overall system.
Figure 4.3 explains the send-message activity in more details, where after the user selects the receiver he/she wants to send the message to, he/she writes his/her message, and allows the application to send the message; automatically the system first, will check the receiver certificate validation time. Second, it will generate the signature, and then encrypt the message. Third, the application will save the message in the outbox database, and update the statistics database. Finally it will inform the user that his/her message has been sent successfully.
Figure 4.3: Activity Diagram for Send Message Use Case.
Figure 4.4 shows the activities in more details for Manipulate Inbox use case, where the user can read, view details, and delete his/her messages. If the user wants to read the message that has a new notification, first the system will check the sender certificate validation time. Second, it will decrypt the message and then will verify the signature. Finally, it will open the message for the user to read, and update the statistics database.
4.4 Class Diagram

The class diagram is a UML diagram showing the object-oriented relationships that attached through the classes in the system. It shows the static view of the structural connections between the core classes that are used to build a system to allow communication and interaction among the classes. It also shows the relationships that hold among the classes in a system including inheritance, aggregation, and association, and shows the attributes and methods of each class. Figure 4.5 illustrates the class diagram for the system, which shows all classes and the relationships among all of them.

![Class Diagram](image.png)

Figure 4.5: Class Diagram.
It is clearly shown that the Elliptic Curve class is the core class, which connects to all other classes to perform all operations that the system has to do such as sending message where it gets the receiver contact number and public key from the contact class then passes it to the EC Helper class to generate the signature and encrypt the message. After that, the core class will open a connection and send the message. Finally it will update the statistics database and pass it to the Message class to save in the outbox database.

4.5 Sequence Diagrams

The sequence diagram is a UML tool for dynamic modeling, which identifies the behavior of the system, and explores the logic of a complex operations, functions, or procedures. Figure 4.6 shows the sequence diagram for sending a message, where the user selects the receiver then the elliptic curve class gets the receiver details from the receiver class and passes the message to EC helper to generate the signature and encrypt it.
Figure 4.6: Sequence Diagram for Sending Message.

Figure 4.7 illustrates the sequence diagram for reading a new message, where the elliptic curve class call the contact class to check the sender certificate validation time and get his/her details and pass it to the EC helper class to decrypt it and verify the signature.
Figure 4.7: Sequence Diagram for Reading New Message.

Where Figure 4.8 and 4.9 shows the sequence diagrams for deleting message from inbox, outbox respectively.
Figure 4.8: Sequence Diagram for Deleting Message from Inbox.

Figure 4.9: Sequence Diagram for Deleting Message from Outbox.
4.6 Conclusion

This chapter inspects the analysis and design phase of this project. Firstly, the chapter discusses the use case diagram for the system which capture each function can be performs, and shows the use cases templates to define the use case details such as preconditions, trigger, and post-conditions which simplify to draw the activity diagrams. Secondly, activity diagram shows the events flow execution and the circumstances that originate those events for the whole system and the complex use cases include Send Message, and Manipulate Inbox use case. Thirdly, class diagram shows the relationships and the structural connections between the system classes to define how these classes will communicates and interacts to build up the system, as well as the attributes and methods for each class include Elliptic Curve, EC Helper, Message, and Contacts. Fourthly, sequence diagrams displays how the classes communicate and behave to accomplish the complex use cases, operations, and functions which is sending message, reading message, and deleting message.
CHAPTER 5
ECSMS IMPLEMENTATION AND TESTING

5.1 Introduction

The implementation and testing of the ECSMS performed using the hardware and software mentioned in chapter 3. This chapter highlights the implementation phase with a segment of code and pseudo code for such important functions, as well as the testing of the application in terms of system functionality and performance.

5.2 Implementation

5.2.1 NetBeans IDE Mobility

NetBeans Mobility is an extension to NetBeans IDE, which support for the MIDP, the CLDC, and the wireless messaging APIs, which use to develop and test the ECSMS as well as applications for the J2ME enabled mobile devices.

Configurations and profiles are the two main building blocks in J2ME. The CLDC is defining the minimum set of Java virtual machine features and Java class libraries for mobile devices, where are two types of classes defined in the configuration level: classes inherited from J2SE such as the java.lang, java.io, and java.util packages, and classes designed specifically for the needs of small devices where are defined in the javax.microedition.io package (Sun Microsystems, 2000)

The second block is MIDP is defining additional APIs and class libraries to access specific functionality such as the graphical user interface, and network communication. The package named javax.microedition.midlet is included primarily for defining how MIDP application should be organized and how they should interact with their environment. It has one class called MIDlet, which implements the methods: startApp, pauseApp, destroyApp,
notifyDestroyed, notifyPaused, getAppProperty, and resumeRequest to define the key behavior of the application (Knudsen, 2002).

