Achieving Security in Messaging and Personal Content in Symbian Phones

Ahmed Enany
This thesis is submitted to the Department of Interaction and System Design, School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author: Ahmed Enany
E-mail: ahmed9200@hotmail.com

University advisors:
Professor Bo Helgeson
Department of Interaction and System Design
School of Engineering
Blekinge Institute of Technology

Professor Rune Gustavsson
Department of Interaction and System Design
School of Engineering
Blekinge Institute of Technology

Department of Interaction and System Design
School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet: www.bth.se/tek
Phone: +46 457 38 50 00
Fax: +46 457 271 25
Abstract

This thesis describes two proposed schemes that could be used to secure mobile messaging (SMS/MMS) as well as one scheme that could be used to secure mobile content. The security services we considered in securing the mobile messages are confidentiality, authentication, non-repudiation and integrity. We used Identity Based Cryptography in order to secure the mobile messaging and Blowfish algorithm to secure the mobile content. Due to some of the disadvantages imposed by the Identity Based Cryptography, we recommended using it along with the RSA algorithm. The proposed schemes were implemented in java and tested on an actual device, Nokia N70. In addition, we measured the time required by each of the algorithms we used to encrypt/decrypt a certain number of bytes. We found that the time taken by RSA and Blowfish algorithms will not be noticeable by the user. However, since the implementation of the Identity Based Cryptography we used was not meant to run on mobile devices, we encountered a noticeable delay whenever encrypting/decrypting the data using this algorithm. Securing the SMS messages will make it to be considered as one of the proposed means that could be used to conduct m-commerce. In addition, securing the MMS messages and the mobile content will increase the usability and the reliability of the mobile phones especially to the users on the move.

Keywords: Security, Symbian, J2ME, authentication, confidentiality, non-repudiation, integrity, IBE, IBS, IBC, SMS, MMS
Acknowledgments

I would like to thank "GOD" almighty for his favors that were obvious during this work. Also I would like to thank my family for their support, Prof. Rune Gustavsson for his fruitful guidelines, Prof. Bo Helgeson for his efforts and help to ensure the completion of the work on time and Dr. Henric Johnson for his encouraging comments on the thesis. It is said that "a friend in need is a friend indeed" so, I would like to thank my friends that supported me during this work especially and warmly Khalil Abu-Elsheikh and Irina Voronovich.

Ahmed Enany
Ronneby, Sweden
February, 2007
# Table of Contents

Abstract .................................................................................................................................................. iii  
Acknowledgments ............................................................................................................................... iv 
Table of Contents ................................................................................................................................. v 
Abbreviations ........................................................................................................................................ vi 
List of Figures ........................................................................................................................................ viii 
List of Tables .......................................................................................................................................... ix  
 **Chapter 1 : Introduction** .................................................................................................................. 1  
1.1. Background ..................................................................................................................................... 1  
1.2. Problem Definition ....................................................................................................................... 2  
1.3. Research Questions ....................................................................................................................... 4  
1.4. Research Approach ....................................................................................................................... 4  
1.5. Contribution of Thesis ................................................................................................................. 4  
1.6. Related Work ............................................................................................................................... 4  
1.7. Outline of the Thesis .................................................................................................................... 6  
 **Chapter 2 : The Symbian OS** .......................................................................................................... 7  
2.1. Symbian OS Overview .................................................................................................................. 7  
2.2. Symbian OS Architecture ........................................................................................................... 8  
2.3. User Interface Platforms .............................................................................................................. 13  
2.4. Developing Applications under Symbian OS .......................................................................... 13  
 **Chapter 3 : Security Services Overview** ..................................................................................... 15  
3.1. Security Services ......................................................................................................................... 15  
3.2. Public Key Authentication ........................................................................................................... 20  
 **Chapter 4 : Securing SMS, MMS and the Mobile Content** ............................................................ 25  
4.1. Achieving Security in Symbian Phones ....................................................................................... 25  
4.2. Implementation of the Proposed Schemes ................................................................................. 32  
4.3. Discussions ............................................................................................................................... 37  
 **Chapter 5 : Conclusion** ............................................................................................................... 41  
 **Chapter 6 : Future Work** .............................................................................................................. 43  
References ............................................................................................................................................ 44
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DES</td>
<td>Triple Data Encryption Standard</td>
</tr>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>3GPP2</td>
<td>3rd Generation Partnership Project 2</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARM</td>
<td>Advanced RISC (Reduced Instruction Set Computer) Machine</td>
</tr>
<tr>
<td>ATM</td>
<td>Automatic Teller Machine</td>
</tr>
<tr>
<td>CDMA</td>
<td>Code Division Multiple Access</td>
</tr>
<tr>
<td>CLDC</td>
<td>Connected Limited Device Configuration</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>EDGE</td>
<td>Enhanced Data Rates for GSM Evolution</td>
</tr>
<tr>
<td>EMS</td>
<td>Enhanced Messages</td>
</tr>
<tr>
<td>FileCF</td>
<td>File Connection Framework</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GDI</td>
<td>Graphics Device Interface</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Services</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
<tr>
<td>HMAC</td>
<td>Keyed-Hash Message Authentication Code</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>IBC</td>
<td>Identity-Based Cryptography</td>
</tr>
<tr>
<td>IBE</td>
<td>Identity-Based Encryption</td>
</tr>
<tr>
<td>IBS</td>
<td>Identity-Based Signature</td>
</tr>
<tr>
<td>ICL</td>
<td>Image Conversion Library</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPSec</td>
<td>IP security</td>
</tr>
<tr>
<td>IrDA</td>
<td>Infrared Data Association</td>
</tr>
<tr>
<td>J2ME</td>
<td>Java Micro Edition</td>
</tr>
<tr>
<td>JCA</td>
<td>Java Cryptography Architecture</td>
</tr>
<tr>
<td>JSR</td>
<td>Java Specification Request</td>
</tr>
<tr>
<td>JTWI</td>
<td>Java Technology for the Wireless Industry</td>
</tr>
<tr>
<td>LCD</td>
<td>Liquid Crystal Display</td>
</tr>
<tr>
<td>m-commerce</td>
<td>Mobile Commerce</td>
</tr>
<tr>
<td>MD5</td>
<td>Message Digest 5</td>
</tr>
<tr>
<td>MIDlet</td>
<td>Mobile Information Device Application (or Applet)</td>
</tr>
<tr>
<td>MIDP</td>
<td>Mobile Information Device Profile</td>
</tr>
<tr>
<td>MIME</td>
<td>Multipurpose Internet Mail Extensions</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>MMF</td>
<td>Multimedia Framework</td>
</tr>
<tr>
<td>MMS</td>
<td>Multimedia Messaging Service</td>
</tr>
<tr>
<td>MNO</td>
<td>Mobile Network Operator</td>
</tr>
<tr>
<td>MSL</td>
<td>Media Support Library</td>
</tr>
<tr>
<td>OBEX</td>
<td>OBject Exchange</td>
</tr>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
<tr>
<td>PIM</td>
<td>Personal Information Manager</td>
</tr>
<tr>
<td>Pixel</td>
<td>Picture Element</td>
</tr>
<tr>
<td>PKI</td>
<td>Public Key Infrastructure</td>
</tr>
<tr>
<td>PPP</td>
<td>Point to Point Protocol</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time transport protocol</td>
</tr>
<tr>
<td>RTT</td>
<td>Radio Transmission Technology</td>
</tr>
<tr>
<td>SHA-1</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td>SMIL</td>
<td>Synchronized Multimedia Integration Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>vCard</td>
<td>Versitcard (Electronic Business Card)</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>W-CDMA</td>
<td>Wideband Code Division Multiple Access</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1.1: Typical MMS Message Format [30]................................................................. 2
Figure 2.1: Symbian Ownership [20]. .............................................................................. 7
Figure 2.2: Symbian OS Licensees [20]. ........................................................................... 8
Figure 2.3: Symbian OS v9.1 Architecture [16].................................................................. 9
Figure 3.1: Symmetric Cryptography............................................................................... 15
Figure 3.2: Asymmetric Cryptography............................................................................. 16
Figure 3.3: Message Authentication............................................................................... 17
Figure 3.4: Message Integrity (With Authentication). ...................................................... 18
Figure 3.5: Sending a Message Maintaining All the Security Services. ......................... 19
Figure 3.6: Public Key Certificate.................................................................................... 20
Figure 3.7: Public Key Infrastructure............................................................................... 21
Figure 3.8: Identity Based Cryptography........................................................................ 22
Figure 4.1: Sending a Signed RSA Public Key ............................................................... 26
Figure 4.2: Secure SMS Message.................................................................................. 27
Figure 4.3: SMS Message Header.................................................................................. 27
Figure 4.4: Secure MMS Message.................................................................................. 29
Figure 4.5: Secure MMS Object.................................................................................... 30
Figure 4.6: MMS Object Status Header........................................................................... 31
Figure 4.7: Secure Repository........................................................................................ 32
Figure 4.8: Secure SMS Main Screen............................................................................ 33
Figure 4.9: Secure SMS Verbose Output....................................................................... 34
Figure 4.10: Secure MMS Main Screen......................................................................... 34
Figure 4.11: Secure MMS Object Verbose Output.......................................................... 35
Figure 4.12: Create a New Folder.................................................................................. 36
Figure 4.13: Adding a File to the Folder......................................................................... 36
Figure 4.14: Retrieving Files from the Folders............................................................... 37
List of Tables

Table 4.1: (KU), (A) and (C) Bits Combinations and Their Meanings................................. 28
Table 4.2: (AA) and (AC) Bits and Their Corresponding Algorithms................................. 28
Table 4.3: (AA), (CAK) and (CAO) Bits and Their Corresponding Algorithms................. 31
Chapter 1: Introduction

The number of GSM (Global Systems for Mobile communication) subscribers hit the two billion milestone in June 2006 [1]. Thus GSM has become the first communication technology to have more users in the developing world than the developed world with more than 690 mobile networks providing GSM services across 213 countries [1].

