A SECURITY FRAMEWORK FOR MOBILE COMMUNICATION

by
MUSTAFA OZAN UYSAL

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
February 2011

A SECURITY FRAMEWORK FOR MOBILE COMMUNICATION

APPROVED BY

Assoc. Prof. Dr. Albert Levi .....

(Thesis Supervisor)

Assoc. Prof. Dr. Erkay Savaş .....

Asst. Prof. Dr. Hasan Sait Ölmez .....

Assoc. Prof. Dr. Özgür Gürbüz .....

Assoc. Prof. Dr. Yücel Saygın .....

DATE OF APPROVAL .....

A SECURITY FRAMEWORK FOR MOBILE COMMUNICATION

Mustafa Ozan Uysal
Computer Science and Engineering, MS Thesis, 2011
Thesis Supervisor: Assoc. Prof. Albert Levi
Keywords: Secure Key Exchange, GSM Security, Mobile Security Framework, Mobile Communication Security

Abstract

The security vulnerabilities in current GSM networks allow eavesdroppers to monitor entire communication between the mobile device and the base station over the air. In this thesis, a security framework for mobile communication is proposed. Within this framework, we develop a secure key exchange protocol using Elliptic Curve Diffie Hellman (ECDH). We further employ double hash chains for session key generation in order not to repeat resource-hungry ECDH operations too often and in order to provide forward and backward secrecy. We adopt this key exchange and generation protocol to short message service (SMS) and voice communication in mobile environment. As a proof of concept, we also implement our framework on Android platform. Moreover, we analyzed the performance of our framework using different mobile equipments. For the voice communication protocol, we also measure the data network performance for various places in the city.

MOBİL İLETİŞİM İÇİN GÜVENLİK ALTYAPISI

Mustafa Ozan Uysal
Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2011
Tez Danışmanı: Doç.Dr. Albert Levi
Anahtar Kelimeler: Güvenli Anahtar Değişimi, GSM Güvenliği, Mobil Güvenlik Altyapısı, Mobil İletişim Güvenliği

Özet

GSM ağlarındaki güvenlik açıkları, kötü niyetli kişilerin cep telefonları ile baz istasyonları arasında yapılan bütün iletişimi izlemesine sebep olmaktadır. Bu tezde mevcut güvenlik sorunlarının önüne geçmeyi hedefleyen bir güvenlik altyapısı önerilmektedir. Bu altyapının bir parçası olarak Eliptik Eğriler Diffie Hellman (ECDH) metoduyla bir güvenli anahtar değişimi protokolü geliştirilmiştir. Ayrıca çift özet zincirleri yardımıyla bir oturum boyunca kullanılan simetrik anahtarlar oluşturulur. Bunun sebebi aşırı güç tüketen ECDH operasyonlarının sıklıkla tekrarlanmaması ve oturum sırasında kullanılan anahtarlardan birinin ele geçirilmesi durumunda, önceki ve sonraki anahtarların ele geçirilen anahtar yardımıyla üretilememesidir. Söz konusu protokol ses iletişiminde ve kısa mesaj iletişiminde kullanılmak üzere geliştirilmiştir. Bir uygulama örneği olarak geliştirilen protokol Android işletim sistemi üzerinde gerçeklenmiştir. Tezde aynı zaman uygulamanın değişik donanım gücündeki mobil cihazlarla performans ölçümleri de yer almaktadır. Ses iletişimi için ek olarak mobil şebekenin veri bağlantı hızı şehrin değişik yerlerinde ölçülmüştür.

Acknowledgements

I would like to thank my thesis advisor, Albert Levi, for all his support throughout my education, answering my questions without caring about what time it is, at short, guiding me in all of my works.

I also thank Erkay Savaş, Özgür Gürbüz, Hasan Sait Ölmez and Yücel Saygın for devoting their time amongst their high volume schedule and joining my jury.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Mobile communication industry is probably the most emerging industry in the last two decades. With billions of consumers around the world and hundreds of mobile operators [18], mobile phones have become an important part of our lives. Beside voice communication, mobile phones serve as newsreaders, reminders, and alarm clocks and even as gaming devices. With so many connection possibilities and application areas, mobile phones have become a must for every person in the 21st century.

The first generation mobile communication network has started in 1981 in Nordic Countries including Denmark, Finland, Norway and Sweden [44]. The Nordic Mobile Telephone (NMT) system is known to be the first cellular phone network. This cellular system allows users to communicate wirelessly with each other like on a regular hard line, but, of course, the user has to be in the reach of a base station. A base station receives and sends signals from users in its coverage area and connects them to the mobile operator. Then the mobile operator makes the connection between the caller and the callee [31]. This basic explanation of the mobile infrastructure is still in use in the latest generation of mobile networks.

Radiolinja in Finland introduced the second-generation mobile network in 1991. The infrastructure was build and provided by Ericsson [44]. The system was designed and developed by a joint work of 13 European countries. The first name was Groupe Special Mobile (GSM) [31], which was changed later to Global System of Mobile Communication (also GSM). In 1993, the system was being used in 48 countries [44]. Today, more than %80 of the world's mobile communication is done with GSM infrastructure.

The second-generation system includes some subservices besides voice communication. The first service to be introduced was Short Message Service (SMS), which allows users to communicate with each other by sending text messages [15]. The number of characters that can be sent for this service is 160 for ASCII letters and 140 for UTF-8 character set because of the bit limitation. Another service, which has also generated the need for a 3rd generation network, was data communication. Global Packet Radio Service (GPRS) was the first data transfer method in GSM, which can

be performed simultaneously with a voice call. That is, one does not need to drop a voice call when transferring data [44]. After GPRS, the EDGE (Enhanced Data Rates for GSM) has been announced. EDGE allows users to reach faster data transfer rates (up to 400 Kbits/s) compared to GPRS (40 Kbits/s). These technologies often referred as 2.5G or 2.75G depending on the bandwidth they provide.

To provide faster connection speed, third generation network infrastructure has been announced in 2001. Japan was the first country to use the new UMTS (Universal Mobile Telecommunications System). Other countries in Europe, where the second generation GSM dominated the market also adopted their network to UMTS [31]. To be able to use this new network technology, a consumer needs to buy 3G compatible handsets. Those new cell phones also provide backward compatibility for second generation GSM system. In this case, 2G network is used mostly for voice communication and also for data communication, where 3G network is not available. 3G networks are able to provide up to 56 MBit/s download speed [31].

Like every communication method in the history of mankind, eavesdroppers also threaten mobile phone networks. For the second-generation networks, a stream-cipher, called A5, was announced [32], where the communication between the base station and mobile handset is encrypted with a key size between 16 to 64 bits depending on the operator. Before the encryption phase, a session key exchange is done using A3 algorithm [32]. The information for the key exchange and user identification is stored in the SIM (Subscriber Identity Module) cards provided by the operator upon subscription. Although the second-generation infrastructure is known to be secure for many years, researchers have found vulnerabilities in the security mechanism, both in algorithm and authentication protocol [32]. After the first published attack on A5 in year 2000 [32], the GSM 2G network cannot be considered as secure anymore.

Subservices, for example short message service, also suffer from this security vulnerability. Furthermore text messages are stored unencrypted in the short message service center (SMSC) [1] until they are sent, which makes them vulnerable to eavesdroppers working inside the mobile operator. It is known that many companies reach their customers via short message service, where they sent private or personal

information. Especially in government applications, the text messages should be encrypted to provide extra security due to network vulnerabilities [35, 36].

In this thesis, we designed a key exchange and secure communication protocol for available communication methods, for both text and voice. Due to the limited computational capabilities of mobile devices and network limitations, algorithms in the protocol were chosen to be computed fast and to provide higher security at lower key sizes.

For text messaging, the network limitation for the size of one message is 1120 bits [1]. Therefore, a key exchange model based on RSA fails to provide enough security in one message due to larger key sizes. Another consideration for the system is to initiate the protocol with minimum number of messages, since every message is charged at the operator. Therefore sending multiple messages for larger key sizes was out of the scope. The final decision for key exchange is to use ECDH (Elliptic Curve Diffie-Hellman), since it provides higher security at smaller key sizes [19]. Furthermore the performance for ECDH makes it suitable for mobile devices.

For voice communication, we modified our protocol used in text message encryption. We developed a VoIP-like communication model, where two mobile clients are connected with each other over the data network (3G where available). This is convenient for the user, since the system does not depend on a third party server to store matching IP addresses with phone numbers. To establish the connection, the IP addresses of both parties are exchanged during the key exchange. The key exchange is achieved by sending text messages to each other.

In this thesis, we have implemented two key exchange protocols, one for voice communication and another one for text messaging. One application is written for J2ME supported mobile phones, where there is less computational power and limited memory. This implementation only covers the secure text message protocol and aims the highest percentage of mobile devices on the market [44]. The second application is written for Android platform, an open source operating system supported and developed by Google and Open Handset Alliance [45]. Android supports Java language in the application level [45] and offers a wide range of libraries to reach the hardware functionality of the phone. Android mainly focuses on mobile smart phones

where the computational power is considerably high as compared to non-smart phones [45]. Therefore, we have chosen to implement the voice encryption in the Android platform.

The performance tests were done in many aspects. The first test aims to show the hardware performance for encryption and key exchange operations. Here, we try to find the optimum security and performance metrics for different key sizes and CPU power. The second test is done for voice communication, where the performance of the network is measured. The delay between sender and receiver is recorded according to the end to end delay of the data packets in different networks. The network type can be either 3G data network or 2G with EDGE capabilities. Different bandwidth sizes result in different delays. Also the regions, where the tests are performed, are recorded to provide extra information on network status. The region is important due to the number of base stations and number of people living in this area, since they both have an impact on network quality.

In the next section, we provide information about GSM networks and cryptography. The third section explains our protocol and also includes the security analysis. The fourth section includes the performance test results and the fifth section summarizes the thesis.

2. BACKGROUND INFORMATION

In this section, we give background information for the terms, systems and algorithms used in this thesis. In Section 2.1, information about the mobile communication networks will be given. In Section 2.2, symmetric encryption algorithms, key exchange methods and hash functions are explained. In Section 2.3, literature on mobile communication security will be summarized.

2.1. Background on Global System for Mobile Communication

Global System for Mobile Communication (GSM) is most widely used mobile communication system around the world. In this section, we give background information on voice and short message infrastructure of the GSM network.

2.1.1. Voice Infrastructure

The cellular infrastructure was developed to increase the capacity of the mobile telephone service [31]. Before the introduction of the cellular infrastructure, the capacity of a wireless telephone system was limited to 25 channels within 80 km radius. The idea behind the cellular network is to use many transmitters with shorter radius and lower power.

The highest power transmitted by a cellular station is about 100 W. Since the radius is small as compared to the older systems, a region can be divided into smaller areas with his own transmitter. Each area in a region has a frequency range and transmitter called base station [31]. Neighboring cells cannot have the same frequency because of the interference, but a frequency can be reused, if the base stations are far away from each other.

The most ideal form to split a region is to build hexagonal shaped areas within the region. With a radius of R, each station is $\sqrt{3}\ R$ away from each other. Having the same distance between all base stations allows users to switch to the other station much easier [31]. In practice however, this hexagonal shape cannot be achieved perfectly because of the geographical shape of the region.

The frequency reuse in the cellular infrastructure allows mobile operators to serve more clients with less frequency bands. This is achieved by repeating a

frequency band with cell stations far away from each other. The main challenge in a base station design is to use the same frequency as much as possible. Since it is not possible to have two neighbor cells with the same frequency band, a design should be made carefully.

With the increasing number of customers, the system needs to be expanded by adding more frequency bands into the network. To increase the capacity, the following methods are available;

1. Adding new channels: If there are still unused frequencies available in the region, the base stations can be updated to use these available frequencies
2. Frequency borrowing: Frequencies of neighboring cells can be set dynamically.
3. Cell splitting: Since the perfect hexagonal pattern is never achieved due to regulations or geographic conditions, it is mostly possible to add new base stations into the network. Splitting the region into smaller areas and redesigning the network and frequency usages allows the network to serve more customers. The downfall is that the more base stations you have in a region, the more handoffs will the user have.
4. Cell sectoring: Cell sectoring means dividing the cell coverage into smaller sectors with different channels. Cell sectoring is only possible with the help of directional antennas. In this case, one cell can be divided into 3 or 6 areas with different frequency bands.

Global System for Mobile Communications (GSM) was a standard developed by European countries to have the same network rules in every country so that users can use their cell phones all around the continent. After the success of GSM system, it became a global standard, which includes countries in America, Asia, Africa and Middle East. In 2010, there were over 4,42 billion GSM subscriptions worldwide [9].

The main components in a GSM system as defined in the standard are as follows [31];

1. Mobile Station: Mobile station is an electronic device, which communicates with the base station. In order to be a mobile station, a device should have a SIM card, radio transceiver and digital signal processor. The Subscriber

Identity Module (SIM) contains all the necessary information for authentication and secure communication with the mobile operator. The mobile operator uses this information to identify his customer. The SIM is vital for a user to make phone calls or data connection over the network.

2. Base Station Subsystem: Also known as BSS, a base station subsystem includes a base station controller (BSC) with a set of transceiver stations attached to it. The Base Transceiver Station (BTS) is a single cell with a radio antenna, radio transceiver and a connection to his Base Station Controller. BSC is responsible for handoffs between its cells and allocating radio frequencies.

3. Network Subsystem: Network Subsystem (NS) controls the communication between the GSM network and public telecommunication network. The main functionalities of MS are authenticating and validating customers, controlling handoffs and enabling roaming for visiting customers. The NS is controlled by mobile switching center (MSC). The information is stored in four databases;

a) Home Location Register (HLR): HLR stores the information of every customer and the base stations that the customers are connected to.

b) Visitor Location Register (VLR): The location information of every customer is stored in the Visitor Location Register. This location information is the current physical location of each customer in the network. This information is used to determine to find the switching center when a call comes to the customer.

c) Authentication Center (AuC): In the Authentication Center Database all the private information about the customers is stored. This information consists of private keys for encryption and authentication.

d) Equipment Identity Register (EIR): In this database, all the information about devices in the network is stored.

The overall GSM infrastructure is given in Figure 2.1.

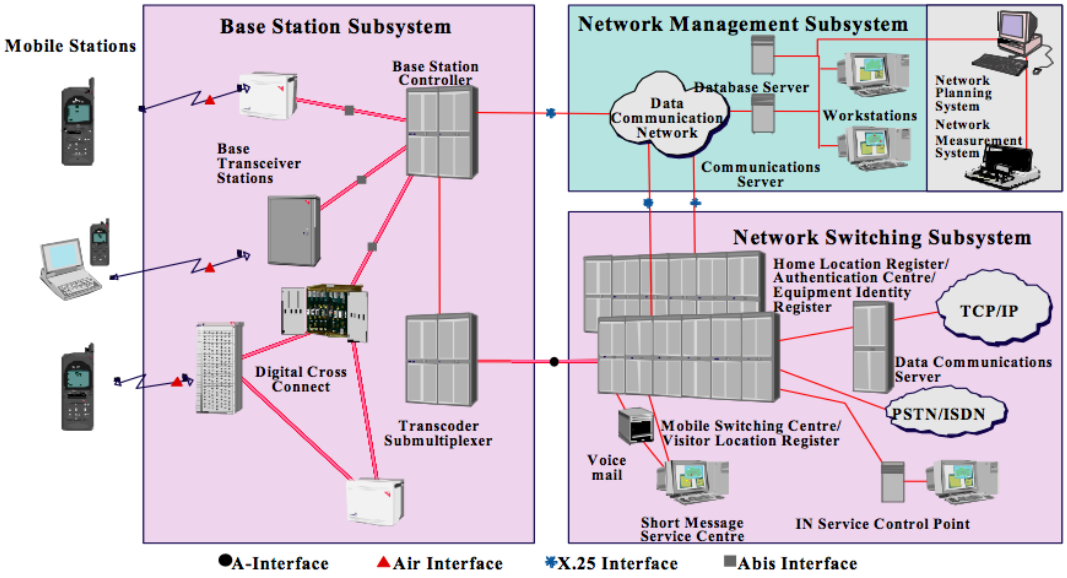

Figure 2.1 – GSM Architecture [29]

In GSM networks authentication of subscribers to the mobile operator is important for network security. Before a user identifies him to the network, an authentication followed by a key exchange occurs. After that, the communication between the mobile station and base station is performed in encrypted manner.