As the ECSMS application means for SMS the WMA package used to send and receive SMS's which included in the NetBeans IDE Mobility package. WMA is a J2ME optional package based on the Generic Connection Framework (GCF); which is useful set of network and input output classes for the J2ME, and it is contains specialized APIs that can be added to enable an application based on the MIDP to send and receive wireless messages, all the WMA components are contained in a single package, javax.wireless.messaging, which defines all the interfaces required for sending and receiving wireless messages (Ortiz, 2002).

In the WMA, the class javax.microedition.io.Connector is used for creating a connection, and the javax.wireless.messaging.MessageConnection class has methods for sending and receiving SMS messages. Sending a message (byte array bmsg) as described below in figure 5.1.

```java
String address = "sms://" + c.getMsisdn() + ":" + PORT;
MessageConnection smsconn = null;
smsconn = (MessageConnection) Connector.open(address);
BinaryMessage bmsg =
(BinaryMessage)smsconn.newMessage(MessageConnection.BINARY_MESSAGE);
bmsg.setAddress(address);
bmsg.setPayloadData(bmsg);
smsconn.send(bmsg);
smsconn.close();
```

Figure 5.1: Sending a message through MessageConnection.
When receiving an SMS message, the application has to open a listening connection for incoming messages. The Application uses the Push Registry mechanism to enable the MIDlet to be lunched for new inbound SMS connection as shown in figure 5.2.

```java
String connections[];
connections = PushRegistry.listConnections(true);
if (connections.length != 0) {
    for (int i = connections.length - 1; i >= 0; i--) {
        try {
            MessageConnection mc = (MessageConnection)
            Connector.open(connections[i]);
            mc.setMessageListener(this);
            LISTENERS.addElement(mc);
        } catch (java.lang.SecurityException ex) {
        } catch (java.io.IOException ex) {
        }
    }
}
```

Figure 5.2: Push Registry for new inbound SMS.

Sent and received messages are stored using the record store. The Record Store (javax.microedition.rms) is a package containing classes used for permanent storage. Storage is opened with `openRecordStore` method, while it is closed with `closeRecordStore` method. Records in the store can be iterated using the `enumerateRecords` method, as well as a new record can be inserted with the `addRecord` method.

Furthermore, NetBeans Mobility package provides a rapid wizard to create a MIDP projects using VMD, where it gives the ability to layout the flow and the screens design of the application graphically and automatically creates the code for the application as shown in figure 5.3 and 5.4.
Figure 5.3: The Flow of the ECSMS application.

Figure 5.4: The Screens Design of the ECSMS application.
On the other hand, the NetBeans IDE Integrated Obfuscation and Optimization support, which upgrades the performance and optimizes the size of the application as shown below in figure 5.5.

5.2.2 The Legion of Bouncy Castle Cryptography API's

Since the Legion of Bouncy Castle for J2ME is an open source lightweight package that contains an arsenal of cryptographic algorithms and as well as the Elliptic Curve Cryptography, it will be used to implement the cryptographic part in the ECSMS application.
The package `org.bouncycastle.math.ec` contains the classes for the elliptic curve arithmetic, it is consists of the following classes: The class `ECConstants`. The abstract class `ECCurve`. The abstract class `ECFieldElements`. The abstract class `ECPoint`.

5.2.2.1 Confidentiality

The ECSMS provides confidentiality through the use of the Elliptic Curve Integrated Encryption Schema (ECIES), which provide secure channel in the presence of an active adversary to prevent unauthorized disclosure. The package `org.bouncycastle.crypto` contains the classes for the ECIES arithmetic such as: `engines.IESEngine`, `params`, `macs.HMac`, `BasicAgreement`, and `DerivationFunction`.

5.2.2.2 Integrity

Data integrity ensures that the message content is not altered during transmission. The ECSMS application provides this feature through the use of the Elliptic Curve Digital Signature Algorithm (ECDSA). After the application decrypts the new SMS, it will check the signature if valid or not, if not valid it will reject the SMS. The packages `org.bouncycastle.crypto`, and `org.bouncycastle.math` contains the class for Digital Signature Generation and Verification such as `crypto.signers.ECDSASigner`, and `math.ec.ECPoint`.