1.1. Background

1.1.1. Short Message Service (SMS)

One of the most vital GSM services is SMS (Short Message Service) messaging service. The estimates of the SMS messages number sent in 2005 worldwide were 1 trillion messages [21]. SMS is used to exchange short text messages, up to 160 Latin characters (letters, numbers or symbols) using 7-bit character encoding or 70 characters using 16-bit Unicode character encoding otherwise, seemingly with any other GSM subscriber around the world.

SMS was invented as a simple messaging system that works when the user's mobile is turned off or out of coverage [3]. In addition, it was perceived at the beginning as a service that notifies the users of voice mails [2]. The usage of SMS messages includes but is not limited to advertising, sending reminders (e.g. medical, payment, appointment etc.), personal communication, receiving information (e.g. bank account balance, weather forecasting, horoscopes etc.) and paying for services (e.g. tickets booking, subscribing to some service offered by the network, purchasing an application to be installed on the mobile etc.). Nowadays the benefits of SMS messages are undeniable. Usually it is the cheapest way, especially if we consider sending overseas SMS messages, to contact or send information to the others among the other available GSM services provided by the Mobile Network Operator (MNO).

Payments using SMS are usually done using a premium SMS message which is a normal SMS message but usually costs more than the regular fees, and is considered to be a feasible solution to collect micropayments, small amount of money that collecting it using the regular methods could be expensive. However premium SMS could be a risky solution [4].

1.1.2. Multimedia Messaging Service (MMS)

MMS messages are considered to be an extension to SMS messages so they are not only limited to plain text content but can also include multimedia content (images, audio, video etc.). In addition, the message capacity is not limited to just 140 bytes, however it is usually limited by the MNO or the mobile phone but at the same time allowing enough capacity to send media objects.

MMS messages resemble the email MIME format in which each MMS message consists of headers that contain the message properties and a body that could virtually contain any media objects [30]. However, the common objects that could be included in a MMS message are pictures, sounds, videos and plain text. Moreover, a presentation object is
usually included that instructs how the other media objects should be presented on the mobile phone. One common presentation standard is SMIL (Synchronized Multimedia Integration Language) which is a known standard used in multimedia presentations. If no presentation object is defined in the MMS message, it is up to the mobile phone specific implementation to determine how the media objects should be presented on the screen. Figure 1.1 shows a typical MMS message format.

![Figure 1.1: Typical MMS Message Format](image)

1.1.3. Mobile Content

Mobile phones are now capable of storing data that could take the form of images, video files, sound files, documents etc. The storage capacity of the mobile phones could reach, in some models, up to a number of Gigabytes. Further to, this capacity could be usually extended using an external solid state memory cards. The availability of good enough storage space along with the versatility of the services that could be provided by the smartphones are one of the most important features that benefit the users on the move.

1.2. Problem Definition

The current SMS/MMS messaging services inherently provide no security measures that could be used to send and receive messages that show mutual commitment i.e. making transactions, speculating etc. SMS messages are usually sent as encoded text [5] that could be decoded with a little effort and time. Therefore, SMS cannot be trusted in conducting m-commerce transactions not mentioning that many people consider SMS messages as private messages that require privacy [6]. The same thoughts also hold in the MMS messages which are not only tied to sending private text but private images and videos as well. In addition, SMS/MMS messages are prone to be sent by mistake to the
wrong recipient(s) i.e. it is common to enter the recipient mobile number incorrectly or choose incorrect recipient from the phonebook when sending a SMS/MMS message.

Usually the mobile user reckons his/her mobile content as private. Mobile content may include SMS messages, MMS messages, images, videos, phone book, documents etc. At the time of writing, there are no built-in security features included in the mobiles that could aid to maintain the privacy of the mobile content i.e. no security features in the underlying file system, no built-in secure phone book, no built-in secure gallery (images/videos storage) etc. Not to mention that the mobile phone could be stolen thus, exposing all the stored content (messages and personal data) to fall into the wrong hands i.e. an adversary, competitors etc. Therefore, the need for securing mobile content seems to be crucial.

Mobiles are usually equipped with security lock key that requires the user to enter this specific key every time the mobile is rebooted or after some period of inactivity, but such security is usually useless since this key could be obtained or reset by any amateur with the right equipment. In addition, most mobiles that are equipped with external memory that provides a functionality by which the user can lock this memory using some password, but again it is possible to unlock such memory using the right equipment.

In the light of the above discussion, it is clear that SMS/MMS services need to have some security measures in order to create a trusted communication channel between the SMS/MMS sender and the receiver. This will lead to increasing the reliability of the messaging services in conducting m-commerce transactions and exchanging personal messages with no fear of any other third party interference. In addition, the mobile content should be protected against unauthorized access for example, in case of mobile theft or being infected with a backdoor.

In order to have an end to end secure communication channel, the following security services should be provided: authentication, confidentiality, non-repudiation and integrity. Authentication is to assure the recipient that the message comes from the source that it claims to be from. Confidentiality is to protect the data from unauthorized disclosure. Non-repudiation prevents both the sender and the receiver from denying/disowning a transmitted message. Integrity means that the message is received as sent with no modifications to the original message.

Implementing the previously mentioned security services could be done by the MNO but this might incur incompatibility issues with the other MNOs i.e. in case of roaming. In addition, the content of the message should be protected from disclosure even to the MNO. On the other hand, the security services could be implemented by the mobile manufacturers but again this might raise incompatibility issues that may result in that phones from two different manufacturers could not exchange SMS/MMS messages between each other. Therefore, the solution needs to be a third party solution that depends on neither the MNO nor the phone manufacturer.

Today’s mobile phones functionalities could be extended easily by third party applications hence; the name smartphones. Those applications usually are developed in java (J2ME) or C++ (under Symbian OS). Almost all the recent mobile phones support J2ME one way or another but in order to develop in C++ the mobile should be
running the Symbian OS as its operating system. Therefore, the security services could be implemented as third party applications and installed seamlessly on the mobile phones.

Considered as embedded systems, mobile phones have different limitations, namely limited memory and limited processing power. Knowing that most of the security services would be implemented using encryption algorithms, this incurs a challenge to the developers in that they should be wise when choosing such algorithms that could be run gracefully on mobile phones thus, providing the desired functionalities and not violating the environment constrains.

Achieving such trusted channel between the two parties that exchange the SMS message will open the door to many m-commerce applications that will not be limited to collecting just micropayments, which is dominant at the time of writing. In addition, mobile users would be able to exchange private SMS/MMS messages with no fear of the message content could be disclosed to any third party as well as maintaining the privacy of the data stored on the mobile.

1.3. Research Questions

In this thesis we will try to answer the following questions:

- How could security services be implemented in mobile messaging?
- How could mobile content be secured?
- Which algorithm(s) should be used to implement the security services in mobile messaging, and will it/they affect the usability in terms of the time taken to send/receive secure messages?
- Which algorithm(s) should be used to secure the mobile content, and will it/they affect the usability in terms of the time taken to secure the mobile content?

1.4. Research Approach

We will investigate the state of the art algorithms/configurations that are used to provide security services to the ordinary mailing systems and whether it is possible to be applied to the mobile messaging. Moreover, we will inspect what changes should be made to the SMS/MMS messages format in order to meet the needs imposed by the security services.

1.5. Contribution of Thesis

The main contribution of the thesis is the two proposed schemes that could be used to provide security services to mobile messaging (SMS/MMS) as well as one scheme that could be used to maintain the privacy of the mobile content. The three schemes are introduced in detail in section 4.1 along with their implementation in section 4.2.

1.6. Related Work

Some of the work done in this area will be discussed in this section. We focused only on the work that was provided as a third party solution i.e. not dependant on neither a
specific network operator nor a specific phone manufacturer since such solutions might incur incompatibility issues as stated previously.

M. Hassinen [10] developed a solution using J2ME in order to provide confidentiality, integrity and authentication in SMS messaging. He used two symmetric encryption algorithms (Quasigroup and Blowfish) to provide the security functions to the SMS messaging. The sender can use either Quasigroup or Blowfish in order to encrypt the message and then send it to the receiver who can decrypt it using a secret key that is known only to the two parties involved in the message exchange hence, ensuring the confidentiality of the message. In order to ensure the authenticity and the integrity of the message the SHA-1 algorithm is used to calculate a MAC (Message Authentication Code) from the plain text, append it to the message and finally encrypt and send it. The receiver will then decrypt the message and check its integrity by calculating the MAC of the decrypted text and compare it with the appended MAC. Certain values of the SMS messages are encrypted in order to store the message on the mobile phone. Therefore, the user will be asked to provide a password whenever reading a stored message is required. The author stated that a protocol like Diffie-Hellman or the like could be used to exchange the secret key between the SMS messages exchangers.

Another work done by M. Hassinen [11] in which a solution was developed using J2ME based on Public Key Infrastructure (PKI) to provide confidentiality, integrity and non-repudiation to SMS messages. He used the RSA algorithm for message encryption/decryption and SHA-1 algorithm to calculate the hash value that is used in both verifying the message integrity and creating the digital signature. The Finnish government PKI, FINUID, was used to authenticate the users to the system using smart cards issued by the population register center in Finland. The main reason for this authentication is to generate a certificate for the public key of the client.

For (A) to send a message to (B), (A) should get (B)'s certificate from the authentication server (AS) in case no previous communications were done between them. This certificate includes (B)'s public key. (A) will send (B) the message which contains \(E_{PKB}(M) | TS_A | SIG_A\) where:

- \(E_{PKX} \rightarrow X's \text{ public key.}\)
- \(M \rightarrow \text{the message.}\)
- \(M' \rightarrow \text{the encrypted message.}\)
- \(TS_X \rightarrow X's \text{ time stamp (used to protect against replay attacks).}\)
- \(E_{SKX} \rightarrow X's \text{ private key.}\)
- \(H \rightarrow \text{the used hash function (SHA-1).}\)
- \(SIG_A \rightarrow X's \text{ signature } = E_{SKA}(h) \text{ where } h = H(E_{PKB}(M) | TS_A).\)

When (B) will receive the message, he/she will retrieve A's certificate if he/she does not have it from a previous communication and then check whether \(H(M' | TS'_A) = E_{PKA}(SIG_A)\) this to ensure that the message \(M\) came from (A) not anyone else, and to ensure
that the message was not modified as well. In addition, he/she can check whether the difference between $T_{SB}$ and $T_{SA}$ is within the predefined time window in order to ensure this is not a replayed message. Finally (B) can read the message $M = E_{SKB}(M')$.