Since the computational power in 1980s was not feasible for public key cryptography, a different method for authentication was chosen in GSM standard. A3 algorithm is used for user authentication. The A5 is used for encryption of data over the air and A8 algorithm is used as the key generation algorithm. The algorithms were kept safe between the contributors of the GSM Memorandum of Understanding, but many attacks on those algorithms were published since the announcement of GSM [32].

A3 and A8 should be same only between the subscriber and mobile operator; therefore, they can be different in every operator. The standard only defines the input and output of those algorithms [32]. COMP128 algorithm, which was a popular choice for A3 and A8, allowed attackers to clone SIM cards and to make duplicate subscribers within the operator [32]. This is done by subtracting the key from the SIM card and copying it to another. Recent algorithms are much more secure than COMP128, which make SIM cards to be more resistant.

A5 is a standard encryption algorithm between the communication of mobile devices and base stations. A5/2 is a weak version of A5 with 16 bits of key length. The non-export version A5 however has a key length of 56 bits. Both algorithms have been broken and attacks were published [32].

Every time a subscriber wants to join the network, the authentication process occurs. The authentication process is shown in Figure 2.2.

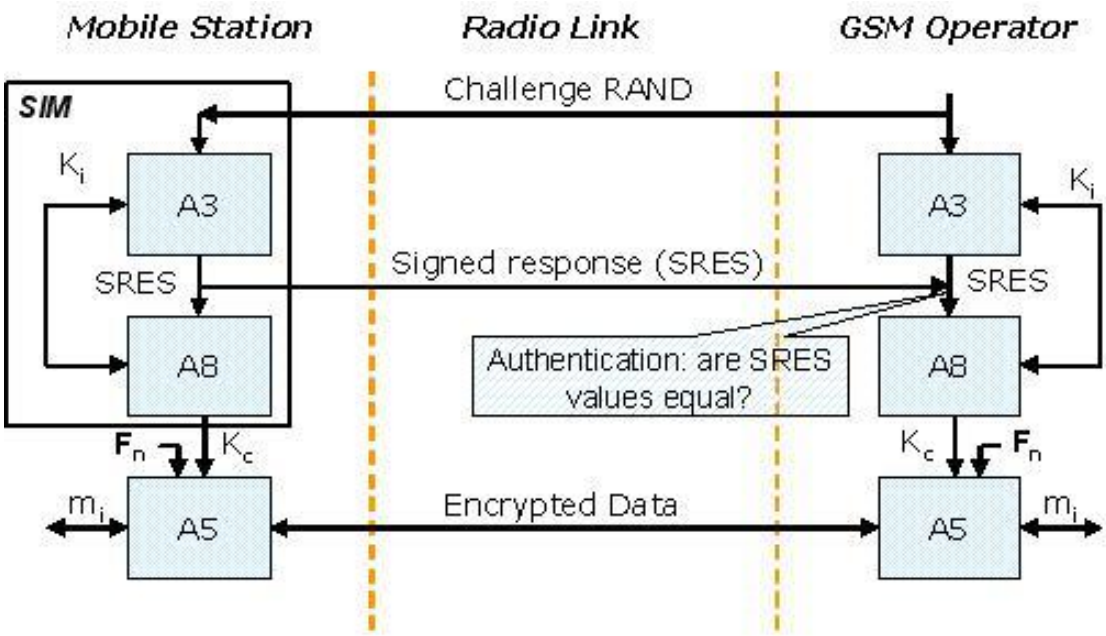

Figure 2.2 – GSM Authentication [34]

The mobile stations send a subscriber identity, either TMSI or IMSI. The VLR send TMSI to IMSI and than the IMSI is sent to HLR/AuC. The AuC creates a random 128-bit challenge called $RAND$ and calculates $RES = A3(Ki, RAND)$ and also the encryption key $Kc = A8(Ki, RAND)$. The values $< RAND, RES, Kc >$ are sent back to VLR, where $RAND$ and Kc are hold and $RAND$ is sent to mobile station [32].

The mobile station calculates $SRES = A3(Ki, RAND)$ with the information stored in the SIM and sends it to VLR. If $SRES$ and RES match, the client is authenticated. In a visited network AuC sends a set of $< RAND, RES, Kc >$ values to the VLR to make the authentication process faster.

Although all traffic over the air is encrypted, the infrastructure can choose to

disable the encryption. It is also possible that a handshake is made between mobile station and network on which algorithms they support [32]. After that, they both decide on which encryption algorithm they will use.

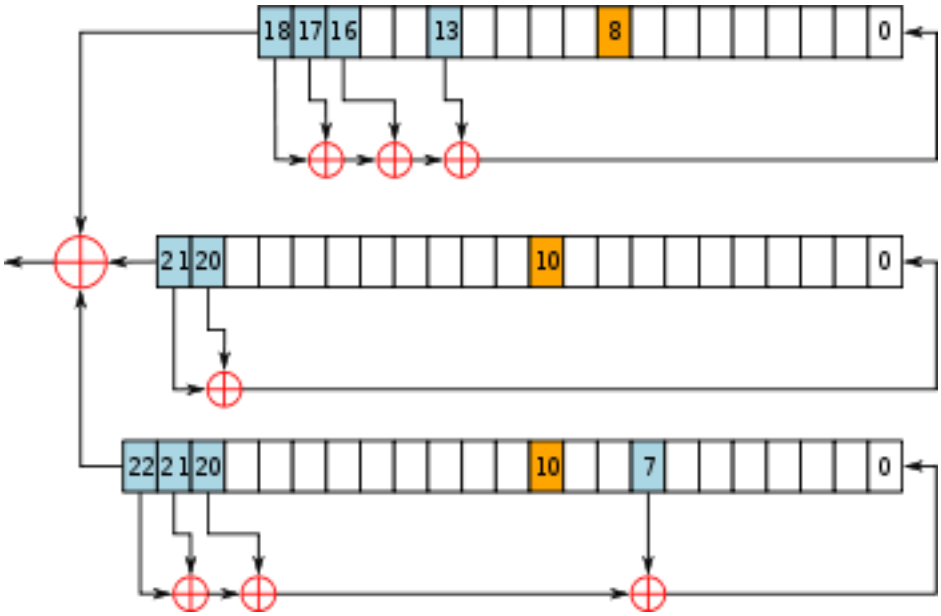

Figure 2.3 – A5 Algorithm [44]

A5 is a stream cipher with 114-bit frames given in Figure 2.3. Each frame has a key generated from Kc and current 22-bit frame number. Since the transmission can contain errors due to various reasons between network and mobile station, a stream cipher is preferred. A block cipher would cause an avalanche effect in the output, if a bit is damaged during the transmission [32]. Despite the block cipher, in a stream cipher only one bit is affected in the same situation.

2.1.2. Short Message Service

A short message can be either 160 characters long with ASCII encoding [14, 15] or 140 characters with UTF-8. Eastern countries like China or Japan, where more characters are required, have Unicode encoding with 70 characters [16, 17]. Assuming that a character can be 7 bits long in GSM 38.03, a message body can carry $160 * 7 = 1120$ bits of data. For longer messages, the devices divide the message and a sequence number is given to each part by the phone.

Short messages can also be used to activate operator specific SIM commands.

Such messages can be used to disable a SIM or a mobile device, upload network settings, etc. Their corresponding bits in the message header identify these messages and mostly the user does not get a notification about them.

Another type of short message is the cell broadcast service, which is sent only to those mobile devices that listen to a specific broadcast channel [16]. This type of message is mostly used to give information to people in a specific area about traffic or weather. Also news headlines or commercial ads can be broadcasted according to subscription [18].

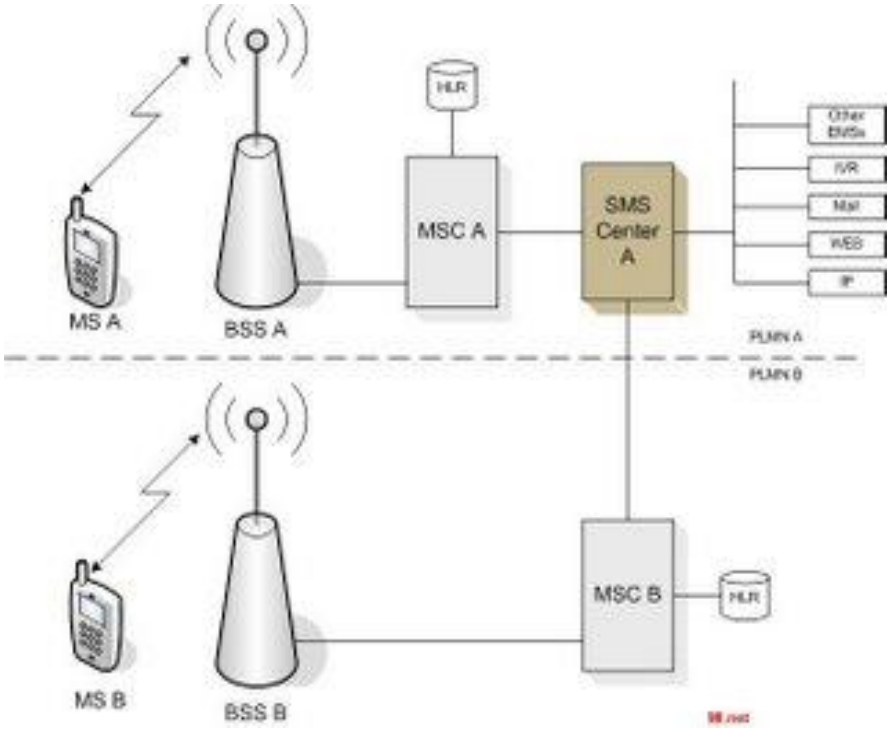

Figure 2.4 – SMS Life Cycle [27]

Unlike phone calls, the short messages should arrive at the destination even when the receiver's phone is off. In order to achieve that the messages should be stored at some point during transport for the case that receiver is either out of range or have his phone turned off. The server to achieve this store and forward system is called SMSC (Short Message Server Center) [13, 16]. The overall life cycle of a short message is given in Figure 2.4.

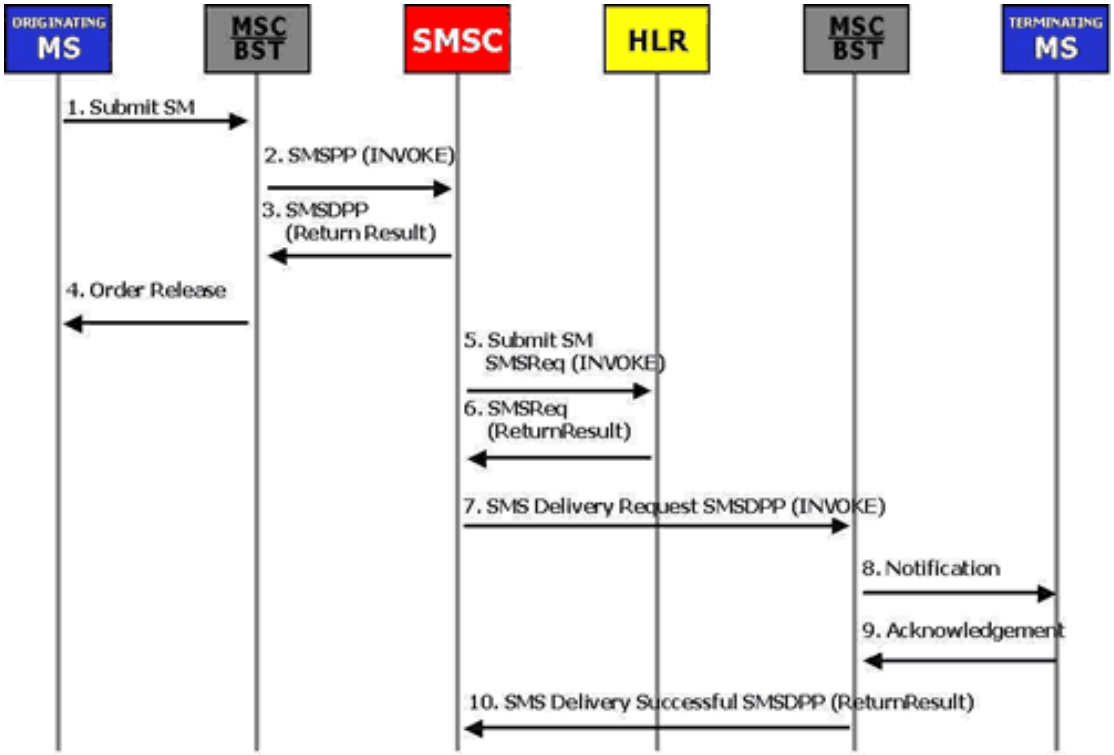

Figure 2.5 – Mobile Originated SMS [25]

When user sends a message, first it goes to the BS over the air. After that the message is delivered to the SMSC using the SS7 (Signaling System 7) network [11]. A short message, which is created by the mobile phone and sent to the SMSC, is called MO (Mobile Originated). A Mobile Originated message lifecycle is shown in Figure 2.5. When the message arrives at SMSC, the destination is questioned at the HLR (Home Location Register). If it is active, the message is forwarded to his receiver, again using SS7 network and air interface. Such a message is called MT (mobile terminated), meaning it is sent from SMSC to the mobile device [16]. Mobile Terminated message lifecycle is shown in Figure 2.6.

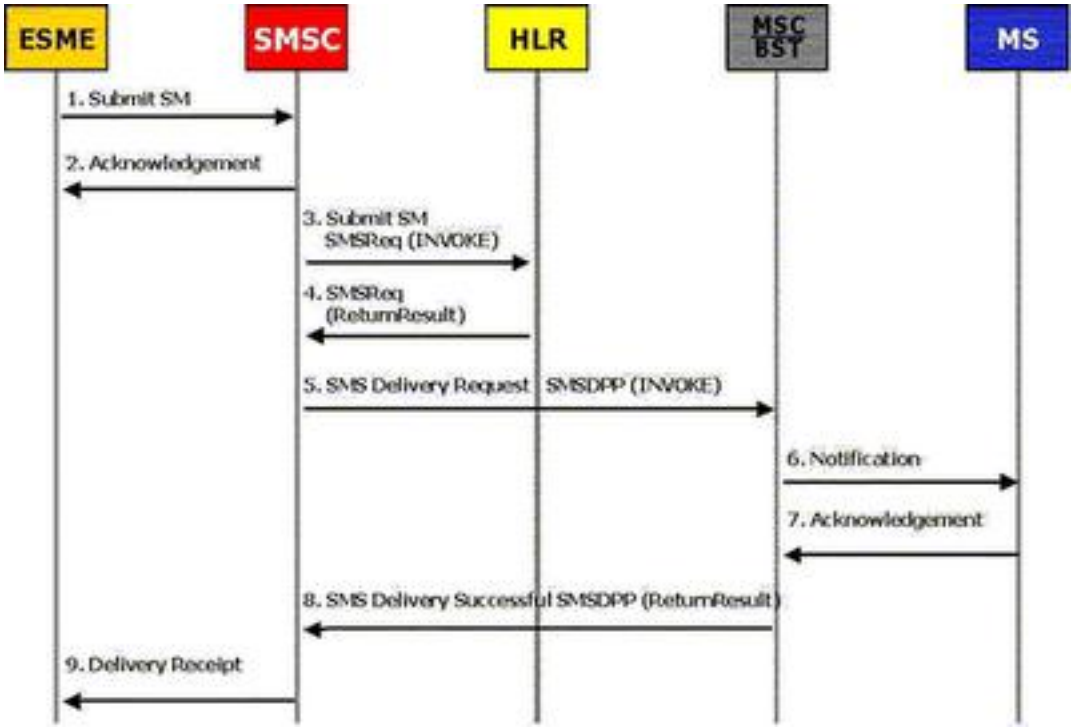

Figure 2.6 – Mobile Terminated SMS [25]

Since the SMSC is responsible to handle huge amounts of messages inside the operator, it is possible that some messages might get lost. When a message is not sent within the expiration time, it is discarded by the SMSC. This is mostly the case, when the receiver side stays inactive for a long time.