5.2.2.3 Authentication & Non-repudiation

In order for users to know whether or not a public key does belong to the purported entity, the ECSMS use a digital certificate issued and signed by certificate authority to provide authentication and non-repudiation. So that users are assured that the public key to which they are encrypting SMS is in fact the public key of the intended recipient and not a
forgery. The packages `org.bouncycastle.x509` and `org.bouncycastle.asn1` contains the classes for certificate generation.

### 5.2.3 Application Graphical User Interfaces

After designing the system in terms of use cases, classes, and objects, come up the final part which is the system Graphical User Interfaces (GUI) representations. The GUI has to provide the entire system functionality, as well as it should be user-friendly and provide the software quality attributes such as usability. Figure 5.6 present the GUI for creating account and login operation while Figures 5.7, 5.8, 5.9, 5.10, 5.11 show GUI for Main List and Messaging list, Settings and Change Login screen, Inbox and Outbox lists, Contacts and Statistics lists, and Generate Key screen respectively.

![GUI for Account Creation and Login.](image)

Figure 5.6: GUI for Account Creation and Login.
Figure 5.7: GUI for Main Menu and Messaging List.

Figure 5.8: GUI for Setting and Change Login Screen.
Figure 5.9: GUI for Inbox and Outbox List.

Figure 5.10: GUI for Contacts and Statistics List.
5.3 Testing

The testing of ECSMS application is performed in the lab with the hardware and software stated in section 3.5.1 and 3.5.2 respectively to ensure that the application achieves the system requirements. Testing consists into four types which are system installation, functionality, performance, and reliability.

5.3.1 ECSMS Installation Testing

The installation of the ECSMS onto the mobile phone performs through the cable using the Nokia PC Suite. After selecting the NetBeans IDE deployment method to be “Nokia Terminal connected via PC Suite” as shown in Figure 5.12, and connected the mobile phone to the PC the installation done through initiated the NetBeans to deploy the application into the mobile as shown in Figure 5.13.
Figure 5.12: NetBeans IDE Deployment Methods.

Figure 5.13: ECSMS deployment.
5.3.2 System Functionality Testing

The testing in this section was for the main activities and functions of the application. These include account creation, sending and receiving SMS, key generation, access the inbox and outbox records, and changing the settings. In order to start using the application, creating account with username and password have to be done to authenticate the user as shown in Figure 5.14.

![Figure 5.14 Account Creation and Login.](image)

The ECSMS application main page consists of five essential functions: Firstly, "Messaging" is the main function of the application where the messaging service can be done. The Messaging list consists of three functions: Compose, Inbox, and Outbox. Sending a new SMS done through Compose function, after the user selecting the recipient and composing the message content as shown in Figure 5.15, Figures 1-4, the application proceed with the cryptographic operations automatically. In order to send the message successfully, the cryptographic operations also must be done successfully as shown in Figure 5.15, figures 5-12.
Figure 5.15: The testing process of sending message operation.
The *Inbox* page is a store of all inbound messages where can *Open, Delete, and view Details* of all received messages as shown in Figure 5.16 figure(1), and Figure 5.16, figures (2-4) shows the message details includes sender, received date, encrypted message, signature, sender public key, decryption time, and signature verification time.

![Figure 5.16: Testing the inbox functionality.](image)

While there is a new message the application promoted to start as well as notified in the main page, messaging page, and inbox page that there is new message (Figure 5.17, figures 1-4). Figure 5.17, figures 5-12 shows the operations that the application did to insure the message authentication, confidentiality, integrity, and non-repudiation.
Figure 5.17: Testing the process of reading new message.
The *Outbox* page is a store of all outbound messages, where can *Open*, *Delete*, and view *Details* of all sent messages as shown in Figure 5.18-1, Figure 5.18, figures 2-4 shows the message details includes recipient, sent date, encrypted message, signature, receiver public key, encryption time, and signature generation time.

![Outbox and Out Message Screenshots](image.png)

Figure 5.18: Testing the outbox functionality.

Secondly, "**Contacts**" is a record list for contacts; after the account had been created the application read the certificates given by the CA and saved their details in the contact list. Where the **Contacts** page is a list of all contacts sorted by their names as shown in Figure
5.19-1, as well as their details which are the contact name, mobile number, public key, and certificates expire date as shown in Figures 5.19-2 and 5.19-3.