1.7. Outline of the Thesis

The following chapters will proceed as follows: in chapter 2 we will discuss the Symbian OS platform, in chapter 3 we will discuss how the security services could be achieved in any communication system, in chapter 4 our proposed schemes along with the implementation to secure SMS messages, MMS messages and mobile content will be discussed and chapter 5 includes our conclusion and future work followed by the list of references.
Chapter 2 : The Symbian OS

On November 2006, Symbian Ltd. announced that 100 million Symbian smartphones were shipped to over 250 network operators worldwide [13]. The number of the commercial applications that were developed by third parties, excluding java applications, was 6120 applications in the end of the third quarter of 2006. Symbian is a joint venture between the leading players in the mobile industry market established in 1998 and owned by Ericsson, Nokia, Panasonic, Samsung, Siemens and Sony Ericsson [14] see Figure 2.1. In this chapter, we will give an overview of Symbian OS history and a brief discussion of the OS architecture, the common Symbian User Interface (UI) platforms and developing applications under Symbian OS.

2.1. Symbian OS Overview

Symbian OS is a modular 32-bit preemptive multitasking operating system for mobile devices. It has its own file system, multimedia support, TCP/IP stack and libraries for all communication features that could be found on today’s smartphones [12]. Symbian was developed from the outset as an operating system that targets mobile devices which is not the case in many other operating systems for example, Microsoft smartphone OS is a variation of Microsoft windows OS for PCs.

The history of Symbian OS starts back in mid 1990s in Psion; a company specialized in developing PDAs and organizers. The aim was to develop an operating system that suits devices with limited resources (memory and computation power) and that will run for long time without being shutdown, such operating system suits mobile devices well. EPOC32 or simply EPOC was the name of the operating system developed by Psion and was written in a strong object oriented C++ language [15]. After Symbian was formed, many efforts were exerted in order to make EPOC suitable for various mobile phones [15] i.e. developing the communication protocols, the user interfaces and the other
components of the system. However the modular microkernel based architecture facilitated this task.

EPOC version 5 was the last version of EPOC and was used in Psion 5, a Psion PDA, after that the operating system was renamed to be Symbian. Symbian version 5 was the first version of Symbian OS, which was a Unicode version of EPOC version 5. Symbian version 5 was used in Ericson R380 series mobile phones released in 2000 which is the first Symbian based mobile phone to go on sale to the public. Thereafter Symbian OS version 6.0 was released and used by Nokia 9210 communicator released in 2001. The process of developing Symbian OS is progressive and at the time of writing, the latest version is 9.3.

The process of deploying Symbian OS goes as follows [12]: Symbian develops and licenses the Symbian OS containing the base (microkernel and device drivers), middleware (system servers), communications protocols and a test user interface. The licensees develop their own user interface that meets their products specific needs also; the licensees can license their user interface to other licensees. Figure 2.2 shows a list of Symbian OS licensees

![Symbian OS Licensees](image)

**Figure 2.2: Symbian OS Licensees [20].**

### 2.2. Symbian OS Architecture

Symbian OS provides the licensees with the tools that enable them to speed up the development process of their customized Symbian based mobile phones. In this section, we will give a brief discussion of Symbian OS version 9.1 architecture [16], see Figure 2.3.
2.2.1. UI Framework

Recalling that Symbian OS enables the licensees to create their own customized user interfaces that suit their smartphones thus, saving much of the development time. In addition, this facilitates porting different applications written for a specific UI platform to another e.g. porting an application written for Series 60 (see section 2.3.1) mobiles to UIQ (see section 2.3.3) mobiles. This is applicable since the underlying operating system is the same and the licensees need just to provide their own user interface without modifying the underlying operating system.

Some of the features of the Graphical User Interface (GUI) provided by Symbian OS are:

- An event-driven GUI i.e. some functionality is activated in case of the occurrence of some event.
- A mechanism for the licensee to customize the look and feel of the GUI.
- A plug-in mechanism for the user to input non-standard data (e.g. voice recognition or hand writing recognition for phones with no keypad).
- A flexible screen indicator and status bar framework.
- Runtime changeable color schemes.
- Bitmap animation performed in the window server thread.
- Semi-transparent windows.
- Support for multiple displays.
- Background images i.e. graphics could be drawn behind text with control of parameters like transparency and background scrolling.
- XML Framework with support for XML parsing.

Along with the previously mentioned features Symbian OS supports internationalization in the following sense:
- The framework conforms to the Unicode Standard version 3.0.
- Supports rendering and editing of all European languages, simplified Chinese, traditional Chinese, Japanese, Korean, Arabic, Hebrew and Thai.
- A front-end processor framework is used to enable input of ideographic characters.

### 2.2.2. Application Services

Symbian OS provides the licensees with the core application functionality e.g. calendar, messaging and device synchronization etc. thus, ensuring compatibility between different Symbian mobile phones. Some of the services provided by Symbian OS are:
- PIM services that includes the agenda engine, To-do engine and contacts model.
- Messaging services that provide protocols for sending and receiving SMS, EMS and emails from any application.
- Internet and Web Application Support (HTTP 1.0, 1.1 and subset of WAP 1.2.1).
- Data synchronization services: OMA Data Sync 1.1.2 compliant.

### 2.2.3. Java

Symbian OS implements J2ME MIDP 2.0 and CLDC 1.1 as well as providing the following JSRs:
- CLDC HI 1.1 and MIDP 2.0
- JTWI (JSR185)
- Java API for Bluetooth 1.0 (JSR082), excluding OBEX
- PIM and FileCF (JSR075)
- Wireless Messaging 1.1 (JSR120)
- Mobile media 1.1 (JSR 135)
- Mobile 3D graphics API for J2ME 1.0 (JSR184)

### 2.2.4. OS Services

This layer provides important elements of the operating system infrastructure including multimedia and graphics subsystems, networking, telephony, shortlink protocols (Bluetooth, USB, IrDA etc.), cryptography libraries and PC connectivity infrastructure. It
is worth mentioning that Symbian OS V9.1 introduces many security features that provide a reliable platform that facilitates implementing confidentiality, authentication, and integrity services by third party application developers, some of these features include:

- **Capability Management**: for each process request to access a system resource e.g. the file system, the process must have the appropriate capability to fulfill this request.
- **Data Caging**: enables the applications to have their own private protected data stores that could not be accessed by other applications.
- **Cryptography module**: includes cryptography algorithms (RSA, DES, 3DES etc.), hash functions (MD5, SHA-1, HMAC) and other significant components that is needed to achieve security functions in any application.
- **Software installation system**: provides a quick and secure process of installing third party applications i.e. using digital signatures to authenticate that the installed application was developed by a trustworthy third party.

### 2.2.5. Communications Services

The communication framework provides the system services for communications and networking which includes:

- **Telephony services**: provide multimode API to the users, the abstraction layer for cellular networks provides support for GSM, GPRS, EDGE, CDMA (IS-95), 3GPP2 cdma2000 1x RTT and 3GPP W-CDMA thus, making it easier for mobile manufacturers to port the Symbian OS from one mobile device to another. Such services are used to retrieve information about the phone and the current network, access the SIM card services, access the SIM card phonebook etc.
- **Shortlink services**: provide support to point-to-point communications with other devices. The Current shortlinks supported by Symbian OS are Bluetooth, infrared, USB and Serial.
- **Networking services**: provide communication for wide area network by facilitating the implementation of different communications protocols through a socket interface. Symbian OS networking support includes TCP, UDP, IP v4/v6 stack, ICMP, PPP, DNS, SSL, TLS, IPSec, telnet protocol, FTP, Ethernet support (wired and wireless interfaces) and RTP.

### 2.2.6. Multimedia and Graphics Services

The **MMF (Multimedia Framework)** provides a multithreaded framework to handle multimedia data, which includes:

- Audio/video playing back, recording and streaming through the **MSL (Media Support Library)**.
- Handling still images through **ICL (Image Conversion Library)**.
• Camera support: providing API to deal with the camera functions.
• Speaker dependent speech recognition API.
• Advanced graphics API through the *GDI (Graphics Device Interface)*.

2.2.7. Connectivity Services

This framework is responsible for providing the connectivity between the mobile phone and the PC using the standard TCP/IP protocol for data transfer. Some of the services provided by this framework include:

• Restore/backup operations.
• Application installation from PC.
• Date/Time synchronization.
• Remote file management.

2.2.8. Base Services

This layer provides the programming framework for all other elements of the operating system. The main visible components are the file system and the user library. The Base layer contains the C standard library, relational database and stream store. The file server provides a shared access to the filing system enabling dynamic mounting/dismounting of the file systems. It is worth mentioning that the filing systems provide data integrity in case of sudden power loss.

2.2.9. Kernel Services and Hardware Abstraction Interface

The kernel runs in privileged mode and allocates the memory for itself and user mode processes (processes that run in unprivileged mode). At the time of writing, Symbian OS runs only on ARM processors thus, the kernel runs natively on ARM processors. Some of the main features provided by the kernel include:

• Process, thread, program and memory management.
• Error handling and cleanup framework.
• Descriptors: strings of characters and buffers of binary data.
• Container classes like arrays and lists.
• Active objects, for event-driven multitasking to avoid the overheads of multithreading.
• Client-server architecture, for simple and efficient inter-process communication.
• Silent running mode in which the device can operate with screen switched off.
• Locale support including currency, time and date formatting.
The kernel can be extended by the use of DLLs that can link dynamically against the kernel.

In addition, this layer provides the device drivers and software controllers for the device components like the LCD, Ethernet, keyboard etc. These devices are usually split into logical layer component and physical layer component that implements the hardware specific functionality.

2.3. User Interface Platforms

In this section, we will briefly discuss the well-known user interface platforms from different Symbian licensees namely series 60, 80 developed by Nokia and UIQ developed by UIQ Technology AB.