2.1.3. Vulnerabilities of Short Message Service

Like the voice communication in GSM network, SMS protocol also suffers from some security threats. In this section, we will mention some of them.

2.1.3.1. Over the Air

The transfer between the mobile device and the base station is established through the air. According to GSM standard [1], this communication can be either unencrypted, if the law does not permit it, or encrypted using A5 algorithm. A5 comes with two additional options; A5/1 with strong encryption using 64-bit key and A5/2 with a 16-bit key [1]. Such key lengths are no more considered as secure according to NIST [21]. Furthermore, recent researches show weaknesses in the A5 algorithm. An eavesdropper with enough equipment can crack all encrypted information within

hours [20]. Also, if the communication is unencrypted, an ordinary attacker with an antenna capable of receiving GSM signals can listen to the communication.

The false BSS attack targets the one-way authentication weakness of the GSM network [22]. By using his own BSS equipment, an attacker can make the mobile station believe that it is communicating over operator's channel. Since choice of encryption is under BSS's control, attacker can manipulate the mobile station not to use encryption. In this way the attacker can easily watch all the communication.

2.1.3.2. Inside the Operator

Although the SMSC is protected through firewalls and other countermeasures, it is still possible for someone to gain access to the contents. Since the messages are kept unencrypted in the database, an attacker can read or manipulate every message. Also someone inside the operator with enough privileges can get the information in a message.

The communication between the SMSC and the base station is performed using SS7. This layer of communication is completely under the control of the operator and it is possible that someone with knowledge about SS7 can eavesdrop or even change the contents of every message going through network. Security in SS7 is not mandatory and the operators mostly keep the security measure in this layer secret.

Another fact about SMSC is that we cannot know how long our messages are kept in the database of the operator. If the operator decides to store every message in its server, our confidential information will stay at that database forever, which may be later accessed by other people. The information can be used for commercial issues or for gaining personal information about a specific person.

2.1.3.3. Modification

The header of a message defines whether it is a normal text message or an operator setting. It also contains information of the sender. Since the header is not protected and does not contain a checksum, the receiver cannot understand whether a modification is made to it. Therefore an attacker can impersonate someone by changing the sender information field in the header.

The body part of the message is also subject to the modification attacks like the header. In this way, an attacker can change vital information in a message and cause problems for the sender or receiver.

2.2. Background on Cryptography

In this section, background information on encryption algorithms and methods are provided.

2.2.1. Diffie Hellman Key Exchange

Diffie Hellman Key Exchange [24] is the first key exchange algorithm which uses public-key cryptography. It is widely used by commercial and non-commercial applications.

The purpose of the algorithm is to provide a secure method for two users (Alice and Bob) to share a secret with each other. It is crucial for applications, which use symmetric encryption algorithms to provide secure communication channels.

The algorithm runs as follows;

1. One of the parties (Alice or Bob) select a large prime number p, a generator g, and an integer a, where $a < p$. The generator g is a primitive root of p. Those values can be shared over an insecure channel.

2. Alice calculates $A = g^a \bmod p$ and sends A, g and p to Bob.

3. Alice calculates $B = g^b \bmod p$ and sends B to Alice.

4. After these values are exchanged, the shared secret K can be generated. The calculation is as follows;

$$K = A^b \bmod p = B^a \bmod p \quad (2.1)$$

The key exchange steps and calculations are given in Figure 2.7.

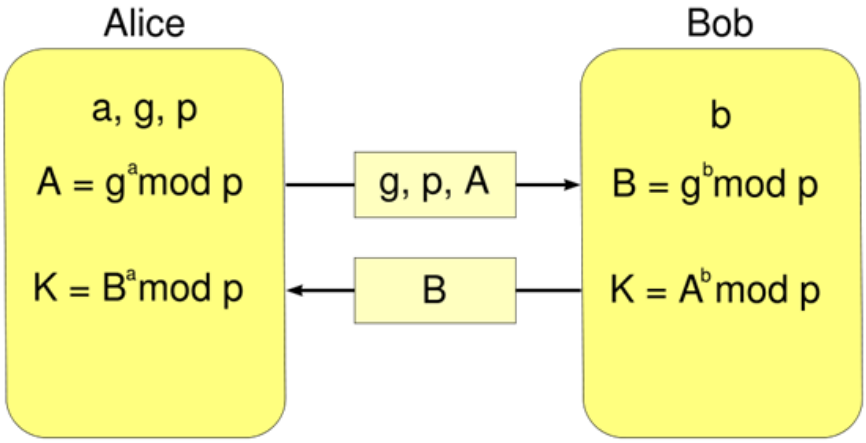

Figure 2.7 – Diffie Hellman Key Exchange [37]

An attacker, who wants to obtain K, should know a. After that, he or she can calculate K. The main problem for an attacker is to calculate the discrete log. This calculation is known to be computationally hard.

2.2.2. Elliptic Curve Diffie Hellman

Elliptic Curve Diffie Hellman is a key exchange algorithm [24, 30, 42]. Suppose Alice and Bob wants to exchange a key. They agree on a point on the elliptic curve $E: y^2 = x^3 + ax + b\,(mod\ p)$ and a base point G on E. The algorithm runs as follows;

1. Alice selects an integer $n_a < n$, where n is the order of G. n_a is the private key of Alice. The public key is generated as;
   $P_A = n_a \times G$ (2.2)
2. Bob also generates a private key n_b and a public key P_B as;
   $P_B = n_b \times G$ (2.3)
3. Alice generates the shared secret $K = n_a \times P_B$. Bob also generates the shared secret as $K = n_b \times P_A$. The shared secret K is the same at both sides because;
   $K = n_b \times n_a \times G = n_a \times n_b \times G$ (2.4)

If an attacker wants to obtain the shared secret, he should be able to solve the Elliptic Curve Discrete Logarithm problem, which is known to be computationally hard [42].

2.2.3. Advanced Encryption Standard

Data Encryption Standard (DES), the successor of AES, is a symmetric encryption algorithm. The algorithm works as a block cipher with the size of 64 bits and uses 56 bit keys to encrypt the plain text [43]. DES was first published in 1977 by IBM and widely used in government and commercial applications.

After the end of life of DES, National Institute of Standards and Technology (NIST) started an election for the next standard in 1997 [43]. Rijndael cipher was selected from a group of five algorithms and became the successor of DES in 2001 with the name Advanced Encryption Standard (AES).

Like DES, AES is also a block cipher with a block size of 128 bits [43]. AES is actually a standardized version of Rijndael algorithm. AES supports 128 bits block size, whereas Rijndael supports various block and key sizes [43].

Although AES has a block size of 128 bits, its keys can be of size 128, 192 or 256 bits. The algorithm works as repetitions of rounds, where each round consists of four steps. For the decryption, those rounds work in reverse to get the plain text back. The algorithm will not be described in detail, since it is beyond the scope of this thesis.

128 bits key size provides enough security until 2030 according to NIST [21]. Furthermore, there are no attacks reported on the algorithm. Therefore, AES is the choice of symmetric encryption algorithm in the protocol implementation.

2.2.4. Cryptographic Hash Functions

A hash function is a function, which gives a fixed size output for an arbitrary length input. The output is called the message digest [42]. A cryptographic hash function should satisfy the following rules;

1. The function should work fast. For a message m, the output $h(m)$ should be produced rapidly.
2. The function h should be one-way. This means that it is computationally hard to find the message m from the message digest $h(m)$.

3. It should be computationally hard to find two arbitrary messages with the same message digest $h(m_1) = h(m_2)$. The property is called strong collision resistance. This property claims that finding collisions should be computationally infeasible [42].

4. Another property for hash functions is the weak collision-resistance. That is for any m, it is hard to find $m' \neq m$ such that $h(m') = h(m)$. It should be infeasible to find a message m' that produces the same digest of a known message m.

There are many cryptographic hash functions in service today. Some of them are the Message Digest (MD) family and the Secure Hash Algorithm (SHA) series of algorithms [42].

The algorithm of the first MD was never published, but the algorithms of MD2, MD4 and MD5 are known by public. After weaknesses were founded in the first MD algorithms, Ron Rivest published MD5, which was an upgraded version of MD4. Due to recent collisions found in MD5, its security is no more certain [42].

The secure hash algorithm was produced and used by National Security Agency (NSA) and given to National Institute of Standards and Technology (NIST) [42]. In 1993, the first version of SHA was published (FIPS 180). The SHA-1 is an improvement version of SHA, which is recommended by NIST.

The length of the message digest produced by SHA-1 is 160 bits for any message. The input message m is sliced into smaller messages $m_0, m_1, m_2, \ldots\ldots, m_n$ with the same length. A compression function works repeatedly taking these blocks and the output of the previous block as an input. Let's say X_0 is the first value. Than $X_j = h'(X_j = 1, X_j)$ will be the formula for the next output. The last output X_l is the message digest for m.

The most important part of a cryptographic hash function is the underlying compression function [42]. The input bits of this compression function should change as many output bits as possible. An important change in SHA-1 is that it uses more input bits to produce output bits during operation compared to MD algorithm. That makes the SHA-1 more reliable but also slower.

As of 2010, SHA-1 is no more considered to be secure due to recent attacks on the algorithm. In 2005, Rijmen and Oswald published an attack, where they make less then 2^{80} operations to find a collision [48]. Also in the same year, Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu published another attack with less than 2^{69} operations to find a collision [47]. Later, they improved their work, where they require only 2^{63} operations [49].

The new SHA-2 family hash functions are named after their output length, SHA-256 with 256 bits output, SHA-384 with 384 bits output and SHA-512 with 512 bits output. The new versions have the same structure and binary operation like SHA-1 [24].

Table 2.1 – SHA family hash functions

	SHA 1	**SHA 256**	**SHA 384**	**SHA 512**
Digest Size (in bits)	160	256	384	512
Message Size (in bits)	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Resistance to birthday attacks (trials)	2^{80}	2^{128}	2^{192}	2^{256}

The hash functions are vulnerable to birthday attacks [41]. Birthday attack implies that it is possible to find a collision in $2^{n/2}$ evaluations, where n is the length of output in bits. To provide equivalent security with the AES-128, SHA-256 was chosen as the hash function in the protocol implementation. Comparisons of different hash functions can be found in Table 2.1.

2.2.5. Using Hash Functions for Integrity Check

To ensure the integrity of the message, a hash function can be used as follows. Suppose we have two parties, A and B, which share a secret password pw only known to them. When A sends B a message, he sends $h\ (M\ ||\ pw)$ with the message as in Figure 2.8;

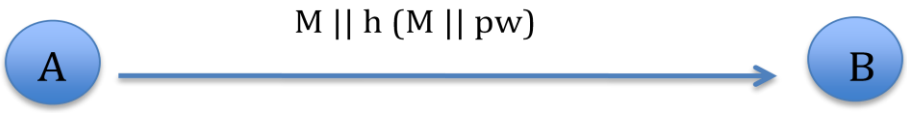

Figure 2.8 – Message integrity check

In order to check the integrity of the message, B will concatenate the message with the pre-shared password pw. If the hash of the message matches the received hash, B can be sure that the message is not tampered. An attacker C, who wants to modify the message, cannot extract the pre-shared password out of the hash due to the one-way property of the hash functions.

2.2.6. Hash Chains

A hash chain is a series of hash functions, where the input of one hash function is the output of the previous hash function. The number of hash functions in the hash chain gives the length of the hash chain. A hash chain of length n can be shown as follows;

$$x, h(x), h^2(\mathrm{x}), h^3(\mathrm{x}), \ldots\ldots, h^n(\mathrm{x}) \tag{2.5}$$

, where $h^n(\mathrm{x}) = \underbrace{\mathrm{h}\ (\mathrm{h}\ \left(\mathrm{h}\ (\ \ldots \mathrm{h}(\mathrm{x})\right))}_{n \text{ times}}$.

For example, a hash chain of length 4 is;

$$h^4\ (x)\ =\ h\ (\ h\ (\ h\ (\ h\ (\ x\ )\ )\ )\ ) \tag{2.6}$$

The hash chains are very easy to store, since you only need the first input value x to produce any element in the hash chain. In our text messaging protocol, this

property is used to generate different keys for every text message. After the key exchange, both parties know the input value for the hash chain, the seed S, where they generate the n^{th} element of the hash chain as the key for the n^{th} message [50].

Due to the one-way property, a hash chain provides backward secrecy, when the chain is employed for key generation. Backward secrecy means that compromising any key during an encrypted communication should not compromise earlier keys. Forward secrecy is that the attacker cannot produce any future keys from a compromised key.

To improve a hash chain to provide both forward and backward secrecy, double hash chain is used to generate symmetric keys in our protocol. A double hash chain is two series of hash chains, which are generated with two different seeds. In our protocol, we created two chains with the different seeds S_1 and S_2 derived from the shared secret after the key exchange. The symmetric key is generated from XORed output values of two hash chains. The generation method for symmetric keys will be given in the Section 3.

2.2.7. Applications of Hash Chains

The Lamport authentication scheme [51] is an authentication scheme using hash chains. In this scheme the server stores the n^{th} hash chain value of the password pw. The user calculates and sends $n-1^{th}$ hash value to server. The server calculates $h(h^{n-1}\ (pw))$. If the stored hash value matches the calculated hash value, the user is authenticated. This time the server stores $n-1^{th}$ hash for future authentication.

Suppose an attacker knows $h^{n-1}\ (pw)$. Since the server waits for $h^{n-2}\ (pw)$ for the next authentication, the attacker cannot produce $h^{n-2}\ (pw)$ from $h^{n-1}\ (pw)$ due to the one-way property of hash functions. The user can authenticate with the server n times.

2.2.8. Digital Signatures

Digital signatures are used to provide undeniable authentication proof about the owner of a message. In order to prove that a message belongs to a particular person, a third party needs to verify the digital signature attached to the message.

A digital signature scheme consists of three parts; a key generation algorithm, a signature algorithm and a verification algorithm. The digital signature for a document can only be created by using a secret known by the owner of the document, for example a private key. In order to verify the signature, a public key should be distributed. The verification algorithm takes the document and the public key as the input and checks whether the document is signed by the owner or not.

In most of the digital signature schemes, public key cryptography is used. It should be infeasible to calculate the private key from the public key to provide security for the digital signature.