![Image of contacts records](image1.png)

Figure 5.19: Testing the contacts records.

Thirdly, "Keys" is the page where different key pairs can be generated as shown in Figure 5.20. Fourthly, "Statistics" is the page where it saves the average time of all cryptographic operations includes encrypting, decrypting, signature and key pair generation, and signature verification as shown in Figure 5.21.

![Image of key pair generation](image2.png)

Figure 5.20: Testing the process of key pair generation.
Fifthly, "Settings" is the page where the username and password can be changed as shown in Figure 5.22-1. In order to change the username and password successfully, the current username and password must correctly entered, as well as the new username selected and the new password selected and correctly confirmed as shown in Figures 5.22-2, 5.22-3, and 5.22-4.
5.3.3 Application Performance Testing

The application performance testing focused on measuring the time needed to perform the cryptographic operations include key generation, message encryption, message decryption, signature generation, and signature verification. The execution time of the cryptographic operations depends on the key size where small key sizes result in fast execution and long key size result in slow execution time. Testing involved for two key lengths 160-bits length, which equals to RSA/DSA 1042-bits length, and 192-bits length, which equals to RSA/DSA 1536-bits length. Figure 5.23 shows the snapshot of statistics.
after testing the ECSMS using Nokia N80 phone. The average execution time for the cryptographic operations for both keys is shown in Table 5.1. While Figure 5.24 showed the histogram for the differences of cryptographic operation execution time between 192-bit key size and 160-bit key size.

![Statistics](image)

Figure 5.23: 1-Statistics of 192-bit key size, 2-Statistics of 160-bit key size.

Table 5.1: Average execution time for the cryptographic operations.

<table>
<thead>
<tr>
<th>Key Size (in bits)</th>
<th>Operation (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Key Generation</td>
</tr>
<tr>
<td>192</td>
<td>2.277</td>
</tr>
<tr>
<td>160</td>
<td>1.595</td>
</tr>
</tbody>
</table>
5.3.4 Reliability Testing

The reliability of the application depends on the mobile phone and network. The ECSMS application developed using Java programming language (J2ME), so the phone should be Java-enabled mobile phone supported for MIDP 2.0 and CLDC 1.1 including all Symbian phones to run the application. Where different kinds of mobile devices used to run and test the application such as Nokia N97, Nokia N80, Nokia E71, and Sony Ericson where all have been done successfully. Since the ECSMS application means for messaging service and most of the mobile network provider supports either GSM or UMTS. The application has been tested using different mobile networks including Maxis, Celcome, and DiGi in Malaysia, and Orang and Zain in Jordan, where those operators supports for short messaging service, where the testing is focused on sending and receiving messages via local networks and international roaming which done successfully.
5.3.5 Discussion

Various testing have been performed on the ECSMS application includes system installation, system functionality, system performance, and system reliability. System installation has been done successfully using the NetBeans IDE deployment method through the PC Suite that connected the mobile phone to the PC through cable. All system functions have been tested includes sending and receiving secure SMS, access the inbox and outbox records and their details, generate different key pair, and change the user account setting, where the correspondent results are displayed in diagrams. The system performance have been testing in terms of the time needed to execute the cryptographic operation, which includes key generation, message encryption and decryption, signature generation and verification using two different key sizes, where the testing achieved acceptable performance time. Besides, the system have been tested on different mobile networks includes GSM and UMTS which both supported for SMS.

5.4 Summary

This chapter highlighted the most important components for implementation and testing parts. The implementation tool, packages and classes, pseudo code for main functions, the cryptographic algorithms, and application GUI have been discussed. Various testing has been performed on the application including the system installation, system functionality, system performance, and reliability, where the time consumed by the operations of key pair generation, message encryption and decryption, and signature generation and verification is presented, as well as implementation on java-enabled phones and messaging service successfully tested in GSM and UMTS network.
CHAPTER 6
DISCUSSION & CONCLUSION

6.1 Introduction

The SMS security concern in the early stages of the messaging service implementation was not important beside the reliability, adaptability, and usability. Nowadays, the persistent handling of short messaging service raised the security concern since their applications need to send private and confidential information.