2.3.1. Series 60

Series 60 (S60) is the most popular user interface of Symbian phones (50 million S60 phones were released by the end of February 2006 [17]). S60 is based on a Symbian reference design called Pearl on which Nokia made significant modifications to have it in the current form known as S60 platform [12]. Although most of the S60 phones have a screen resolution of (176 x 208 pixels), other S60 phones start to hit the market with higher resolutions i.e. Nokia N92 (240 x 320 pixels), Nokia N90 (352 x 416 pixels) and Nokia E70 (352 x 416 pixels). Nokia 7650 phone was the first release in S60. Nokia licenses S60 to other mobile manufacturers e.g. Lenovo, LG Electronics, Panasonic, Samsung, and Siemens.

2.3.2. Series 80

Series 80 (S80) provides enterprise and professional customers with optimized facilities to handle business data and multimedia content. S80 is based on a Symbian reference design called Crystal. S80 phones are known as the communicator series that are provided with a full QWERTY keyboard and a half VGA screen (640 x 200 pixels). The first S80 phone is Nokia 9210 communicator. At the time of writing, Nokia did not make any announcement on licensing the S80 platform to other mobile manufacturers.

2.3.3. UIQ

UIQ platform is based on a Symbian reference design called Quartz. UIQ phones use touch screens (pen-based interface) with a resolution of 208×320 pixels (UIQ 1.x & 2.x) and 240×320 (UIQ 3.x). The first UIQ phone was Sony Ericsson P800, which was the first Symbian phone with no keypad. UIQ licenses its platform to a number of mobile manufacturers that includes Sony Ericsson, Motorola, Benq and Arima also it is worth mentioning that Nokia 6708 was the first and the only Nokia phone, at the time of writing, which uses UIQ platform.

2.4. Developing Applications under Symbian OS

Since Symbian OS was written entirely in C++, the native programming language of Symbian OS is C++. Developing applications using C++ enables the developer to have a wide control over virtually all the capabilities of the phone through the provided APIs. In
addition, Symbian OS natively supports J2ME; hence it is possible to develop java applications (MIDlets) to run on the phone. The APIs provided by J2ME are usually limited and do not provide full accessibility to all the programming functionalities of the phone, not to mention that in terms of performance J2ME applications are usually slower than C++ application since J2ME applications run on a Virtual Machine (VM) as bytecode whereas C++ applications run directly on the processor as a binary code. However, efforts are being done, e.g. jazelle [18], to enable running J2ME bytecode directly on the hardware thus increasing J2ME applications performance. J2ME is supported by many mobile phones that are not Symbian based i.e. Nokia Series 40 are J2ME enabled. Thus, it is possible to a limited extent, to have a J2ME application that runs on both Symbian and non-Symbian phones provided that no phone specific APIs are used i.e. only the common APIs provided by J2ME are used.

Along with C++ and Java native support for developing applications that target Symbian phones, it is possible to develop applications written in other languages using third party solutions, for example, AppForge [19] provides solutions to facilitate writing applications in Visual Basic, VB .NET or C# that target Symbian phones.
Chapter 3 : Security Services Overview

Sending and receiving messages, in general, require some security features to be satisfied based one the nature of the message. For example, for personal or sensitive messages the sender may require the message content to be protected from disclosure to any third party (confidentiality). Other messages may need its authenticity to be assured. For example, a message from a network administrator to the network users saying that they should apply some patch in order to fix some security issues. In this case the network users want to ensure that the message came from the administrator not anyone else. In addition, a message from a buyer to a merchant ordering some product, in such a case the merchant wants to ensure that the buyer will not deny the order and the buyer wants to ensure that the merchant will not deny receiving the order (non-repudiation). Finally all the communicating parties want to be sure that the sent messages will reach the recipients as they were sent (integrity) i.e. were not tampered with either intentionally (third party/attacker) or unintentionally (transmission error).

In this chapter, we will discuss in detail the different security services required to secure the SMS/MMS messages, which are also applicable to any other messaging system, and how they could be implemented. In addition, we will discuss two arrangements to manage the parties involved in messages exchange that are Public Key Infrastructure and Identity-Based Cryptography.

3.1. Security Services

In this section we will discuss the critical security services that should be available in any messaging system namely confidentiality, authentication, non-repudiation and Integrity.

3.1.1. Confidentiality

Confidentiality is concerned with protecting the message content from disclosure to any third party. This could be done by encrypting the message using a key either known only to the parties involved in the message exchange (Symmetric cryptography) [7] or using the recipient's public key (Asymmetric cryptography) [7].

![Figure 3.1: Symmetric Cryptography.](image)

Figure 3.1 shows how to use symmetric cryptography to protect the message content from disclosure; that could be done as follows:
1- Alice encrypts the message (M) using the key (K) and then send (M'), the encrypted message, to Bob.

\[ M' = E_k(M) \]

2- Bob in turn will decrypt the message (M') using the key (K) to retrieve the original message (M).

\[ M = D_k(M') \]

In case Bob does not know the key (K) beforehand, usually other protocols are used e.g. Diffie-Hellman [7] in order to exchange the key between Alice and Bob.

Asymmetric cryptography could be used as well to protect the confidentiality of the message content as shown in Figure 3.2. For each communicating party a pair of keys known as public key (KU) and private key (KR) should be generated. The public key is meant to be public i.e. should be known to the public, whereas the private key must be kept from disclosure i.e. should be known only to the owner. The process of sending a confidential message using public key cryptography proceeds as follows:

1- Alice will get Bob's public key KU_{Bob} and use it to encrypt the message (M) and then send (M'), the encrypted message, to Bob.

\[ M' = E_{KU(Bob)}(M) \]

2- Bob in turn will use KR_{Bob} and decrypt the message (M') and retrieve the original message (M).

\[ M = D_{KR(Bob)}(M') \]

One of the main concerns in using asymmetric cryptography is how to make sure that the public key is authentic i.e. KU_{Bob} is Bob's genuine public key. This introduces an arrangement known as Public Key Infrastructure which will be discussed in section 3.2.1.

3.1.2. Authentication

Authentication is assuring the recipient that the received message comes form the source that it claims to be from. Authentication is achieved by digital signatures [7] implemented using asymmetric cryptography e.g. RSA algorithm [7].
Figure 3.3: Message Authentication.

Figure 3.3 shows how authentication is performed between the communicating parties Alice and Bob which is done as follows:

1- When Alice wants to send a message to Bob, she uses $KR_{Alice}$ to encrypt the message ($M$) and sends the encrypted (signed) message ($M'$) to Bob.

$$M' = E_{KR(Alice)}(M)$$

2- Bob in turn will use Alice's public key $KU_{Alice}$ in order to decrypt (verify the signature) the message ($M'$) and retrieve the original message ($M$). Bob will be sure about the authenticity of the message since only Alice owns $KR_{Alice}$ and thus the message could be signed only by her.

$$M' = E_{KU(Alice)}(M)$$

3.1.3. Non-repudiation

Non-repudiation means neither the sender nor the receiver of the message can deny being participated in the conversation, thus non-repudiation includes Non-Repudiation of Origin (NRO) and Non-Repudiation of Receipt (NRR) [7]. NRO means that the sender cannot deny that he/she sent the message whereas (NRR) means that the recipient cannot deny that he/she received the message. NRO could be achieved using digital signatures exactly as discussed before in the authentication service. However, the difference is that in NRO the public key of the recipient must be authentic [22], the authenticity of the public key is usually held by a Trusted Third Party (TTP).

In order to illustrate the difference between authentication and NRO, consider the following scenario: suppose Alice and Bob are two parties involved in some message exchange. Alice wants to send a digitally signed message to Bob. So she will encrypt the message using $KR_{Alice}$ and send it to Bob. When Bob receives the message, he will use $KU_{Alice}$ in order to authenticate it. Bob can get $KU_{Alice}$ over the phone or by any other way that is appealing to Bob. Bob then can verify the signature and be sure that Alice was the actual sender of the message. The previous scenario satisfies Bob needs i.e. ensuring that the message came from Alice. However Bob can not prove the authenticity of the message to any other third party i.e. a judge in case of a dispute. In other words, the authenticity of the message is satisfied but without NRO i.e. Alice can disown sending the message and repudiate the public key she sent to Bob. Hence, in order to prove the authenticity of the message to a third party, Alice's public key must be authenticated by a TTP. In such case Alice cannot repudiate sending the message since Bob can prove the...
authenticity of Alice's public key that he used to verify the message signature, which means that the message non-repudiation of origin feature was satisfied. A known arrangement that is used to maintain the public keys authenticity that involves a TTP along with the communicating parties is the Public Key Infrastructure and will be discussed in section 3.2.1.

In order to achieve (NRR) in a message exchange, the involvement of a TTP is usually needed. Implementing a non-repudiation protocol that provides both NRO and NRR is a complex task; since such protocol should satisfy some features [23] like fairness, timeliness etc. discussing a full functional non-repudiation protocol is out of the scope of this thesis however, for a complete discussion of such protocols refer to [23]. It is worth mentioning that some messaging services provide, one way or another, a method to confirm the message receipt without needing any action from the recipient part. The SMS/MMS provides such service by which the sender is able to know the status of a sent message i.e. delivered, pending or failed in a form of a report that could be received after sending the message. This feature is available in all of nowadays mobiles and provided by most of the MNO usually free of charge.

3.1.4. Integrity

Integrity is concerned with the correctness of the message i.e. the recipient will receive the message as sent by the sender. In other words, integrity is used to ensure that the message was not modified or tampered with after being sent. Integrity could be achieved using a fingerprint (Message Digest) that could be generated using hash functions [7] e.g. SHA-1, MD5 etc. Integrity is usually combined with digital signing of the message digest in order to protect it from modification as well as providing authenticity of the message.

![Figure 3.4: Message Integrity (With Authentication).](image)

Figure 3.4 shows the process of sending a message from Alice to Bob maintaining its integrity along with its authenticity that proceeds as follows:
1- Alice generates the message digest $h = H(M)$ of the message $(M)$ using the hash function $(H)$, after that she signs the message digest using $KR_{Alice}$ and appends the signed digest to the message then sends it to Bob.