2.2.9. Digital Signature Algorithm

The Digital Signature Algorithm (DSA) was proposed as the Digital Signature Standard (DSS) in 1991 by NIST [53]. The public and private key generation phase is as follows;

1. User selects a prime q such that $2^{159} < q < 2^{160}$
2. User chooses an integer t with $0 \leq t \leq 8$ and a prime p with $2^{511+64t} < q < 2^{512+64}$, so that $p-1$ can be divided by q
3. User selects α with $1 < \alpha < p-1$ and computes $g = \alpha^{(p-1)/q} \, mod \, p$. If $g = 1$, a new α must be chosen.
4. User selects a with $1 \leq a \leq q-1$
5. Computes $y = g^a \, mod \, p$
6. a is the private key and (p, q, g, y) is the public key

To sign a message, the following steps are performed;

1. User selects a random k with $1 \leq k \leq q-1$
2. Computes $r = (g^k \, mod \, p) \, mod \, q$
3. Computes $k^{-1} \, mod \, q$
4. Computes $s = k^{-1} \, (h(m) + ar) \, mod \, q$

In the fourth step, $h(m)$ denotes a hash function, such as SHA-1. The signature is the pair (r, s). Verification of the message is as follows;

1. r and s is verified as $1 \leq r \leq q$ and $1 \leq s \leq q$

2. The verifier computes $w = s^{-1} \, mod \, q$
3. The verifier computes $u_1 = w \, h(m) \, mod \, q$ and $u_2 = rw \, mod \, q$
4. The verifier computes $v = (g^{u_1} \, y^{u_2} \, mod \, p) \, mod \, q$

The signature is verified, when $v = r$.

2.2.10. ECDSA

Another variation of DSA is the Elliptic Curve DSA (ECDSA) [53], which works with elliptic curve cryptography instead of integers of modulo prime p.

To sign a message, the following steps are performed;

1. User selects a random k with $1 \leq k \leq n - 1$
2. Compute $r = x_1 \, mod \, n$, where $(x_1, y_1) = kG$.
3. Compute $s = k^{-1} \, (z + rd_A) \, mod \, n$

z is the leftmost bits of $h(m)$. The signature is the pair (r, s). Verification of the message is as follows;

1. r and s is verified as $1 \leq r \leq n - 1$ and $1 \leq s \leq n - 1$
2. Compute $w = s^{-1} \, mod \, n$
3. Compute $u_1 = zw \, mod \, n$ and $u_2 = rw \, mod \, n$
4. Compute $(x_1, y_1) = u_1 \mathrm{G} + u_2 \, Q_A$

z is the leftmost bits of $h(m)$. The signature is verified, if $r = x_1 \, mod \, n$.

2.3. Related Work about Mobile Communication Security

Before explaining our proposed security framework, we will briefly summarize other works about the subject.

In [35], authors propose a security protocol to use with mobile payment systems over short message service. The proposed protocol provides confidentiality, integrity, authentication and non-repudiation of short messages. This model involves generating a public/private key pair with a certificate authority and distributing them. The private key is stored at the SIM card of the user. The protocol ensures secure communication between a customer and a merchant.

In [37], authors propose an end-to-end security protocol for short message communication over the GSM network. In this protocol, each user has a public and private key pair and a certificate verified by an authority. After certificate distribution, users exchange their public keys with each other in order to verify the signature of short message and decrypt it.

In [38], authors present a software framework written in Java language, which provides an end-to-end security between two users over the short message communication. The messages are encrypted and digitally signed in order to provide confidentiality and authentication. The secure communication between two users is accomplished using public key cryptography. For the key exchange, they minimize the number of messages sent between users in order to prevent costs at the operator.

3. THE PROPOSED SECURITY FRAMEWORK FOR GSM

The security vulnerabilities in a GSM network were explained in the previous chapters. In this thesis, we have built a security mechanism to ensure secure communication between two parties over the GSM network. The main purpose of this protocol is to overcome the security vulnerabilities in the network. We first developed a peer-to-peer key-exchange protocol, which will be explained in Section 3.1. This key exchange protocol is adopted to use for voice communication and for text messaging. These applications will be explained in Section 3.2.

3.1. Peer-to-peer Key Exchange Protocol for Multiple Sessions

Our proposed protocol allows secure key exchange and communication between two parties, by exchanging their certificates with each other. Those certificates are generated by a certificate authority (CA) and both parties have the public key of CA to verify the certificate. The certificates are signed using ECDSA algorithm.

The key exchange is achieved by using Elliptic Curve Diffie Hellman (ECDH). After the key exchange is done, a double hash chain is used to derive different symmetric keys during communication. In the SMS version of the protocol, every message is encrypted using a different key generated from the double hash chain. In this way, backward and forward secrecy is achieved. For the voice communication, a new key is derived using the hash chain for every new communication session.

The notations used in the protocol are given in Table 3.1;

Table 3.1 – Notations in the protocol

S	Shared secret after ECDH
S_1	Seed of the first hash chain
$MSISDN_A$	Phone number of user A
$H(x)$	Hash value of x
M	Message
N	Negotiated length of the hash chain
K_i	i^{th} symmetric key during communication
$E(x)$	Symmetric encryption of x
exp_date	Expire date for hash chain
R_A	ECDH Public key of user A
n_A	ECDH Private key of user A
IP_A	IP address of user A
$\|\|$	concatenation
$Sign_A(X)$	Signature of message X by user A
$Cert_A$	Certificate of user A
$ECDSA_A$	ECDSA Public Key

In the initialization phase, the computational powers of the mobile devices are measured. This is important for deciding the length of the hash chain, because longer hash chains require longer time to be calculated on slow processors. Longer calculation time means bad user experience during communication. In this phase, our protocol calculates how many hash operations the processor can perform for a constant time. The result is the number N. During the handshake both parties send each other their hash chain length (N) and the smaller number is selected as the hash chain length of the protocol. Along with the number N the user also sends his certificate $Cert_A$. This certificate contains the following information;

- Elliptic Curve Digital Signature (ECDSA) Public Key
- Phone number of A
- Expiration date for the certificate
- Public key of the certification authority (CA)
- Signature of the CA for the certificate

The handshake phase is shown in Figure 3.1a.

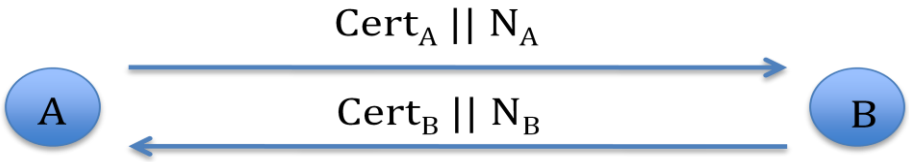

Figure 3.1a – Handshake message

For the case that a user does not want to obtain and exchange certificates, we also offer a simplified version of the handshake. In this version the user A sends his ECDSA public key to user B with number N. When B receives the ECDSA public key of user A, B has to confirm that this public key belongs to A. If B denies the handshake message, meaning that the ECDSA public key does not belong to A, the protocol is canceled. The alternative handshake message is shown in Figure 3.1b.

Figure 3.1b – Alternative handshake message

After the handshake shown in Figure 3.1a, both parties verify other parties' certificate with the public key of CA. Once the certificates are verified, the protocol can be initiated. If the users have chosen not to exchange certificates, they have to confirm manually that they have exchanged legitimate ECDSA public keys. After that, the initiator A sends his ECDH public key R_A, new session identifier Q and the chosen expire date exp_date. The user also adds a signature as a proof that the message belongs to him. The signature is generated with the following data;

$$\text{Sign}_A(\text{MSISDN}_A \,||\, Q \,||\, R_A \,||\, \text{exp\_date}) \tag{3.1}$$

The receiver party B verifies the signature with the certificate of A $Cert_A$. If the signature or data is tampered, the protocol is canceled. The key exchange phase of the protocol is shown in Figure 3.2.

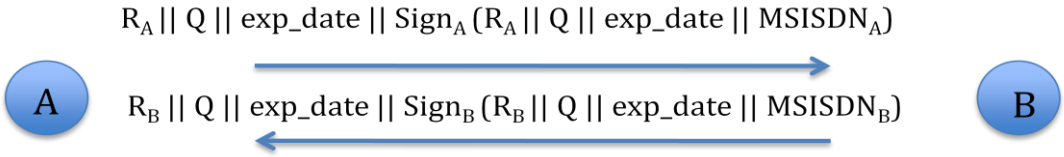

Figure 3.2 – Key exchange

In this step, B calculates the shared secret S using Elliptic Curve Diffie Hellman explained in Section 2.2.2. After receiving the public key R_A, B is able to calculate the shared secret S as;

$$S = R_A \times n_B, \tag{3.2}$$

where R_A is the ECDH public key of A and n_B is the ECDH private key of B. After that, B also sends to A his ECDH public key R_B along with new session identifier Q and expire date exp_date. The signature for this message is as follows;

$$\text{Sign}_{B}\ (\text{MSISDN}_{B} \,||\, Q \,||\, R_{B} \,||\, \text{exp\_date}) \quad (3.3)$$

After receiving these values, A verifies the signature with the certificate of B. If the message is not tampered, A also calculates the secret value S as;

$$S = R_B \times n_A \quad (3.4)$$

For key generation, a double hash chain is used. In order to build two hash chains, we need to generate two different seeds from the shared secret S. The seed generation is as follows;

$$S_1 = H(S \,||\, \text{"First seed"}) \quad (3.5)$$

$$S_2 = H(S \,||\, \text{"}Second\ seed\text{"}) \quad (3.6)$$

To calculate a key, the values of both hash chains are XORed. To calculate k_i the user needs to calculate $h^i(S_1) \oplus h^{N-i}(S_2)$.

The protocol ends when the expiration date is reached or the hash chain is totally consumed. In this case, the protocol needs to be restarted in order to build new hash chains with new seeds.

3.2. Applications of our Key-Exchange Protocol

The protocol is adopted into two areas of GSM communication. One version is used in text messaging, where every message is encrypted using a different key. The other version is used in voice communication, where users communicate using AES encryption over the data network.

3.2.1. Securing SMS Communication

The main focus of the protocol was to improve the security of the text messages. Text messages are the most widely used communication method after voice calls [44]. The security of messages is very important due to the information in the message content. It is known to us that many companies interact with their clients or end users over SMS channel and exchange personal information which cause problems if eavesdropped by a third party [1]. Also financial sector and banks send personal information or one-time passwords for their on-line operations to their

customers via SMS [10]. If the contents of those messages are exposed to an attacker, the result can be the loss of valuable information or money for end users.

Text messages have short data transfer capabilities. The international limit for a text message is 160 characters for ASCII alphabet. Which means that $160\,x\,7 = 1120$ bits of data can be sent in each message to the receiver [1]. Therefore, a security protocol should consider this important drawback. If a key exchange is performed before the communication, more than one message could be sent between two parties because of the long key sizes of key exchange algorithms like RSA [40]. An application with sending multiple messages is not acceptable for an end user, since the mobile operator charges for every message. It is also known that short key sizes are no more considered to be secure [19]. Considering these drawbacks, Elliptic Curve Diffie Hellman is chosen as the key exchange method in our protocol. It provides higher security at lower key sizes [19] and also known to have better performance compared to RSA.

Text message version of the protocol starts with the initialization phase, where the performances of both devices are measured. This way the length of the hash chain is determined after the handshake as explained in Section 3.1. The lower N value is chosen as the hash chain length during the session. Furthermore, users exchange their certificates to verify the signatures in the key exchange phase.

After the handshake, the initiator A sends the following values to B; ECDH public key $R_{A,}$ expire date for the session exp_date and a new session identifier Q. A session is defined as the time until expiration date is reached or the number of messages has reached the length of the hash chain. In this way, the new session identifier specifies the beginning of a new session with fresh variables. Along with the message, A also sends a signature containing the phone number ($MSISDN$) of A, session identifier Q, public key and expire date.

$$\text{Sign}_{A}\,(\text{MSISDN}_{A} \,||\, Q \,||\, R_{A} \,||\, \text{exp\_date}) \qquad (3.7)$$

The signature is sent to prove that the message is not tampered and the message is from A. An attacker C, who is eavesdropping on the communication between A and B, cannot modify the message, since he cannot generate the same signature without knowing the private key for ECDSA. After receiving the message, B verifies the

signature with the ECDSA public key of A.

In this step, B can calculate the secret S as follows;

$$S = R_A \times n_B \tag{3.8}$$

In order to calculate the same secret S, A needs the ECDH public key R_B of B. B sends his ECDH public key R_B, the expire date exp_date, which he received from A and the new session identifier Q. Expire date exp_date and session identifier Q are send back to A to make sure that both parties have the same values. B also sends the signature of his phone number $MSISDN_B$, R_B, expire date exp_date and session identifier Q. The signature is;

$$\text{Sign}_B\ (MSISDN_B \,||\, Q \,||\, R_B \,||\, \text{exp\_date}) \tag{3.9}$$

After A has received the message, it verifies the signature. A also checks expire date exp_date and session identifier Q with his own values. If everything matches, A does the following calculation;

$$S = R_B \times n_A \tag{3.10}$$

At this step the hash chain can be generated. As said before, we use double hash chain of the length N with the following seeds;

$$S_1 = H(S \,||\, \text{"First seed"}) \tag{3.11}$$

$$S_2 = H(S \,||\, \text{"}Second\ seed\text{"}) \tag{3.12}$$

For every message during the session, we use a different key. The key generation for each message is done with the help of the double hash chain. To calculate the key K_i for the i^{th} message in the session, the i^{th} value of one hash chain is XORed with the value of $N - i^{th}$ value of the other hash chain.

$$K_i = H^i(S_1) \oplus H^{N-i}(S_2) \tag{3.13}$$

In order to keep track of the synchronization, the number i should be sent with the message. When messages get lost in the SMSC or arrive later than the messages sent after, the synchronization between A and B can be lost. To prevent this, we send

the sequence number of the message unencrypted attached to the cipher text. If the user receives an encrypted message, he will know which key to generate from the hash chain.

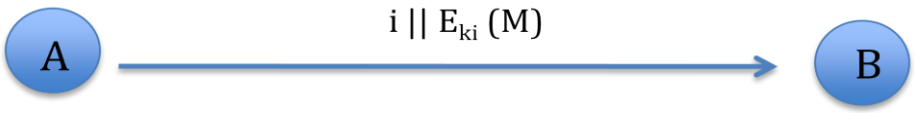

Figure 3.3 – Encrypted message contents

The session is over when the number of messages has reached the length of the hash chain or the expiration date is reached. In this case, the session needs to be restarted to generate a new hash chain with a fresh seeds.

The advantage of using a new key for every message is to improve the security of every message. Let's say a third person C gets the symmetric key for an arbitrary message during a session. Since every message is encrypted with a different key, C cannot decrypt other messages even if he has eavesdropped on the entire communication. He also cannot generate any other key from the compromised key, since the key generation method provides forward and backward secrecy.

3.2.2. Securing Voice Communication

The standard voice communication of the GSM network is encrypted with the algorithm A5, which is proven to be insecure [32]. The second implementation of the protocol aims to provide extra security for the voice communication. The difference from the GSM voice communication is that we use the data network. 3G networks provide enough bandwidth to perform a voice call [16].

In our protocol, both parties will be able to call each other over the data network by using their phone numbers. To achieve that, both users exchange their *IP* addresses with each other in the initialization phase. Otherwise, a third party server is used to match *IP* addresses of those two users. The advantage of sharing the *IP* address over SMS is that both parties will be independent from a third party server. The packets are sent encrypted over the data network, and decrypted at the receiver.

At the initialization phase, both parties calculate their hash length *N* and send it

to each other. Also the certificates are exchanged at this phase. This process is shown in Figure 3.4.

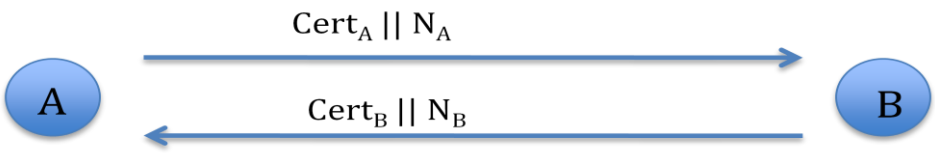

Figure 3.4 – Voice communication handshake message

The IP address is needed to make the data connection between two devices. Since we do not use a third server to store the IP addresses with matching phone numbers, the IP address exchange is crucial. After the handshake, A sends his public key R_A, expire date exp_date, IP address IP_A, new session identifier Q and the following signature;

$$\text{Sign}_A\,(MSISDN_A \,||\, IP_A \,||\, Q \,||\, R_A \,||\, exp\_date) \qquad (3.14)$$

After the signature verification, B sends his protocol variables to A. The key exchange is show in Figure 3.5.