Currently, the short message services provide only some security services and the messages stored in network provider unprotected. Therefore, many applications have been developed to solve SMS security issues using either symmetric or asymmetric key cryptography.

Since the ECC provides strong security level and acceptable performance using small key size, the main objective of this research was to implement end-to-end secure mobile messaging application using the ECC to provides message confidentiality, integrity, authentication and non-repudiation, as well as protecting the saved messages in case the phone lost or stolen.

6.2 Accomplished Objectives

The accomplished objectives are:

- Implementing end-to-end secure application that is capable of sending a private and confidential SMS using PKC which provides the four security services (confidentiality, integrity, authentication, and non-repudiation).

- Enhancing the SMS security level by using the ECC, whose security relies on the difficulty of the ECDLP that allows the delivery of the highest strength per bit of all previously known public key cryptosystems.
- Enhancing the performance by using the ECC with 160-bit key size, the result of which will be smaller key sizes, bandwidth savings, and faster implementation.
- Protecting the saved messages in case the phone lost or stolen. It is achieved by protecting the application with a username and password, where the user with correct username and password can access the application as well as the saved messages.

6.3 Significance of the Study

The elliptic curve cryptosystem provides the strongest public key schema of any previous public key cryptosystems as it relies on the strongest cryptographic mathematical problem the ECDLP. On the other hand ECC provides the same level of cryptographic security as DSA and RSA systems using shorter key sizes, which contribute to minimize the processing power, increase the implementation speed, decrease the certificate size, and make it suitable for small processors such as the mobile devices. As a result the ECSMS application implemented using the ECC to secure the SMS routine.

6.4 Conclusion

Nowadays, the number of mobile subscriber connections increases daily, and billions of SMS are sent every day. Since the mobile users needs to exchange and send private and confidential messages, there is demand to secure the mobile SMS communication channel. Many applications and solutions have been developed to solve SMS security issues using symmetric or asymmetric key cryptography. Symmetric key cryptography solutions have been developed in view of the fact that it is considered simple, fast, and solved the mobile phone processing power limitations. Nevertheless it has a problem with key transformation process. Although, asymmetric key cryptography provides security level more than
symmetric one using two different keys; private key is kept securely by the owner for decryption and public key for encryption which can be made public. However, implementing it on the mobile devices achieved unacceptable performance since it used long keys size. Therefore, the asymmetric solutions have been developed based on small key size to achieve acceptable performance. While the security strength is based on the key size; key pair with higher bit is more difficult to be broken compared to those in lower bit.

The ECC came to solve the low speed in all known public key cryptography, which provides the same security level using smaller key size. This research applied the public key cryptography to the SMS where the ECC have been used to implement the four security services to provide end-to-end secure mobile short messaging services. The system interface is designed based on the system functionality and design flow to meet the defined requirement. The overall flow of interface is satisfied in term of usability and interaction between user and application. The entire system functionality has been tested in a complete manner as well as the performance.

It is quite normal that systems enjoy strengths and suffers from weaknesses. While the ECSMS application provides secure environment to send and receive confidential messages, it has certain weaknesses that could be mentioned includes the cryptographic operations increased the message size where the message exceeded the GSM standard which is 160 bits per message, therefore the message split into two messages. Besides, the application supports English language only.

6.5 Future Works

Proposals for enhancements the ECSMS can be made in order to improve the features to make it more efficient or adding new features. Enhancing the message size by applying one of the compression techniques to make it one message instead of two messages will
save the cost, as well as upgrade the application to support multi-language such as Arabic, Bahasa Melayu, and Chinese to increase the number of users from different geographical areas. Introducing biometric authentication features such as fingerprint or voice recognition to protect the saved messages from unauthorized persons, in the other hand can add a remote locking feature in case the phone is lost or stolen where the owner sends a lock message in such cases to lock the application or the phone.

The ECSMS is an ECC solution means for short messaging service, it can be upgraded and transformed for various wireless applications such as mobile payment system, and mobile banking. Also applying the PKC to the voice and video communication will be interesting and meaningful.
REFERENCES


Edoh, K.D. (2004). Elliptic curve cryptography: Java implementation. *InfoSecCD Conference ’04, Kennesaw, GA, USA*


Hassinen, M. Markovski, S. (2003). Secure SMS messaging using Quasigroup encryption and Java SMS API. IEEE.


Nokia Corporation, 2003, A brief introduction to secure SMS messaging in MIDP, *forum.nokia.com*