The signed message = $M \ || \ E_{KR(Alice)}(h)$ where, $h = H(M)$

2- Bob in turn will use $KU_{Alice}$ to decrypt and retrieve the hash value $h$, then he will calculate $h' = H(M')$ where $M'$ is the received message. Finally in order to authenticate the message he will check whether $h = h'$, if so then $M = M'$ and the message is authentic otherwise, the message is not authentic.

$$h' = H(M'), \ h = D_{KU(Alice)}(E_{KR(Alice)}(h))$$

If $h' = h$ then

$M' = M$

It is noteworthy that using this method to ensure the message authenticity is usually preferable to what we discussed in section 3.1.2, since only the message digest is encrypted (signed) not the whole message thus, increasing the performance of signing/verifying the message. Of course, this could be used to ensure non-repudiation of origin if the public keys are authenticated by a TTP.

3.1.5. Combining Different Security Services

All the previously mentioned security services could be combined together in order to ensure confidentiality, authenticity, non-repudiation and integrity in messaging. We illustrated in the previous sections that both message integrity and authenticity could be guaranteed as shown in Figure 3.4. In order to provide non-repudiation just we must ensure the authenticity of the public keys used to verify the signatures. So far, we guaranteed all the security services we mentioned but confidentiality. In order to add confidentiality we can combine the same configuration discussed in Figure 3.1 along with the configuration shown in Figure 3.4 thus, Figure 3.5 shows how to send a message ensuing its confidentiality, authenticity, non-repudiation and integrity.

Figure 3.5: Sending a Message Maintaining All the Security Services.
Alice will first encrypt the message content using $K_{U\text{Bob}}$ and then send the message to Bob who can decrypt the message using $K_{R\text{Bob}}$. The rest of the procedure will proceed as previously discussed.

### 3.2. Public Key Authentication

Recalling that the authenticity of the public keys should be maintained in order to fulfill providing the security services to the message exchange. In this section two arrangements will be discussed that deal with public key authentication that are *Public Key Infrastructure* and *Identity Based Cryptography*. The former is widely used in nowadays applications whereas the latter is somehow new and it is used, at the time of writing, on a limited-scale.

#### 3.2.1. Public Key Infrastructure (PKI)

*Public Key Infrastructure (PKI)* is an arrangement used on a wide scale in most of the applications that employ asymmetric (*public key*) cryptography. The idea is to have a TTP that could be trusted by all the other message exchangers. The TTP is usually called the *Certificate Authority (CA)* which is responsible for producing *public key certificates*.

A public key certificate, in its simplest form, contains the public key of the user along with his/her ID signed by the CA's private key (see Figure 3.6). As with the other public key – private key pairs, the CA's public key is known to all the users in the community.

![Public Key Certificate](image)

*Figure 3.6: Public Key Certificate.*


Figure 3.7 shows a simple PKI with CA and two communicating parties; Alice and Bob. When Alice wants to send a signed message to Bob the procedure goes as follows:

1- Alice will authenticate herself to the CA and then send her public key to the CA in a secure way. The CA will generate her public key certificate and then Alice can publish this certificate for the other users in the community.

2- Alice signs the message using her private key and sends it to Bob along with her public key certificate.

3- Bob will use the CA's public key to authenticate Alice's public key that is stored in the certificate. Afterwards Bob can use Alice's authenticated public key to verify the message signature.

In case Alice wants to send an encrypted message to Bob the procedure goes as follows:

1- Alice will bring Bob's public key certificate from a certain directory, she will then verify the certificate using the CA's public key.

2- Alice will encrypt the message using Bob's public key and sends it to Bob.

3- Bob will use his private key in order to decrypt and read the message.

Public key certificate usually includes, besides the user ID and his/her public key, some other information related to the user, the issuer and the certificate i.e. the expiry date. The CA also maintains a Certificate Revocation List (CRL) that contains the revoked (invalid) certificates. Certificates could become invalid for some reasons like compromising the corresponding private key or the like. One known certificate standard that is widely accepted for producing public key certificates is X.509. X.509 is used in most network applications where security is considered i.e. IPsec, SET, SSL etc.
3.2.2. Identity-Based Cryptography (IBC)

Identity Based Cryptography (IBC) is another arrangement to organize applications that require public key cryptography. The idea is to use any information that identify the user (email, phone number, account number etc.) to be his/her public key thus, eliminating the overhead of PKI certificates i.e. the sender no longer needs the recipient's certificate to authenticate his/her public key.

Identity based cryptography was first proposed by Shamir [24] in 1984. Shamir was able to construct an Identity Based Signature (IBS) however; he was not able to construct an Identity Based Encryption (IBE). It was not until 2001 that the first fully functional implementation of IBE appeared. Actually there were two proposed IBE algorithms in 2001. The first one is based on bilinear pairings on elliptic curves [25], such as the Weil or Tate pairings by Dan Boneh and Matthew K. Franklin. The other one is based on the quadratic residues [26] by Clifford Cocks and encrypts one bit at a time with ciphertext expansion (the resulted ciphertext is longer than the plaintext) thus, making this algorithm inefficient for all but short messages i.e. sending the key for a session that is encrypted using symmetric cryptography. It worth to mention that some researches were done in order to enhance the overall performance of IBE like [27].

The IBE scheme developed by Dan Boneh and Matthew K. Franklin is the most efficient one and it consists of four algorithms [25]:

1- Setup: generates global system parameters and a master key.

2- Extract: uses the master-key to generate the private key corresponding to an arbitrary public key string ID, the owner of the master-key is called Private Key Generator (PKG) who generates the private keys.

3- Encrypt: encrypts the messages using the public key ID.

4- Decrypt: decrypts the messages using the corresponding private key.

Figure 3.8: Identity Based Cryptography.
Figure 3.8 shows how IBE works that could be described as follows:

1- When Alice wants to communicate with Bob she will send him a message encrypted using Bob's identity i.e. his phone number.

2- Bob will contact the PKG and get his private key based on his identity via a secure channel after authenticating himself to the PKG.

3- Bob can then use his generated private key to decrypt the message.

In addition to encrypting and decrypting messages using IBE, also it is possible to sign/verify messages using IBS similar to Figure 2.1. IBS could be done as follows:

1. When Alice wants to send a signed message to Bob she will sign the message using her private key.

2. When Bob receives the message, he can verify the signature using Alice's identity i.e. her phone number.

It worth to note that IBS was first constructed by Shamir, however after Boneh and Franklin IBE that uses bilinear mapping, cryptographers tried to constructs IBS schemes that are based on bilinear mapping [28].

IBC has some advantages over the PKI that are:

1- It is possible to send a message to some recipient even though the recipient may have not setup his/her public key at the time he/she received the message.

2- The need for public key certificates is eliminated hence; avoiding the overhead of looking up the recipients' certificates to authenticate their public keys.

3- It is possible to send messages that can be read only on a specific day if the recipient's public key is composed of his/her identity + that specific day.

However, there still some disadvantages associated with IBC [28]:

1- Key escrow: which means that the owner of the master key (the PKG) is also able to sign or decrypt any messages since the PKG is the issuer of the private keys.

2- Key revocation: in PKI each certificate has an expiry date also it is possible to generate another certificate in case of compromising the private key but, in IBC if the private key is compromised the user will need to change his identity i.e. the phone number in order to generate a new private key.

As a solution to the first problem Boneh and Franklin suggested that the master key should be distributed over a number of PKGs using Shamir's secret key sharing technique [25]. So, the master key will not exist in a single location. The user could then get a partial private key from each of the PKGs, which are associated with his identity, in order to construct his/her private key. This solution ensures that only the user with specific identity possesses the corresponding private key hence; non-repudiation is guaranteed i.e. nobody else owns that private key hence; no one can sign/decrypt the message but the private key owner.
In order to solve the second problem Boneh and Franklin suggested that a string representing a time period to be concatenated with the identity that is used as a public key. For example, if we use the email address as the user's identity (Bob@Company.com) we would attach the time period over which that key is valid for example, Bob@Company.com||MAY,2007. However, such solution is required to be standardized. In other words, what would the time period format look like, should the length of the time period be long (a year) or short (a day). By the looks of it, such standard based mainly on the application that will employ the identity based cryptography. Of course using this solution means that the user is going to have different private keys periodically each of which corresponds to some public key over some time period (identity + time period).
Chapter 4 : Securing SMS, MMS and the Mobile Content

In this chapter we will discuss how to achieve security in Symbian phones. We will handle three items that are securing SMS messages, MMS messages and the mobile content. This chapter is divided into three main sections in the first section we will explain our proposed schemes to secure the previously mentioned items, in the second section we will discuss our implementations of the proposed schemes and finally section three will include some discussions regarding the proposed schemes in general.

4.1. Achieving Security in Symbian Phones

In this section we will present our proposed schemes that could be used to maintain security in SMS messages, MMS messages and the mobile content.

4.1.1. Securing SMS Messages

Our proposed scheme to secure SMS messages is to use the IBC discussed earlier in the previous chapter. The identity that would be used as the public key is the mobile phone number and the PKG would be the MNO. Since the PKG has the ability to generate the private keys, compromising the master key will make the whole system prone to failure (a single point of failure). In addition the non-repudiation service is not satisfied since, for each message that is signed by some user there is still some probability that he/she was not the original signer i.e. another party (PKG) has the master key and hence; they could generate the private key for any public key and sign the message on behalf of the user.