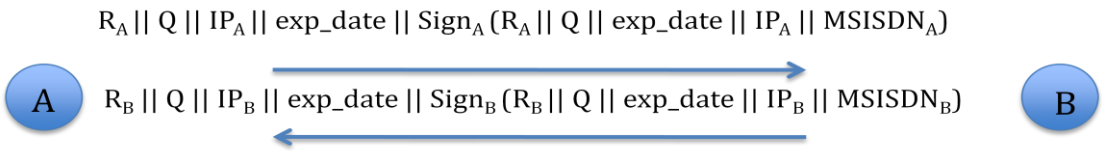

Figure 3.5 – Voice communication key exchange

After B verifies the signature of the message, it calculates the shared secret S as follows;

$$S = R_A \times n_B \qquad (3.15)$$

In order to calculate the same secret S, B sends his own ECDH public key R_B with expire date exp_date, IP address IP_B, new session identifier Q and the signature;

$$\text{Sign}_B\ (MSISDN_B \,||\, IP_B \,||\, pw \,||\, Q \,||\, R_B \,||\, exp\_date) \qquad (3.16)$$

After A has received the message, it verifies the signature. User A also checks expiration date exp_date and session identifier Q with his own values. The shared secret is calculated by A as;

$$S = R_B \times n_A \tag{3.17}$$

The shared secret S is used to generate two different seeds for the double hash chain. The seeds S_1 and S_2 are generated as;

$$S_1 = H(S \,||\, \text{"First seed"}) \tag{3.18}$$

$$S_2 = H(S \,||\, "Second\ seed") \tag{3.19}$$

For the first voice communication, the first key is used according to the following key generation method;

$$K_i = H^i(S_1) \oplus H^{N-i}(S_2) \tag{3.20}$$

The synchronization problem in the text message protocol is solved differently in the voice implementation. Here, we send the number i to generate the key K_i as the first data packet over the data network. The receiver party will know which key to generate from the hash chain. The voice chat ends after receiving a packet that says that the voice chat is over. After that, the program listens to the packet, which will start a new chat and generate a new key K_i.

$$Start\ (i) -> E_{K_i}(M) -> \ \ldots\ldots -> E_{K_i}(M) -> End\ (i) \tag{3.21}$$

The session is over when N voice calls are made between user A and user B or the expiration date is reached. Another reason to restart the protocol is the change of IP address. The IP address can change because of the data network connection.

We do not encrypt every packet with a different key during a voice communication because of two reasons. First, the number N is too small for the number of packets sent during a voice call. Second, it requires a powerful processor to generate a different key using the key generation method above for every packet during the voice stream. The encryption and decryption of voice packet should be done fast in order to have a smooth voice chat.

3.3. Security Analysis

Here we explain the security analysis of both protocols. The advantages and disadvantages will be explained in detail.

3.3.1. Header Change Attacks on Short Messages

Since the header is not protected in the text messaging infrastructure of GSM, it can be maliciously modified. The sender information of a text message is stored in the header. Any person, who can modify the header of a short message, can impersonate a phone number. In the protocol, the phone number $MSISDN$ is protected by the signature. Once the certificate is verified by the public key of CA, both users know that the message is signed by the sender.

3.3.2. Man in the Middle Attack

The man in the middle attack in Diffie-Hellman key exchange scenarios involves a third device listening both parties and also sending messages to them. During the key exchange, the attacker receives the public key of A and responds with his own public key, while it will also send B his own public key and makes a key exchange with both A and B. Since there is no authentication, A and B will think that the key exchange is successful and begin their "secure" communication, but every message that A sends will be decrypted by the attacker and re-encrypted for B. This way, the attacker will listen to the entire communication.

Since all the variables are signed with a valid signature at the seed exchange phase of our protocol, a third party cannot perform a man-in-the-middle attack, because it needs to know the ECDSA private key to generate a valid signature.

3.3.3. Forward and Backward Secrecy

If an attacker somehow gets the key for one SMS during the text messaging, it will be impossible to calculate any future or past keys from that key. Also in the voice communication, when an attacker gets the key for one voice chat, he cannot produce other keys used in one session. Since the elements of two chains are XORed to calculate the key, the attacker cannot know any values of the hash chains. Even if the attacker somehow gets the value of one hash chain, the hash function will be secure

enough to protect the seed of the chain from attacker. If an attacker gets a key during the session, he cannot calculate the keys before that key, since it is not possible to go backward in the hash chain. Therefore, we provide backward secrecy. Also it is impossible for an attacker to calculate future keys in the session, since he has to go back in at least one hash chain, which is impossible according to the one way property of hash functions. Forward secrecy is achieved in this way.

3.4. Further Discussion

As symmetric cipher, we use AES-128 in counter mode. Because of the bit limitation of the text message ($160 * 7 = 1120$ bit for message body), symmetric operation modes that require padding of plain text to fix the block sizes are not suitable. Padding could result in sending multiple messages, which is not preferred to avoid extra charges by the mobile operator. As the hash algorithm, SHA-256 is chosen to provide equal security strength.

The elliptic curve used for Elliptic Curve Diffie Hellman is created with the parameters $secp256r1$ as stated in [23]. This curve is a 256-bit elliptic curve over $\mathbb{F}_p$with the following parameters;

p = FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFF

FFFFF

a = FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFF

FFFFC

b = 5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E

27D2604B

The curve is defined as $E: y^2 = x^3 + ax + b$ over $\mathbb{F}_p$. The base point G, the order n and cofactor h are;

G = 046B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945

D898C2964FE342E2FE1A7F9B8EE7EB4A7C0F9E162BCE33576B315ECECBB64

06837BF51F5

n = FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2F

C632551

Message sizes during protocol initialization are optimized as follows;

1. Handshake messages contain the hash chain length N, which is an integer of size 16 bits. The certificate contains the ECDSA public key with 256 bits, public key of certificate authority with 256 bits, signature with 520 bits and expiration date with 32 bits. The message size is calculated as $16 + 256 + 256 + 520 + 32 = 1080$ bits, which can fit in a text message.

2. Key exchange message includes the public key R, the new session identifier Q and the exp_date. The public key is of size 256 bits. The expiration date exp_date is a timestamp with the size 32 bits. The signature costs 520 bits. The session identifier Q is a number with 8 bits. The message size is calculated as $256 + 32 + 520 + 8 = 816$ bits. In the voice communication protocol, the IP address is included in the message, which can be 15 characters at max.

4. TESTS AND RESULTS

In this section we will give performance test results for the implementation of protocols in Section 3. The performance tests are based on the running time of symmetric encryption, digital signature and key agreement. Furthermore, for the voice communication the 3G network performance is measured for various places in a city. In section 4.1 the development platform for the voice and text communication protocol will be explained. In Section 4.2 performance test results for text messaging protocol will be given. Voice communication performance results are given in Section 4.3.

4.1. Development Platform

As development environment Android and J2ME platforms were chosen. J2ME is currently the most widely used [44] application platform in mobile phone industry. Mainly focused on non-smart phones, J2ME offers to develop powerful applications in Java language on devices with limited computing capabilities. Android is an open source operating system focused on smart phones. Developed by Google and supported by Open Handset Alliance [45], Android has become the most emerging platform in the smart phone world. Android is basically a linux system with a Java Virtual Machine (dalvik) [45] build on top of it. JVM allows the operating system to run rich Java applications, which can use the capabilities of phone's hardware. Like J2ME, Android does not support the standard Java JDK; it owns his own libraries and a development kit [45].

The iOS platform, which is used in Apple's mobile products like iPhone and iPad [46], is another powerful operating system known by its stability. Unlike Android, iOS does not run applications on a virtual machine, which results in better performance for Apple devices [46]. Although this platform seems to be a good alternative for developing mobile applications, it lacks the flexibility of accessing device's core abilities like receiving and sending SMS messages in background. Therefore, our protocol cannot be realized on Apple platforms.

For the implementation of the protocol, Bouncy Castle cryptography library is used. Bouncy Castle library provides a large set of crypto algorithms for developer. The elliptic curve Diffie Hellman, Elliptic Curve Digital Signature Algorithm and

AES are both available in the package. It is also supports J2ME and Android (Java) platforms [5]. To the best of our knowledge Bouncy Castle is the most popular and widely used open source project with support for elliptic curve cryptography. The library can be downloaded from its web site (http://www.bouncycastle.org) and the appropriate jar files can be easily mounted into any project. Bouncy castle also supports C# language, but Windows Mobile platforms are not in the scope of this thesis.

Bouncy Castle is often criticized because of his performance on JVM. Although there are other non Java solutions for calculating elliptic curve operations on mobile devices with low computational power [52, 54, 55], we choose Java environment because of its wide usage. Furthermore, an application written in Java is easy to export and distribute among users in platform independent way.

The Java supported platforms, both J2ME and Android, are very suitable to implement SMS applications, since they allow low-level access to text messages. In J2ME platform, it is possible to send binary SMS, which makes it possible to send encrypted data array without converting to base64 format. Android also supports binary SMS. Furthermore, with Android it is possible to access mobile phones SMS storage and to listen to SMS traffic in background. With the background process in Android operating system, the user does not have to run the application as the main process. The encrypted messages are decrypted on the background when they arrive and then stored in the database. The default pre-installed messaging application of the phone cannot read encrypted messages; they can only be read from the application.

Android provides rich streaming libraries for voice and video streaming. Besides streaming, the developer can also implement networking applications, which support the network communication libraries of Java language. Our secure voice communication application sends encrypted packets over the UDP protocol.

4.2. Performance Tests for Message Communication

For the message communication, the following measurements are done;

1. ECDH Key Agreement calculation time
2. ECDSA signature verification and generation time

3. Hash chain generation time
4. AES encryption performance

4.2.1. Devices Used in Message Communication Tests

For performance tests, we used different mobile phones from various vendors. The CPU metrics are taken from jBenchmark, a famous site known for performing tests on every mobile device on the market [4]. The CPU comparison for our test phones can be seen in Table 4.1.

Table 4.1 – Processors of different mobile phone hardware

	SE K750i	**SE W580i**	**SE K550i**	**HTC Touch HD**
Processor Speed [4]	ARM9 110 MHz	ARM9 187 MHz	ARM9 201MHz	Qualcomm® MSM 7201A™ 528 MHz

The Sony Ericsson series K and W both support J2ME platform. The HTC Touch series run Windows Mobile as operating system. The J2ME support in this device is not native, which means the program runs with the help of a Java Virtual Machine.

4.2.2. Algorithm Performance Tests

In this section, we give performance results for symmetric encryption algorithms, ECDSA signature generation and ECDH key agreement. The key agreement phase for our protocol with 256-bit key size can be seen in Table 4.2.

Table 4.2 – ECDH Performance Results (seconds)

	SE K750i	**SE W580i**	**SE K550i**	**HTC Touch HD**
Processor Speed [4]	ARM9 110 MHz	ARM9 187 MHz	ARM9 201MHz	Qualcomm® MSM 7201A™ 528 MHz
ECDH Key Agreement (512 bits)	158,75	44,85	31,61	18,45
ECDH Key Agreement (384 bits)	152,80	27,38	19,30	8,033
ECDH Key Agreement (256 bits)	92,25	14,25	10,26	5,67

The calculation of the shared secret with 256-bit key size takes 152 seconds on the slowest hardware with 110 MHz CPU power. Smart phones with more powerful CPUs can run the same calculation with 256-bit key size in 8 seconds. Therefore, during the protocol, the application will require most of the time for the key agreement before the encrypted communication can start. For faster devices with 500+ MHz CPU, the key agreement will have less effect on the user experience.

Table 4.3 – ECDSA Performance Results (seconds)

	SE K750i	**SE W580i**	**SE K550i**	**HTC Touch HD**
Processor Speed [4]	ARM9 110 MHz	ARM9 187 MHz	ARM9 201MHz	Qualcomm® MSM 7201A™ 528 MHz
ECDSA Sign (512 bits)	205,45	59,52	43,19	24,92
ECDSA Verify (512 bits)	251,14	73,41	54,35	31,24
ECDSA Sign (384 bits)	165,74	30,01	19,73	8,34
ECDSA Verify (384 bits)	195,45	35,72	24,30	10,12
ECDSA Sign (256 bits)	67,85	10,12	7,21	3,62
ECDSA Verify (256 bits)	79,84	11,95	8,69	4,32

The signature generation takes 205 seconds on the slowest device with 512 bit key size. For the key size of 256 bits, which is used for the protocol, the key generation takes between 68 seconds and 7 seconds. For faster devices with 500+ MHz CPU, the signature generation will be calculated in less than 4 seconds. Verification of the signature takes a little longer. On the slowest devices, ECDSA signature verification takes 251 seconds, where the fastest device calculates in less

than 32 seconds for 512 bits key size.

Table 4.4 – Length of Hash Chains generated in one second

	SE K750i	**SE W580i**	**SE K550i**	**HTC Touch HD**
Processor Speed [4]	ARM9 110 MHz	ARM9 187 MHz	ARM9 201MHz	Qualcomm® MSM 7201A™ 528 MHz
SHA 256 (length of chain)	1045	5507	5769	10598
SHA 512 (length of chain)	482	1088	1113	3346

For the hash chain generation performance, we measured how long the hash chain will be for every device after one second. As an input x for the hash chain, the string "*testdata*" is given. The results can be seen in Table 4.3. When SHA512 is used as hash algorithm, the chain length for SE K750i measured as 482. For SHA256, we get a hash chain with length 1045 after one second. For the fastest processor, we get a hash chain of length 10598 for SHA256 and a hash chain of length 3346 for SHA512.

Table 4.5 – AES encryption performance

	SE K750i	**SE W580i**	**SE K550i**	**HTC Touch HD**
AES 128 (milliseconds)	1,20	0,24	0,23	0,10
AES 256 (milliseconds)	1,51	0,30	0,29	0,12

AES performance is measured as the time elapsed for encrypting the string "*deneme12deneme12*". The performance of AES128 on slowest processor is measured as 1,20 milliseconds. The fastest CPU can encrypt the string in 0,10 milliseconds. Therefore, we can say that symmetric encryption causes almost no delay on the protocol.

4.3. Performance Tests for Voice Communication

For the voice communication, the following measurements are done;

1. AES Encryption performance
2. Average end to end delay for EDGE and 3G network
3. 3G network performance

4.3.1. Devices Used in Voice Communication Tests

For the voice implementation, we targeted smart phones, since they offer more CPU power. Two devices were chosen, Samsung i7500 and Google Nexus. The performance metrics for these phones can be seen in Table 4.5.

Table 4.6 – Device Comparison Table

	Samsung i7500	**Google Nexus**
Processor	Qualcomm MSM7200A 528 Mhz	1 GHz Qualcomm QSD 8250 Snapdragon ARM
Memory	192 MB	512 MB DRAM
OS	Android 1.5	Android 2.2

The Samsung i7500 is one of the first Android devices on the market and the Google Nexus is the first official phone from Google himself. Both devices are powerful smart phones with high computational power and Android operating system. The implementation is written using Android 1.5 SDK. Since Android is a backward compatible operating system, applications written in older SDKs can also run in future versions. Therefore, for Google Nexus we used the same implementation.

4.3.2. Encryption Test Results

Since the packet sizes and CPU powers are different than text messaging implementation, we also performed encryption tests for voice implementation.