Therefore, in order to relieve the single point of failure and maintain the non-repudiation service we could distribute the master key over a number of MNOs as described in [25], the underlying mathematical foundations that are used to distribute the master key and then constitute the private key is out of the scope of the thesis. So, each MNO will have only one share (part) of the master key, the user then will have to authenticate himself/herself to each MNO in order to get the partial private key that corresponds to the master key share. Having all the shares of the private key, the user can then construct his/her complete private key. As a result of the previous scenario, the master key will not exist in one single place hence; only the user will have his/her private key. However we may consider an imaginary situation where an encrypted SMS message must be opened by the force of law for one reason or another. Therefore, the different MNOs might be forced to work together in order to generate the receiver's private key and then open the message. In other words, the confidentiality of the message cannot be always guaranteed. This situation could be resolved using a combination of the regular public key cryptography i.e. RSA along with IBC, where the message confidentiality would be achieved using RSA and the authenticity of the public key would be ensured using IBS.
The following steps clarify how to ensure complete message confidentiality and maintain the other security services as well, see Figure 4.1:

- In order to differentiate between the public and private keys that are related to the IBS algorithm from the ones that are related to the RSA algorithm we will use the following conventions:
  - $K_{U(IBS)}$ → public key of the IBS algorithm.
  - $K_{R(IBS)}$ → private key of the IBS algorithm.
  - $K_{U(RSA)}$ → public key of the RSA algorithm.
  - $K_{R(RSA)}$ → private key of the RSA algorithm.

1- When Alice wants to communicate with Bob for the first time, she will send a message to Bob showing her intention to have a secure SMS message exchange.

2- Bob in turn will use $K_{R(IBS)Bob}$ to sign his $K_{U(RSA)Bob}$; of course he generated a pair of $K_{U(RSA)Bob}$ and $K_{R(RSA)Bob}$ beforehand.

   Authentic $K_{U(RSA)Bob} = E_{K_{R(IBS)Bob}}(K_{U(RSA)Bob})$

3- Alice will then receive $K_{U(RSA)Bob}$ signed by $K_{R(IBS)Bob}$ thus, she can verify the message using $K_{U(IBS)Bob}$ which is Bob's mobile number, Alice will then need to store the signed $K_{U(RSA)Bob}$ for future use (sending encrypted messages to Bob) or in case of a dispute (Bob may deny this $K_{U(RSA)Bob}$ belongs to him).

   $K_{U(RSA)Bob} = D_{K_{U(IBS)Bob}}(E_{K_{R(IBS)Bob}}(K_{U(RSA)Bob}))$

4- Only then Alice can use the authentic $K_{U(RSA)Bob}$ in order to send confidential messages to Bob. In addition, Alice can use $K_{R(RSA)Alice}$ in order to sign the message that will be sent to Bob; of course she should have generated a pair of $K_{R(RSA)Alice}$ and $K_{U(RSA)Alice}$ and sent Bob her $K_{U(RSA)Alice}$ signed by her $K_{R(IBS)Alice}$ beforehand.
The above scheme guarantees sending SMS messages that maintain the different security services which are:

- **Confidentiality**: the message is confidential since it is encrypted using the receiver's public key and could be decrypted only using the receiver's private key.

- **Authenticity**: the message is authentic since it is signed by the sender's private key (KR\(_{\text{RSA}}\)) and this signature could be verified using an authentic public key (KU\(_{\text{RSA}}\)) that was signed by the sender's private key (KR\(_{\text{IBS}}\)).

- **Non-repudiation**: neither the sender nor the receiver can deny sending/receiving the message since the sender will receive a report from the MNO indicating the SMS status i.e. pending, delivered or failed and the receiver would have the message signed by the sender thus; the sender cannot deny sending the SMS nor can the receiver deny receiving the SMS.

- **Integrity**: the receiver could ensure the message integrity by verifying the signature on the message digest.

![Figure 4.2: Secure SMS Message.](image)

In order to apply the previous scheme we propose the following SMS message format, see Figure 4.2, that includes:

**SMS Header**: one byte that contains information about the type of the SMS message (secured SMS or a signed KU\(_{\text{RSA}}\)) and whether the SMS is signed, encrypted or both. In addition, it is used to determine which algorithms were used to signed and/or encrypt the message, see Figure 4.3.

**SMS Size**: one byte that indicates the size of the SMS message that gives a maximum size of 255 bytes (2^8).

**Message Payload**: the message is included here either as a plain text or as an encrypted text according to the sender's security preferences (authenticity, confidentiality or both).

**Message Digest**: the signed digest of the message is appended to the SMS message using some hash function e.g. SHA-1, MD5 etc.

![Figure 4.3: SMS Message Header.](image)

Figure 4.3 shows the SMS header which contains:

- **KU (Public Key)**: one bit that indicates whether it is a signed KU\(_{\text{RSA}}\) or just a secured SMS.

- **A (Authentic)**: one bit that indicates the message is signed.
C (confidential): one bit that indicates the message is encrypted.

AA: one bit that is used to determine which algorithm was used to sign the message.

CA: one bit that is used to determine which algorithm was used to encrypt the message.

The SMS message header will be used to indicate what actions are needed to be taken after receiving the message e.g. decrypting the message, verifying the signature, which algorithm is used to decrypt/verify the message etc. Table 4.1 shows the first three bits of the SMS header i.e. (KU), (A) and (C) combinations and their meanings.

<table>
<thead>
<tr>
<th>Bits combinations</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>011</td>
<td>Signed KU_{RSA}</td>
</tr>
<tr>
<td>010</td>
<td>Authentic SMS message</td>
</tr>
<tr>
<td>100</td>
<td>Confidential SMS message</td>
</tr>
<tr>
<td>110</td>
<td>Authentic and confidential SMS message</td>
</tr>
</tbody>
</table>

Table 4.1: (KU), (A) and (C) Bits Combinations and Their Meanings.

The algorithm that is used to sign the message and hence, will be used to verify the signature is determined by one bit (AA) which gives the possibility to choose between two \(2^1\) different algorithms that could be used in message signing. In addition, the one bit (CA) is used to indicate which algorithm was used to encrypt the message and hence, will be used to decrypt it; the one bit gives the possibility to choose between two \(2^1\) different algorithms that could be used to encrypt the message. Table 4.2 shows each of the algorithms that will be used and their corresponding bit values.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AA</th>
<th>AC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBC(IBS/IBE)</td>
<td>0 (IBS)</td>
<td>0 (IBE)</td>
</tr>
<tr>
<td>RSA</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.2: (AA) and (AC) Bits and Their Corresponding Algorithms.

It is noteworthy that it is possible to exceed the 140 bytes limit using our scheme for example, if we sign and encrypt the message using the RSA algorithm with a key size of 1024 bit. Such long message could be sent as a concatenated SMS message which is a feature that is available in most MNOs as well as nowadays mobile phones. Concatenated SMS messages work as follows: if the SMS message exceeds the 140 bytes limit it would be divided into a number of SMS messages with a size of 140 bytes each. Each SMS message part contains some information that is used by the receiver's mobile to recombine the message again. SMS message concatenation is usually transparent to the user, however he/she might be notified by the phone that the maximum length of the message was reached so, the message will be sent as a number of messages.

In order to fulfill our scheme a secure repository has to be used to save the private keys in an encrypted format. Moreover, the mobile phone book could be extended to add the capability of storing public keys along with the other information of the contacts i.e. phone numbers, name, address, email etc. The scheme we propose to encrypt and store the private information is discussed in section 4.1.3.
4.1.2. Securing MMS Messages

Our proposed scheme to secure MMS messages is to use symmetric cryptography along with public key cryptography. The symmetric cryptography will be used to encrypt the actual content e.g. image, video or sound using a session key (KS) which is automatically generated and used only once. The public key cryptography will be used to encrypt the session key as well as sign the MMS message content. We assume that each user has an authentic public key of the other using the same procedure discussed in the previous section or using a MMS message as will be discussed shortly. The reason for using symmetric cryptography is that it is known to be much faster than public key cryptography.

![Diagram](image)

**Figure 4.4: Secure MMS Message.**

Figure 4.4 depicts how the MMS message content could be secured using our proposed scheme. Securing the MMS message could go as follows:

1. When Alice wants to send a secure object in a MMS message to Bob, she will first sign the object digest using KR$_{\text{Alice}}$.

2. Alice will generate a KS and use it to encrypt the object using a symmetric algorithm and then encrypt the KS using KU$_{\text{Bob}}$. Finally, Alice will send the MMS
message to Bob that includes the encrypted object along with the encrypted KS and the signature.

3- Bob will receive the message and decrypt the KS using his KR_{Bob}.

4- Bob will then decrypt the object using KS and finally verify the object signature using KU_{Alice}.

![Secure MMS Object Diagram]

**Figure 4.5: Secure MMS Object.**

The above scheme would be implemented individually to each object that is included in the MMS message. We proposed the following format shown in Figure 4.5 to maintain the information required to secure the MMS objects. The MMS object format includes:

- **SMMS:** The first four bytes contains a fixed string "SMMS" that indicates that the object contains some security features i.e. signed, encrypted or signed and encrypted.

- **Object Status Header:** is one byte, as shown in Figure 4.6, that is used to determine the status of the object that is signed, encrypted or signed and encrypted also, it determines which algorithms were used in signing and encrypting the message as well as whether a signed public key is included in the object.

- **Object size:** three bytes are used to indicate the length of the attached object; the three bytes give a maximum size of 16,777,215 bytes ($2^{38} - 1$) for the attached object.

- **Key Length:** indicates the length of the session key if the object is encrypted otherwise it will not be included in the message.

- **Encrypted key:** in case the object is encrypted the session key is included. If the object is not encrypt the session key will not be included in the message.
• **Object**: the object itself is included, either plain or encrypted based on the security preferences (authenticity, confidentiality or both) the sender set during creating the MMS message.

• **Object Digest**: the signed digest of the object using some hash function e.g. SHA-1, MD5 etc. if the authentication of the object is required.

![Figure 4.6: MMS Object Status Header.](image)

Figure 4.6 shows the MMS object status header. The bits in the MMS object header refer to:

**KU (Public Key)**: indicates whether it is a signed KU<sub>RSA</sub> or just a secured MMS object.

**A (Authentic)**: indicates whether the message is signed.

**C (confidential)**: indicates whether the message is encrypted.

**AA**: one bit that is used to determine which algorithm was used to sign the message.

**CAK**: one bit that is used to determine which algorithm was used to encrypt the session key.

**CAO**: three bits that are used to determine which algorithm that was used to encrypt the object, this gives the possibility to choose between eight (2<sup>3</sup>) different symmetric cryptographic algorithms.

The meaning of the first three bits in the MMS object status header is the same as the meaning of the SMS header shown in Table 4.1. The (AA), (CAK) and (CAO) bits values and their corresponding algorithms are shown in Table 4.3. We set (CAO) bits value only to the Blowfish algorithm however, other symmetric algorithms e.g. DES, AES, RC5 etc. could be used and thus, will have their own values.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AA</th>
<th>CAK</th>
<th>CAO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBC(IBS/IBE)</td>
<td>0 (IBS)</td>
<td>0 (IBE)</td>
<td>-</td>
</tr>
<tr>
<td>RSA</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Blowfish</td>
<td>-</td>
<td>-</td>
<td>000</td>
</tr>
</tbody>
</table>

**Table 4.3: (AA), (CAK) and (CAO) Bits and Their Corresponding Algorithms.**

The above scheme guarantees sending MMS messages that maintain the different security services which are:

• **Confidentiality**: the MMS message objects are confidential since they are encrypted and could be decrypted only using an encrypted session key that could be only decrypted using the receiver's private key.

• **Authenticity**: the message is authentic since the object(s) it contains is/are signed and this signature could be verified using the sender's authentic public key.
• **Non-repudiation**: neither the sender nor the receiver can deny sending/receiving the message since the sender will receive a report from the MNO that indicates the MMS status i.e. pending, delivered or failed and the receiver would have the message objects signed by the sender, thus the sender can not deny sending the MMS message nor can the receiver deny receiving the MMS message.

• **Integrity**: the receiver could ensure message integrity by verifying the signature on the object digest for each of the attached objects.

### 4.1.3. Securing the Mobile Content

Our proposed scheme to secure mobile content is to build a repository that uses symmetric cryptography in order to encrypt the data to be stored in this repository. Recalling that the symmetric cryptography outperforms the public key cryptography this is the reason why symmetric cryptography was chosen. A number of well-known symmetric algorithms is available e.g. DES, Triple DES, Blowfish etc.

All the private items (pictures, sound files, videos, documents etc.) as well as the private keys (KR_{RSA}, KR_{IBC}) of the user would be encrypted and stored in this repository. In order to store and retrieve the items from the repository, a secret key is needed that is known only to the mobile owner. So, for example if the user wants to sign a message using his/her private key, he/she should first provide the secret key in order to access the repository and then decrypt and retrieve the private key. The secret key hash value would be stored on the mobile in order to verify the correctness of the provided secret key whenever data storage or retrieval is needed. Figure 4.7 shows how the secure repository works in general.

![Figure 4.7: Secure Repository.](image)

Storing the personal data using the previous scheme maintains its privacy in case of mobile theft, being infected with a backdoor or any other reason that may cause inadvertent disclosure of the private data.

### 4.2. Implementation of the Proposed Schemes

In this section we will discuss the implementation issues of the previously discussed schemes along with the cryptographic algorithms that were used in the implementation. Three java mobile applications (MIDlets) were developed (SecureSMS, SecureMMS and SecureRep) that are used to implement the schemes of securing SMS, MMS and mobile content respectively. The focus of the implementation was to make sure that the proposed schemes along with the required cryptography algorithms could be implemented and
executed on the mobile phone on one hand, on the other hand they will not exceed the memory constrains set by the target phone.

We expected a delay in the time that is required to encrypt/sign or to decrypt/verify the data thus; we measured that time in order to give an estimate of the expected delay. The elapsed time of a certain operation could be measured in Java as follows:

```java
long time;

time=System.currentTimeMillis(); // get the current time in milliseconds

// a certain operation (generate the keys, encrypt, sign, decrypt or verify)

time= System.currentTimeMillis()-time; // get the elapsed time
```

The tools used in the development are Borland Jbuilder 2006 [31], Carbide.j 1.5 [32] and The Java SDK for S60 2nd Edition FP3 [32]. The actual Symbian device that was used to test the MIDlets was Nokia N70. We used the RSA (public key cryptography), SHA-1 (message digest) and Blowfish (symmetric cryptography) implementations provided by Bouncy castle lightweight cryptography API for Java [33]. In addition, we used the IBE and the IBS implementations provided by [29] and [34] respectively. Both the IBE and the IBS implementations were slightly optimized and modified in order to run on the target device. The optimization was done by enhancing the use of the BigInteger class in Java and the modification was mainly to separate the algorithm from the Java Cryptography Architecture (JCA) since it is not supported by J2ME at the time of writing. Although we used SHA-1 hash function in our implementation, it is possible to use any agreed upon hash function that is recommended by the security community.

### 4.2.1. Secure SMS Implementation

Figure 4.8 shows the main screen that is used to send a secure SMS message. The user would have the ability to choose between two different algorithms RSA/IBC whenever encrypting or signing the message is required. In order to simulate sending and receiving SMS messages on the same device, we generated two RSA key pairs (1024-bit) KU₁, KR₁ and KU₂, KR₂ then the two pairs were used to encrypt/sign and decrypt/verify the message adhering to the SMS message format that was described previously.

![Secure SMS Main Screen](image)

**Figure 4.8: Secure SMS Main Screen.**

Figure 4.9 shows some verbose output of creating and reading a secure SMS message along with the time elapsed in encrypting/signing and decrypting/verifying the message.
The results in Figure 4.9 show that less than a second is required in order to encrypt/sign a SMS message and also less than a second is required to decrypt/verify the SMS message. This indicates that such delay will not be noticeable by the user. However generating a RSA pair of KU, KR will need approximately 22 seconds; such delay could be accepted since the key generation is required only once and for all.

The message would be sent as a binary SMS message which is a supported type of SMS messages. Recalling that the generated private key should be stored on the mobile in an encrypted format using a tool like the secure repository that would be discussed in 4.2.3.

4.2.2. Secure MMS Implementation

Figure 4.10 shows the main screen of sending an MMS object which is an image in this case. The user could choose between one of the listed images to create a secure MMS object. In addition the facility of choosing which algorithm will be used to encrypt the object, encrypt the session key, and sign the object is provided.

Figure 4.11 shows a verbose output of creating and reading a secure MMS object by generating a 256-bit session key for the Blowfish algorithm, encrypting the key using IBE, signing the object using IBS and finally encrypting the object using Blowfish and the session key.
It is clear from the results shown in Figure 4.11 that the Blowfish algorithm requires less than a second to encrypt and decrypt the object. However the IBE algorithm takes nearly five minutes to encrypt and decrypt the session key, a period of time that would be for sure noticeable by the user. In addition, the IBS algorithm takes about nine minutes to sign and verify the object, a period of time that also would be noticeable by the user.

Recalling that both the IBE and the IBS algorithms were ported form implementations that are meant to work on desktop computers and were not meant to work on mobile phones. So we think that if the IBE and the IBS algorithms could be optimized like the algorithms in the bouncy castle cryptography API we may get better performance by which the elapsed time used in encrypting, signing, decrypting and verifying the data will not be noticeable by the user.

### 4.2.3. Secure Repository Implementation

Our implementation of the secure repository enables the user to create different folders that serve the purpose of storing different types of data e.g. images, sound files, video files etc. each of the created folder would be protected using some password that is known only to the user. In a nutshell our secure repository implementation constructs are:

**Create a folder where the private data would be stored.**

The user would provide the folder name and the password by which the data in that folder would be encrypted/decrypted. As shown in Figure 4.12 an image folder will be created using the provided password. The hash value, using SHA-1, of the password would be stored in a file in the same folder in order to verify the correctness of the provided password later in case of adding or viewing the data stored in the folder.
Adding files to a folder

In order to add new files to a certain folder, the user should provide the password that protects the folder then select the file that should be encrypted, before encrypting the file the provided password hash value would be compared with the stored hash value if they are equal the provided password would be used to encrypt the file otherwise the user would be prompted with invalid password message. Figure 4.13 shows the process of choosing the images folder and then add the Stonehenge.png image file to it.

Retrieving files from a folder

In order to retrieve a file from a folder, the user should provide the password and again the hash value of the password would be checked against the stored one, if they are equal the contents of the folder would be listed otherwise the user would be prompted with invalid password message. The provided password would be used to decrypt the file and view its content to the user. Figure 4.14 shows an example of retrieving a stored image and a text file by first decrypting them and then viewing them on the phone screen.
Recalling that RSA/IBE private keys should be stored in such repository in order to be secured and would be retrieved using a password that is known only to the owner. We found Blowfish algorithm to be a good choice to be used in encrypting the sensitive data since it provides various advantages that are not available in the other symmetric cryptography algorithms, some of those advantages include [35]:

- Blowfish is much faster than DES and IDEA.
- A variable key length could be used: 32-bits to 448 bits.
- Unpatented, free to use.
- No license required to use it.
- Has been considerably analyzed.

4.3. Discussions

In this section we will discuss some considerations regarding the proposed schemes and the implementations

4.3.1. Master Key Distribution

In our proposed scheme we suggested distributing the master key over a number of MNOs. Such suggestion was to ensure that only the private key owner would have this private key and nobody else can use this key to sign or decrypt the messages. However, even if the master key would exist in one place we do not deem this may affect the usability of the system since the same issue exists in the known PKI arrangement where the CA's private key is owned by a single entity (the CA). In other words, it is possible for the CA to produce fake certificates and impersonate any person in order to read the messages that target that person. Therefore, not only the trust in PKG should be high but in CA as well. However, in order to alleviate the consequences of compromising the master key, we suggested using the RSA to maintain complete confidentiality of the messages whenever the message confidentiality is crucial.

One advantage of distributing the master key over the MNOs is that it creates a unified system in which all the users can communicate using IBC regardless the network the user uses. In other words, a user of a certain MNO could communicate securely with the other users of the other MNOs. Of course this would be beneficial for the operators that...
provide mobile number portability, a feature with which the mobile numbers could not be distinguished based on the MNO, allowing the users to use the same mobile number with any of the available providers at will.