Table 4.7 – Average Encryption / Decryption Times

	Samsung i7500	**Google Nexus**
Processor	Qualcomm MSM7200A 528 MHz	1 GHz Qualcomm QSD 8250 Snapdragon ARM
AES Encryption (milliseconds)	32	5
AES Decryption (milliseconds)	37	7

The decryption time is the time after the device receives the packet as an encrypted byte array and decrypts it. The encryption time is the time when the sound is turned into byte array and gets encrypted before sending to the receiver. As can be seen from Table 4.6, the powerful the hardware, the shorter it takes to encrypt and decrypt the packets. The packet size for the encryption is bigger than the packets used in message encryption. The packet size is 16384 bits.

The overhead of the symmetric encryption on the protocol is negligible, since the encryption and decryption delay has no effects on user experience during communication.

4.3.3. Network Performance

The network performance is important for the conversation quality during a voice communication. More delay means bad user experience during a conversation, therefore we also measured the average delay for GSM and 3G networks.

Table 4.8 – Average End to End Delay

	Average Delay
EDGE	32 s
3G	9 s

The advantage of 3G networks can be seen from the results in Table 4.7. The average end to end delay is significantly higher than standard data networks. But 9 seconds average in 3G network is still not enough to perform a conversation, since the delay is too high. Since the performance of the network dependent to population and network coverage, we will also examine the impact of location in the next experiment.

Table 4.9 – Average Delay in Different Location over the 3G Network

	Location 1	**Location 2**	**Location 3**	**Location 4**
Average Delay	34 s	3,6 s	5,0 s	4 s
Standard Deviation	25,4 s	3,3 s	3,9 s	3,2 s
Latitude	40.792406	40.982338	41.021791	41.0707635
Longitude	29.467952	29.105075	29.041483	29.061937
Province	Gebze / Kocaeli	Kozyatağı / Kadıköy	Altunizade / Üsküdar	Kandilli / Üsküdar

From the results of location comparison in Table 4.8, we can see that the 3G network performance is highly affected from the location. The main reasons for differences of the performance are the distance from the base station, the population in the area and the network coverage. Also at some places different 3G standards are available, like dual channels, HSDPA or HSPA, which result in different download and upload speeds.

In all locations, there is a variable delay. The standard deviation for the first location is calculated as 25.4 seconds, which means that the delay changes between 9 seconds and 59 seconds. For other locations, where there is an average delay between 3.6 to 5 seconds, the standard deviation is also high. That means the mobile network cannot guarantee an equal delay for every data packet.

Location 4 in Table 4.8 is an urban area with low population, which has a positive effect on the data performance. Location 3 is more crowded, which results in more delay. Location 2 is also a populated urban area, but number of base stations and network coverage is higher than Location 3. Although Location 1 has less population than other locations, the number of base stations and network coverage is low, because it is a rural area. Therefore, Location 1 has the highest delay.

The results show that voice communication can be done over the data network. With the old generation data network, the sound quality should be as low as possible, but still the average end to end delay shows that there will be long delays during communication. 3G networks allow faster data communication. However the performance of the network depends on coverage, population around the base station and location. There can be significant differences in the network performance, which affect the sound quality badly.

5. CONCLUSION

The security vulnerabilities in current GSM networks allow eavesdroppers to monitor entire communication between the mobile device and the base station over the air. Over the air communication is encrypted using the A5 cipher. However, A5 algorithm is a weak one and has already been broken [20]. Furthermore, SS7, the communication protocol between the base station and operator, has no encryption.

In this thesis, we designed and implemented a multipurpose security platform for mobile communication over GSM / 3G. First, we developed an authenticated key exchange protocol using Elliptic Curve Digital Signature Algorithm and Elliptic Curve Diffie Hellman algorithm. After the key exchange, keys for symmetric encryption are generated with a method using double hash chains. Double hash chain provides forward and backward secrecy for messages during communication. In this way, if any key during the communication is compromised, earlier or future keys cannot be generated from the compromised key.

The key exchange protocol is applied to short message communication, which is the most popular service in a GSM network after voice communication. According to GSM standards, short messages are not end-to-end encrypted, such as header modification and eavesdropping. In our protocol, two users run ECDH for agreeing on a secret key. After the key exchange, a different symmetric key for each message is generated. The key generation method provides forward and backward secrecy.

The key exchange protocol is also applied to voice communication. In our protocol, encrypted voice communication is done over the data network. Therefore, users need to share their IP addresses before the key exchange. During the initialization before the key exchange, IP addresses are sent to each other. The key exchange is done over the short message protocol. After the handshake, the voice communication is performed over the data network. Like the short message protocol, the symmetric keys for each session are generated using a double hash chain. If the symmetric key of a session is compromised, earlier or future keys cannot be generated from the compromised key.

In this thesis, we also implemented our protocols for voice and message communications. For the development, we used open-source Android platform

developed by Google. For ECDH, ECDSA and AES implementations, we used an open-source library called Bouncy Castle, which provides implementations of many popular encryption algorithms in Java.

Although mobile devices are known for their lack of computational power, performance tests show that AES symmetric encryption with 128-bit key sizes for a 16 characters long string can be done in less than 1 millisecond. ECDH key agreement with 512-bit key size can be calculated on the slowest CPU with 110 MHz around 158 seconds. Faster machines, for example smart phones with 500+ MHz CPU, require less than 10 seconds.

For the voice communication protocol, we also measured the data network performance for various places in the city. First, the network type (GSM and 3G) performance is measured. GSM networks show slow performance on data communication, even if EDGE is used as the connection type. On the other hand, 3G networks offer more bandwidth and less delay. Despite the bandwidth they offer, 3G network performance is highly affected by population and topography. Our tests show an average delay between 3 seconds to 34, which is highly unstable.

6. REFERENCES

[1] Johnny Li-Chang Lo, Judith Bishop, J.H.P. Eloff (2008), SMSSec: An end-to-end protocol for secure SMS, *Computers&Security, Volume 27, Issues 5-6, October 2008, Pages 154-167*

[2] Marko Hassinen (2005), SafeSMS - End-to-end encryption for SMS Messages, *Proceedings of the 8th International Conference on Telecommunications, 2005. ConTEL 2005*

[3] He Rongyu , Zhao Guolei, Chang Chaowen, Xie Hui, Qin Xi, Qin Zheng (2008), A PK-SIM card based end-to-end security framework for SMS, *Computer Standards & Interfaces, Volume 31, Issue 4, June 2009, Pages 629-641*

[4] jBenchmark (2010), *http://jbenchmark.com/,* retrieved on 03/05/2010

[5] BouncyCastle (2010), *http://www.bouncycastle.org*, retrieved on 09/01/2010

[6] Kemal Bicakci, Nazife Baykal (2002), Infinite Length Hash Chains and Their Applications, *Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002.*

[7] Maw-Jinn Tsaur, Wei-Chi Ku, and Hao-Rung Chung (2008), An Improved Password-Based Authenticated Key Agreement Scheme for Pervasive Applications, *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2008. EUC '08.*

[8] ITU Internet Report 2006: digital.life, Chapter 3, *http:// www.itu.int/osg/spu/publications/digitalife/docs/digital.life-chapter3.pdf,* retrieved on 06/05/2010

[9] GSM / 3G Stats, *http://www.gsacom.com/news/statistics.php4,* retrieved on 30/01/2011

[10] Garanti Mobil İmza, *http:// www.garanti.com.tr/subesiz/internet_bankaciligi/guvenlik/mobil_imza/*, retrieved on 03/05/2010

[11] SS7 Tutorial, *http://pt.com/page/tutorials/ss7-tutorial*, retrieved on 09/09/2010

[12] Exploiting Open Functionality in SMS-Capable Cellular Networks, *http:// www.smsanalysis.org/*, retrieved on 03/04/2010

[13] SHORT MESSAGING SERVICE (SMS), *http://www.ida.gov.sg/doc/Policies%20and%20Regulation/Policies_and_Regulation_Level3/information_on_sms.pdf*, retrieved on 04/03/2010

[14] The 7 bit default alphabet, *http://www.dreamfabric.com/sms/default_alphabet.html*, retrieved on 09/02/2010

[15] Digital cellular telecommunications system (Phase 2+); Alphabets and language-specific information (GSM 03.38), *ETSI, GSM 03.38 Version 5.3.0: July 1996*

[16] Kaveh Pahlavan, Prashant Krishnamurthy, Principles of Wireless Networks, *Prentice Hall, ISBN 0-13-093003-2*

[17] UCS2 SMS, *http://www.gsm-modem.de/ucs2.html*, retrieved on 09/03/2010

[18] Turkcell, *http://www.turkcell.com.tr/yardim/teknolojisozlugu/c*, retrieved on 02/05/2010

[19] Recommendation for Key Management, *Special Publication 800-57 Part 1, NIST, 03/2007.*

[20] Timo Gendrullis, Martin Novotný and Andy Rupp, A Real-World Attack Breaking A5/1 within Hours, *Springer Berlin / Heidelberg, ISBN 0302-9743*

[21] NIST Recommendations for Key Lengths (2007), *http://www.keylength.com/en/4*, retrieved on 05/06/2010

[22] GSM Cloning, ISAAC, University of California, Berkeley,

http://www.isaac.cs.berkeley.edu/isaac/gsm.html, retrieved on 09/09/2010

[23] SEC 2: Recommended Elliptic Curve Domain Parameters, *Certicom Research, September 20, 2000, Version 1.0*

[24] William Stallings, Cryptography and Network Security, *Prentice Hall, ISBN 0-13-187316-4*

[25] Short Message Infrastructure Figure, *http://telco-learn.blogspot.com/2009/05/wireless-short-message-service-sms.html*, retrieved on 01/01/2011

[26] Apple iPhone Developer Center, *developer.apple.com/iphone/*, retrieved on 09/09/2010

[27] Telecommunications Site, *http://blog-telecom.blogspot.com/2010/01/short-message-service.html*, retrieved on 01/11/2010

[28] HTC Mobile, *http://www.htc.com*, retrieved on 02/11/2010

[29] GSM Infrastructure Figure, GSM Architecture Training Document, *Nokia, 2002*

[30] Andreas Enge, Kluwer Academic Publishers, Elliptic Curves and Their Applications to Cryptography, *Hardbound, ISBN 0-7923-8589-6*

[31] William Stallings, Wireless Communications & Networks, *Prentice Hall, ISBN 0-13-196790-8*

[32] Dieter Gollmann, Computer Security, *Wiley, ISBN 0-470-86293-9*

[33] Cell Phone Sketch Image, *http://www.funnycoloring.com/img/mobile-phone-3-b3590.jpg*, retrieved on 01/01/2011

[34] GSM Authentication Figure, *http://www.computerjagat.org/online_issue_article.php?newid=20*, retrieved on 09/09/2010

[35] Mohsen Toorani, Ali Asghar Beheshti Shirazi (2008), SSMS - A Secure SMS Messaging Protocol for the M-Payment Systems, *IEEE Symposium on Computers and Communications, 2008. ISCC 2008.*

[36] Hany Harb, Hassan Farahat, and Mohamed Ezz (2008), SecureSMSPay: Secure SMS Mobile Payment Model, *2nd International Conference on Anti-counterfeiting, Security and Identification, 2008. ASID 2008.*

[37] Diffie – Hellman Key Exchange Figure, *http://philipfox.net/dh/dh.html*, retrieved on 01/01/2011

[38] David Lisoněk, Martin Drahansky, SMS Encryption for Mobile Communication (2008), *International Conference on Security Technology, 2008. SECTECH '08.*

[39] Alfredo De Santis, Aniello Castiglione Giuseppe Cattaneo, Maurizio Cembalo, Fabio Petagna, Umberto Ferraro Petrillo (2010), An Extensible Framework for Efficient Secure SMS, *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), 2010*

[40] Mary Agoyi, Devrim Seral, SMS SECURITY: AN ASYMMETRIC ENCRYPTION APPROACH, *6th International Conference on Wireless and Mobile Communications (ICWMC), 2010*

[41] Paul C. van Oorschot, Michael J. Wiener, Parallel Collision Search with Application to Hash Functions and Discrete Logarithms, *Proceedings of the 2nd ACM Conference on Computer and Communications Security, CCS '94*

[42] Wade Trappe, Lawrance Washington, Introduction to Cryptography with Coding Theory, *Pearson Education, ISBN 0-13-198199-4*

[43] Johannes A, Buchmann, Introduction to Cryptography, *Springer, ISBN 0-387-20756-2*

[44] Wikipedia, *http://www.wikipedia.org,* retrieved on 09/06/2010

[45] Android OS, *http://www.android.com/*, retrieved on 09/05/2010

[46] Apple, *http://www.apple.com*, retrieved on 04/09/2010

[47] SHA-1 Broken, *http://www.schneier.com/blog/archives/2005/02/sha1_broken.html*, retrieved on 09/12/2010

[48] Vincent Rijmen and Elisabeth Oswald, Update on SHA-1, *CT-RSA 2005*

[49] New Cryptanalytic Results against SHA-1 ,*http://www.schneier.com/blog/archives/2005/08/new_cryptanalyt.html*, retrieved on 02/02/2010

[50] Thomas Page, The application of hash chains and hash structures to cryptography, *Technical Report. RHUL–MA–2009–18, 4 August 2009*

[51] L. Lamport, "Password Authentication with Insecure Communication", *Communications of the ACM 24.11 (November 1981), pp 770-772*

[52] Ahmadi, H.R., Afzali-Kusha, A., "Low-Power Low-Energy Prime-Field ECC Processor Based on Montgomery Modular Inverse Algorithm", *12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, 2009.*

[53] National Institute of Standards and Technology, Announcing Approval of Federal Information Processing Standard (FIPS) Publication 186–3, "Digital Signature Standard (DSS)", *Docket No.: 0810011295–81636–02*

[54] M. Aydos, E. Savas, and C. K. Koc, "Implementing network security protocols based on elliptic curve cryptography", *Proceedings of the Fourth Symposium on Computer Networks, May 20-21, 1999*

[55] E. Savas, T. A. Schmidt, and C. K. Koc, "Generating elliptic curves of known order", *Cryptographic Hardware and Embedded Systems - CHES 2001, Lecture Notes in Computer Science No. 2162, pages 145-161, Springer Verlag, Berlin, Germany, 2001*

APPENDIX

In this appendix, instruction to the implementations will be given. In the first section, the short message implementation will be explained. In the second section, the voice implementation will be given in more detail.

A1. SMS

The following is the user's manual for the Android version of the SMS encryption application. The application is developed using Android 1.5 SDK.

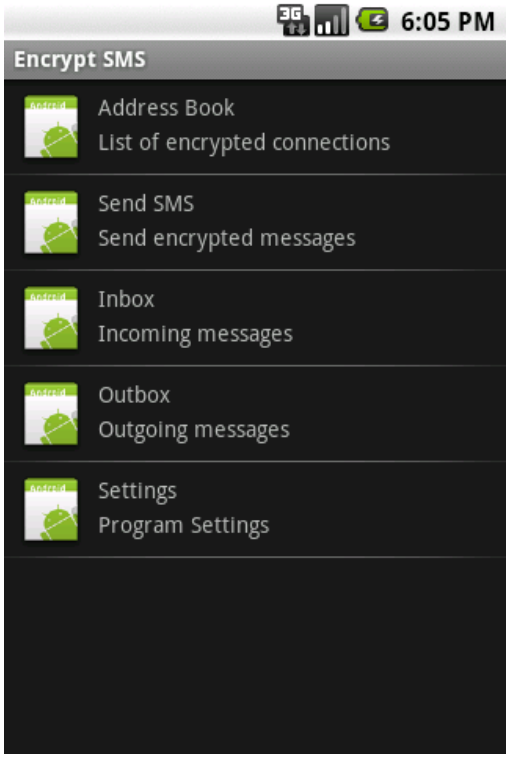

Figure A.1 – Main Screen

In the main screen, which is the first screen to appear when the application starts, the user has the following options as in Figure A.1;

1. Address Book: where the user can add / remove friends
2. Send SMS: where the user can send encrypted SMS to his contacts
3. Inbox: where encrypted SMSs from others are stored unencrypted
4. Outbox: messages the user has sent
5. Settings: protocol specific settings

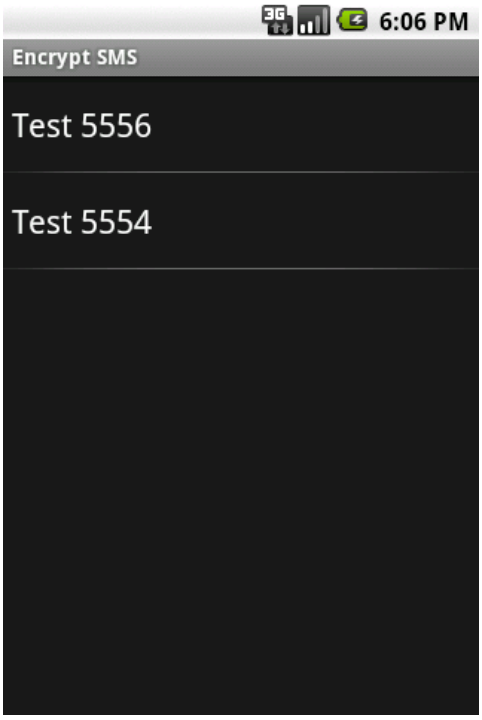

Figure A.2 – Address Book

In the address book, the user can see contacts he has created. Here, we have two contacts, Test 5556 and Test 5554 in Figure A.2. In order to create a new contact, the user presses the Menu button of his Android device and selects “Add new contact” from the menu. The user cannot send encrypted messages to a person, unless that person is in the contact list.