However, if the master key will be distributed over a number of MNOs this may incur extra overhead on the user since he/she should authorize himself/herself to the different MNOs in order to get all the key parts that constitute his/her private key. As a suggestion to reduce the overhead on the user, the key parts could be obtained electronically via a secure connection with the internet (e.g. SSL/TLS) through which the user will be asked to enter some password that was provided with the purchased SIM card packet for example. After authenticating the user using the different passwords for different MNOs, the key parts could be generated and sent to the user. In order to ensure only one user would have those keys, the automatic generation of the key parts should be generated electronically only once. So, if the user managed to get the key parts electronically, that will make him/her sure no one else obtained these keys. But if he/she failed to get the key parts this way this means that someone else managed to get this key part. However, the user should still be able to obtain this key part by authenticating himself/herself to the MNOs physically or by receiving a call on his/her mobile number.

4.3.2. Public Key Certificate Elimination

The main advantage of using the IBC is the elimination of the overhead incurred by the PKI arrangement in which each public key should be bound to the user's identity in a certificate that is signed by the CA's private key. In addition, the users will not need to lookup some directory looking for the public key certificates of the other users they want to contact. Such overhead no longer exists since the public key that would be used is simply the user's ID i.e. the phone number.

4.3.3. Public Key Revocation

One of the disadvantages of the IBC is that no public key revocation is provided. In other words, if the private key is compromised the corresponding public key (the user's identity) can not be used anymore. Recalling that in the PKI arrangement, public key revocation is provided in the form of an expiry date that is attached to each certificate issued by the CA. We can use [25] remedy to this problem in which a string representing some period of time to be concatenated with the user's identity. As a result, the user should obtain each private key that corresponds to each public key (the identity + some time period). So for example, when Alice wants to send a message to Bob she will use Bob's (mobile number || current month-current year) e.g. (+0123456789||MAY-2007) in order to encrypt the message. This means Bob should contact the PKG in order to obtain the private key that could be used in a certain month. In addition to providing public key revocation, this enables Alice to send Bob a message that could be only read in a certain time in future.

4.3.4. Replay Attacks

Replay attack is one of the common attacks in any communication system. In the replay attack an eavesdropper may listen to the communication channel and capture a message in order to replay it later. For example, a user may send a signed message ordering X
number of Y product, an eavesdropper may capture this message and replay it later even though the original sender wanted to send it just once. A worst scenario where that message contains some encrypted information (login password) that provides access to some services, again the eavesdropper may capture this message and replay it later and then get access to certain services that are available only to the authorized users.

Replay attacks could be overcome using either timestamps or challenge response authentication. Timestamps mean to attach some timing value to the sent message to indicate its freshness and that it is not a replayed message. The receiver will then check the message timestamp and will accept it if it lies within a reasonable timeframe otherwise, it will be rejected. Using timestamps requires time synchronization between the clocks of the communicating parties. In challenge response authentication, the sender will attach some random value (challenge) with the message and then the receiver will read this value and do some calculations to it and attach the new value (response) with the response to the first message. When the original sender gets the response, the new value will be checked and then compare it with the expected value, the expected value is a result of calculations that is known to both of the communicating parties. For example, Alice will send Bob an encrypted message and attach a random number (N) to it. Bob will receive the message, read the random number (N), increase it by one (N+1) and send Alice an encrypted message that includes his response || (N+1). Alice will verify the result of the calculations Bob did on the random number and see whether is conform to the value she expects.

As a countermeasure to the replay attacks in mobile messaging we suggest the challenge response authentication. We think that time-tamping will not fit well with mobile messaging due to the following reasons:

- Time-stamping needs synchronization between the communicating parties' clocks which is a thing that is difficult to accomplish in mobile phones. Mobile phones clock can get out of synch easily if the battery is removed or drained for a long time. The result will be usually resetting the mobile clock.

- Even though the mobile messages are usually time-stamped by the MNO, we still need a totally independent solution that would be acceptable by all the communicating parties, not to mention that those time stamps could be forged.

- Time-stamping does not suit the nature of mobile messaging where a message delay could be usually expected for example, in case of a network congestion or a delay in the message delivery due to the receiver is being out of network coverage or simply the mobile phone is turned off (store and forward mechanism).

- Although most mobile phones are capable of using automatic time synchronization service, not all MNOs provides this service.

The challenge response authentication could be applied in mobile messaging as the following scenario: suppose Alice wants to buy some stuff from an electronic merchant. Alice will send a signed and encrypted message that contains the payment and the order information. The merchant will then answer with a signed and encrypted message that contains a random number and ask Alice to reply with the number included in the message in order to process the order. From the previous scenario the merchant will be
sure that the message is not a replayed message if he/she got a response from Alice. Alice on the other hand, will be sure that the merchant will process the order only at her request.
Chapter 5 : Conclusion

Security is usually the main concern in most of nowadays systems. Due to the lack of security services in mobile messaging it is not possible to send and receive messages that show mutual commitment. In addition, smartphones provide many services to the users some of those services are: image capturing, video recording, sound recording, word processing etc. this makes smartphones a good choice for the users on the move. However, usually the security measures provided by smartphones are not enough to maintain the privacy of the data stored on the phone or protect it from falling in the wrong hands in case of mobile theft for example.

Most of the products we investigated focus only on the confidentiality and the integrity of the data. We did not find any of the products that provide authentication or non-repudiation to mobile messaging. Even for those products that provide confidentiality, they are based on symmetric cryptography which means that both the sender and the receiver should know the secret key. Since the secret key is known by more than one person, this increases the probability of compromising such a key. Even if some key agreement protocol like Diffie-Hellman would be involved in order to secure the process of sending the key to the receiver, such protocol per se may incur some security risks like man-in-the-middle attack [7].

We described both Public Key Infrastructure and Identity Based Cryptography as two configurations that are used in today's messaging systems. Public Key Infrastructure is widely used in mostly any system that involves public key cryptography, and binding the users to their IDs is done by authenticating them to the Certificate Authority. Identity Based Cryptography is an emerging new configuration that could be used instead of the commonly used Public Key Infrastructure configuration in order to eliminate the overhead of the certificates, and binding the users to their IDs is done by authenticating them to the Private Key Generator. Authenticating the users to the Trusted Third Party (Certificate Authority or Private Key Generator) is subject to change according to the context, regulations and/or available technologies. Some of the ways used to authenticate the users to the Trusted Third Party include, but are not limited to: a valid government issued ID, a valid driving license, a biometric ID (fingerprint, eye iris etc.), legitimate papers that shows the ownership of some properties etc.

In this thesis we proposed three schemes that could be used to provide security services to mobile messaging (SMS/MMS) as well as maintain the privacy of the mobile content. In the light of this thesis, we can provide the following answers to our research questions:

How could security services be implemented in mobile messaging?

We proposed two schemes that could be used to maintain the security services to mobile messaging (SMS/MMS). In addition, we discussed the required changes in both of the SMS and the MMS messages format in order to fulfill our proposed schemes. The two schemes were implemented in J2ME and tested on Nokia N70.

How could mobile content be secured?
In order to secure the mobile content, a secure repository, like the one we described in this work, should be used. In this repository, the personal data should be encrypted in order to be stored, and it will be decrypted whenever retrieving the data is required. Both encryption and decryption would be done using a secret key which is known only to the owner. This scheme was implemented in J2ME and tested on Nokia N70.

Which algorithm(s) should be used to implement the security services in mobile messaging, and will it/they affect the usability in terms of the time taken to send/receive secure messages?

We mainly suggest Identity Based Cryptography since it eliminates the overhead of public key certificates however; if the message confidentiality is crucial, RSA should be used. The authenticity of the public key of the RSA algorithm could be achieved by signing it using the private key of the Identity Based Signature algorithm. Both RSA and Blowfish algorithms did not take noticeable time to encrypt/decrypt the data. However, Identity Based Cryptography took a noticeable time that was actually expected since the used implementation was meant to work on desktops not mobile devices; however we think that it is possible to optimize such implementation to make it work smoothly on mobile devices.

Which algorithm(s) should be used to secure the mobile content, and will it/they affect the usability in terms of the time taken to secure the mobile content?

Any algorithm that falls in the category of symmetric cryptography fits well with the purpose of securing the mobile content. However, we suggest Blowfish since it is considered to be the fastest among the other choices. Symmetric cryptography algorithms usually run on mobile phones without noticeable delay, but of course the time taken also depends on the amount of data that are needed to be encrypted.

Providing security services to mobile messaging will extend the usability of the messages to be used in applications where showing mutual commitment is a must i.e. m-commerce. In addition, securing mobile content is an important feature for any user who is willing to store personal data on the mobile.
Chapter 6 : Future Work

As an extension to our work, it is worth to investigate the ability to implement an optimized version of the Identity Based Cryptography algorithm on Symbian phones that will not take that much time as the one we used in our implementation. Recalling that Identity Based Cryptography is based on elliptic curves so, the first thing that could be done is to try to use the elliptic curves API provided by Bouncy castle lightweight cryptography API for java which is supposed to be optimized for mobile devices. If no satisfying results would be obtained i.e. the algorithm is still taking a noticeable time, then optimizing the algorithm itself should be carried out.

Moreover, it is worth to investigate some other aspects in smartphones for which security seems to be critical. Some of the proposed aspects are sending and receiving authentic electronic business cards (vCard) and applying security services to enable secure payments using Bluetooth. Securing payments done via Bluetooth could be achieved by developing protocols/schemes that could accommodate applying security services to messages sent and received via Bluetooth in order to show mutual commitment between both the message sender and receiver. Securing the messages transferred via Bluetooth could be utilized in many daily used payment systems such as gas stations, vending machines (photo booths, beverage dispensers, ticket dispensers etc.), ATMs, parking meters etc. Actually we think that using Bluetooth in such payment systems could eliminate some of the mechanical faults that are usually caused by the currency detector. Our preliminary vision is that making payments via Bluetooth would be similar to using a credit card i.e. the Bluetooth message would contain the same credit card details that are required by such payment systems in order to complete the transaction. Again, using Bluetooth has an advantage over using credit cards as it would eliminate the mechanical faults that might be caused by the credit card reader.
References


All websites were validated on 15/01/2007