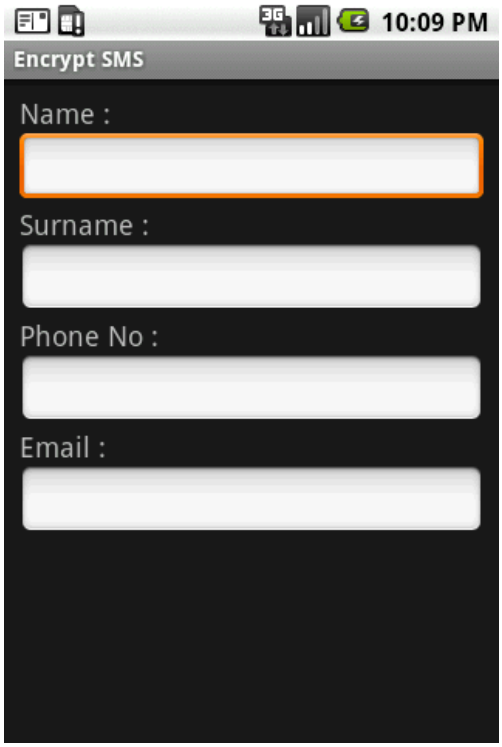

Figure A.3 – Add new user

In the “Add new user” screen, the user can create a new contact. The screen can be seen in Figure A.3. User email is reserved for future development. Currently it is not necessary. In order to send a new message or start the protocol, the user has to be recorded in the database.

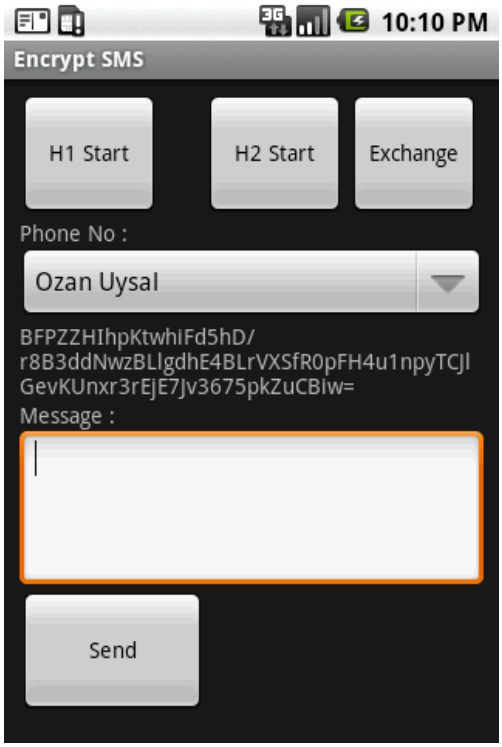

Figure A.4 – Send SMS Screen

Once the user has created his contacts, he can choose the name of the contact to send an encrypted SMS. If the user wishes to start a new protocol or restart an existing one, he presses either the “H1 Start” or “H2 Start” button for the choice of handshake as explained in Section 3. The protocol will send two messages to the selected contact; one for the protocol initialization and one for the ECDH key exchange. After a successful key exchange; the user will receive a notification that the protocol is set and ready. Now the user can write his message in the textbox and click “Send”. This screen can be seen in Figure A.4.

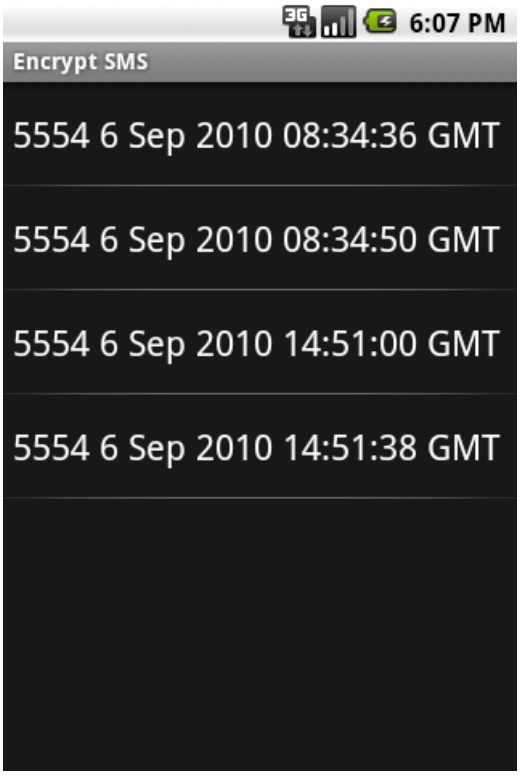

Figure A.5 – Inbox

In the inbox, the user can see his received messages. If the user uses the inbox of the default SMS application bundled in the Android operating system, he will not be able to read his messages, since they are encrypted. Encrypted messages can only be read from the application. The inbox screen can be seen in Figure A.5. When a message from the list is selected, the window in Figure A.6 will appear:

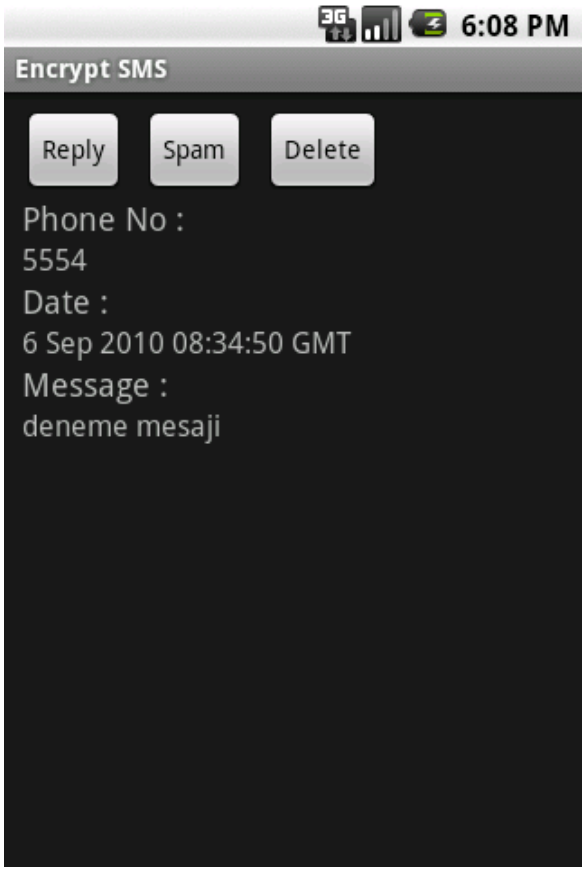

Figure A.6 – Read SMS Screen

The decrypted content of a message can be read in this screen. Here, the user can see sender info, the time the message was received and the contents of the message in plain text. “Delete” button removes the message from the inbox and “Reply” button opens the compose screen, where the user can send an answer to the sender. A simple message is shown in Figure A.6.

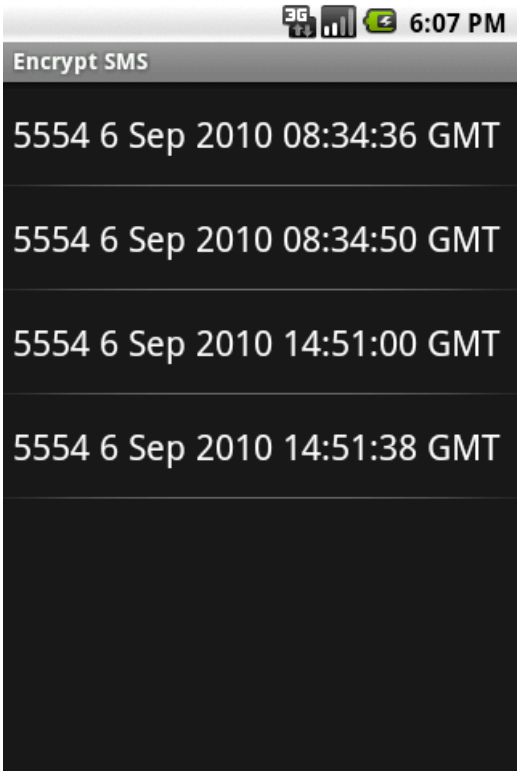

Figure A.7 – Outbox

The outbox button in Figure A.7 has the same design like the inbox. Here, the user can see the messages he has sent. When the user wants to read a message, a similar window like Figure A.6 will appear where he can see the contents of the message.

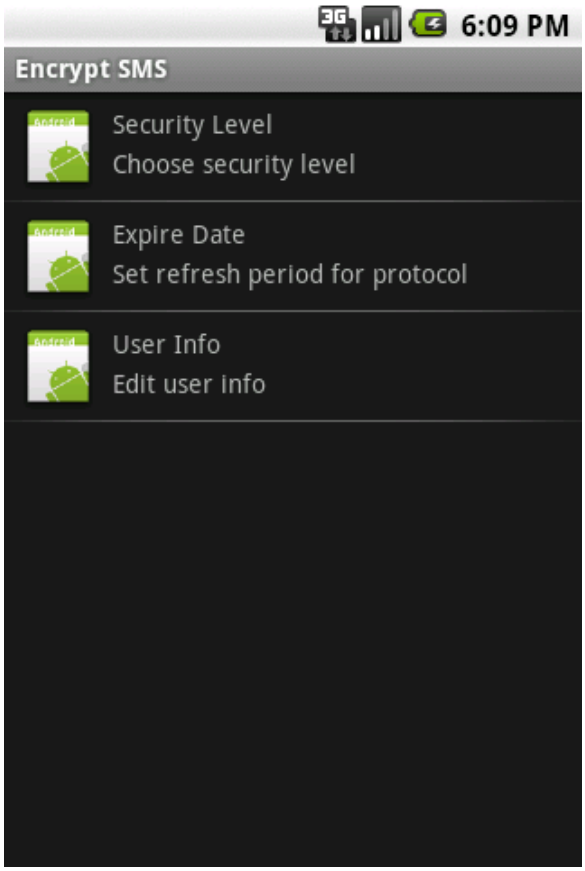

Figure A.8 – Settings Screen

In the settings screen shown in Figure A.8, the user can set following properties that will affect the protocol;

1. The security level
2. Expire date for the protocol
3. User info

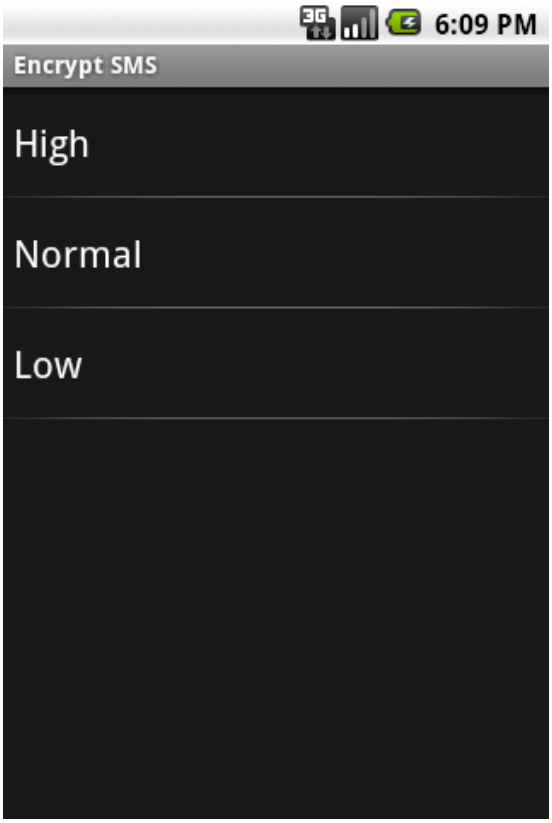

Figure A.9 – Security Level Screen

The security level affects the length of the hash chain with respect to device's hardware. Higher security means more frequent restarts in protocol, which will result in sending more messages.

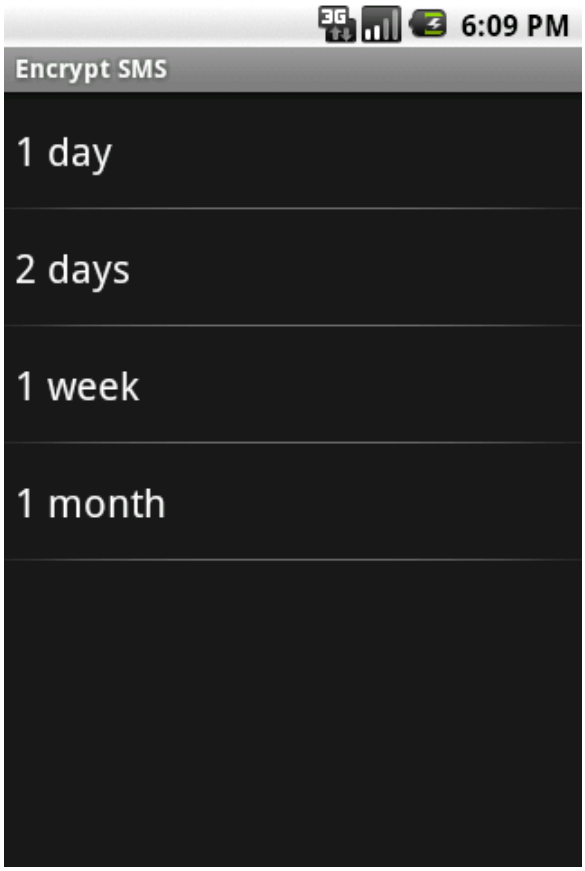

Figure A.10 – Expire Date Settings

In the “Expire Date” screen shown in Figure A.10, the user can set the expiration date for the protocol. Shorter values require more restarts for the protocol, which causes sending more messages.

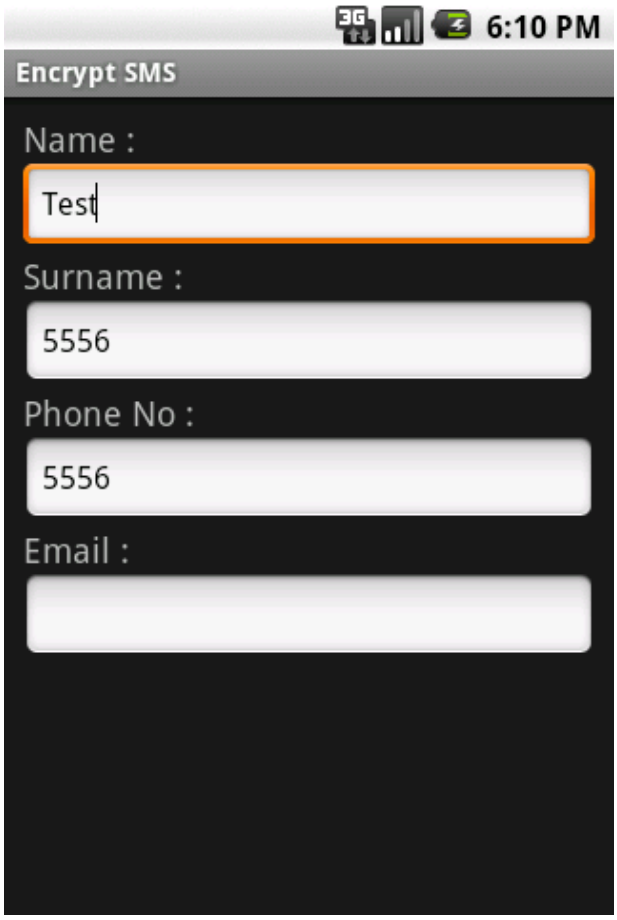

Figure A.11 – User Info Settings

In the “User Info” screen shown in Figure A.11, the user can set his personal information. The phone number is vital, since it cannot be read from SIM card and required by the protocol. The phone number is used in the hash of the key exchange message during protocol initialization.

A2. Voice

The voice encryption implementation is also written in Android platform using version 1.5. When the application starts, the screen in Figure A.12 will appear;

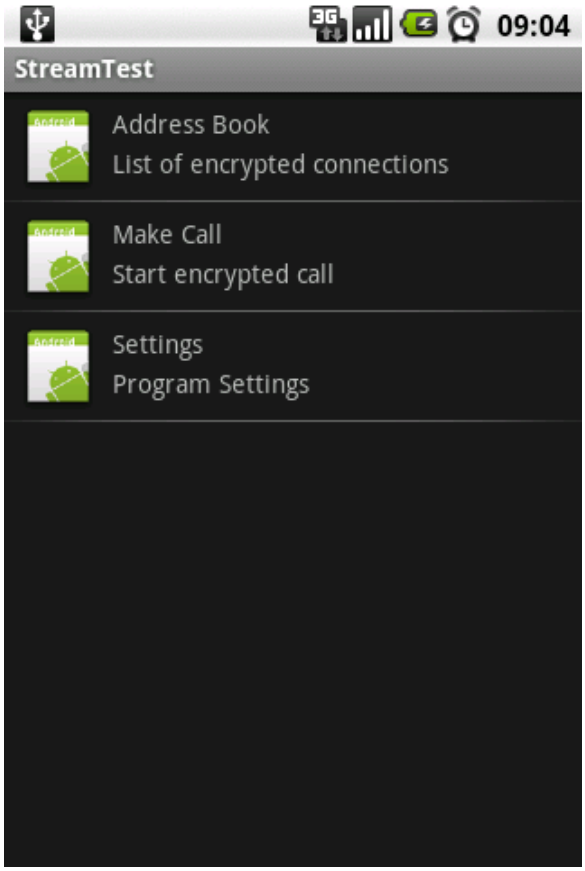

Figure A.12 – Main Screen

In the main menu shown in Figure A.12, the user will have three choices;

1. Address Book: here the user can add / remove contacts
2. Make Call: the user can start communication with his stored contacts
3. Settings: the user can set program specific settings

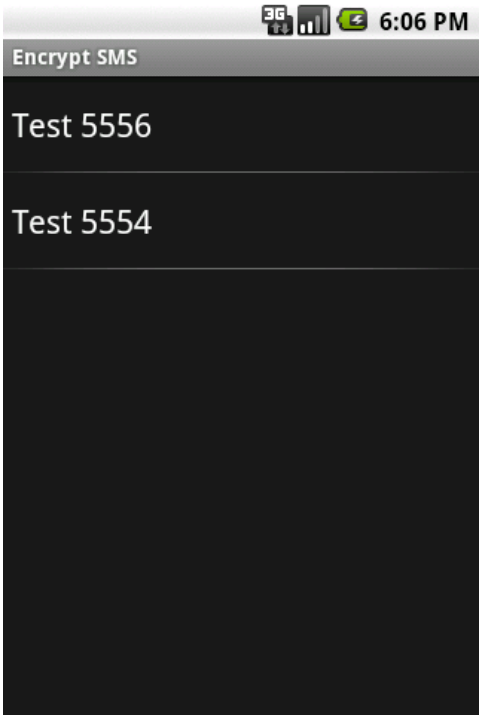

Figure A.13 – Address Book

In the address book the user can see contacts he has created. Here we have two contacts, Test 5556 and Test 5554 as shown in Figure A.13. In order to create a new contact, the user presses the Menu button of your Android device and select “Add new contact” from the menu. The screen will appear in Figure A.14;

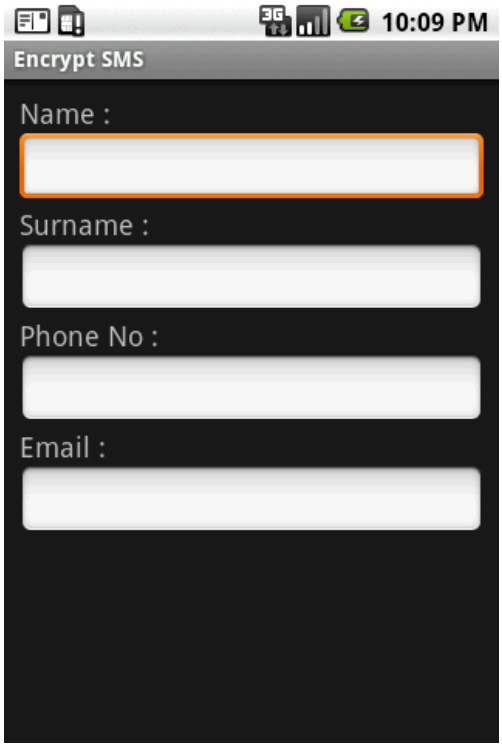

Figure A.14 – Add new user

The user can enter the properties for his new contact in the “Add New User” screen shown in Figure A.14. The user cannot send an encrypted message, unless the contact is in his address book.

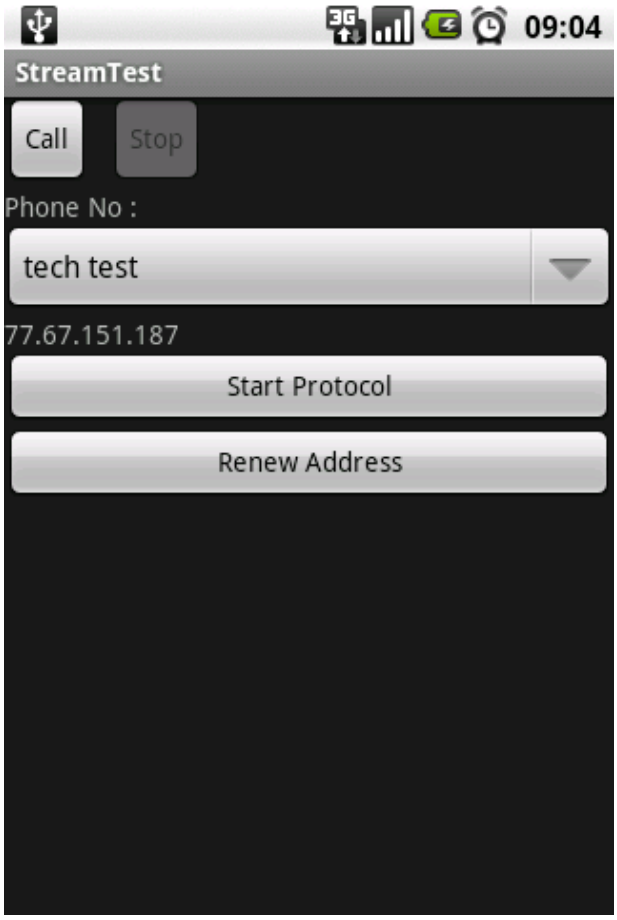

Figure A.15 – Make Call Screen

To start a call, the user should select the contact he wishes to communicate from the list. By pressing the “Start Protocol” button, the user initiates a new key exchange or restarts an existing session. After the protocol is done, he can start his encrypted communication by pressing the “Call” button. If the user wishes to see the current IP address, clicking the “Renew Address” button shows the current IP. To drop a call, the user presses the “Stop” button. The screen is shown in Figure A.15.

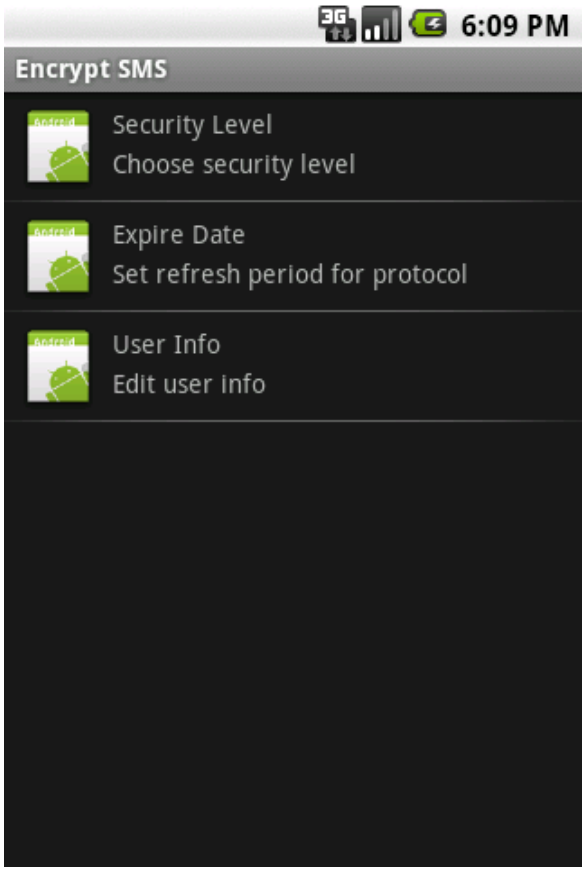

Figure A.16 – Settings Screen

In the settings screen shown in Figure A.16, the user can set following properties that will affect the protocol;

1. The security level
2. Expire date for the protocol
3. User info

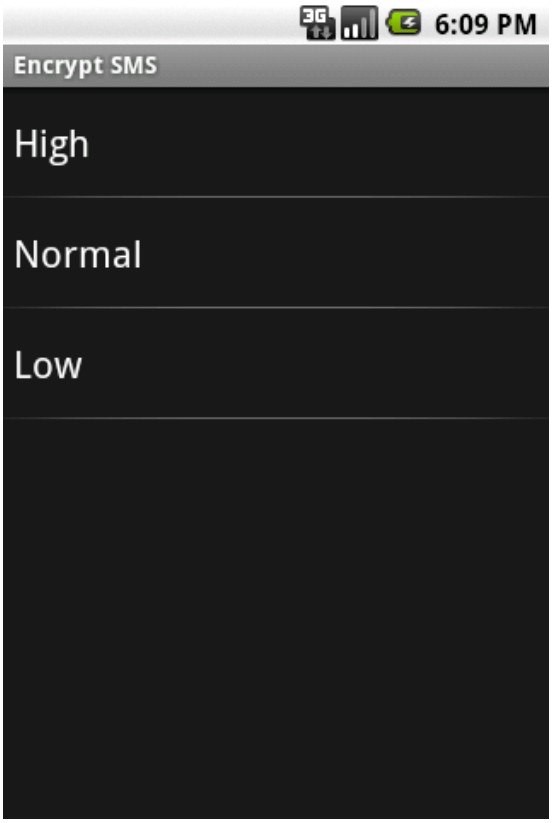

Figure A.17 – Security Level Screen

The security level affects the length of the hash chain with respect to devices hardware. This is a simpler way for the user to modify the variables in the protocol, like expire date and hash chain length.

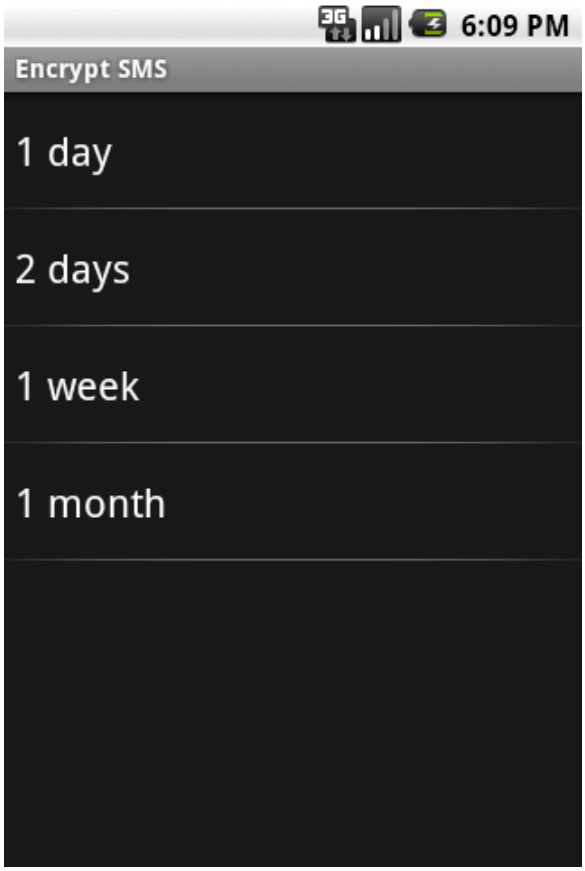

Figure A.18 – Expire Date Settings

In the screen shown in Figure A.18, the user can set the expiration date for the protocol. Shorter values require more restarts for the protocol, which causes sending more messages.

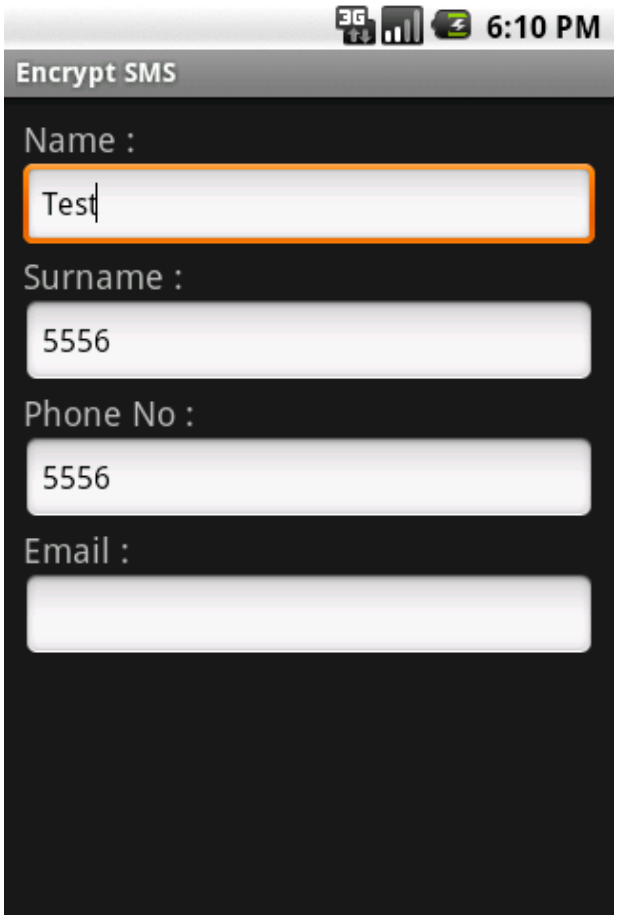

Figure A.19 – User Info Settings

In the screen shown in Figure A.19, the user sets his personal information. The phone number is vital since it cannot be read from SIM card and required by the protocol.