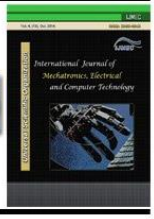
Vol. 8(27), Jan. 2018, PP. 3712-3724

Contents list available at IJMEC

International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)

Journal Homepage: www.aeuso.org

PISSN: 2411-6173 - EISSN: 2305-0543



An Android Mobile Application for an Improved Version of SMSSec, for Secure SMS Communication

Anshu Prakash Murdan and Damien Jérôme Cédric Clain

Department of Electrical and Electronic Engineering, Faculty of Engineering, University of Mauritius, Reduit, Mauritius

Phone Number: *+230 4037851*

****Corresponding Author's E-mail:*** *a.murdan@uom.ac.mu*

Abstract

Short Message Service (SMS) technology is increasingly being used by companies for marketing purposes and for financial transactions, involving the exchange of confidential information. In this paper, the various threats associated with SMS technology such as man-in-the-middle attacks, eavesdropping, replay attacks and SMS spoofing are identified in relation to the above-mentioned services. Proposed protocols to cater for those security issues are then analysed. It is found that most of them rely simply on symmetric and asymmetric encryption which are not sufficient to prevent interception and data modification. Thus, focus is put on the SMSSec protocol proposed by J.Li-Chang Lo et al. This is an end-to-end protocol for SMS, designed in such a way so as to be secure and at the same time computationally inexpensive. In this research work, the limitations of the SMSSec protocol are identified. For instance, the protocol has been designed for a client/server architecture. The handshake involves two messages which is not sufficient for ensuring two-way authentication. To cater for those limitations, a new protocol based on SMSSec is proposed and a mobile application for the Android platform is then implemented based on the proposed protocol. During implementation, the actual protocol and the other parts such as database access are kept separate to allow the protocol to be reused in other applications. A thorough security analysis has proved that using the mobile application to send SMS messages ensures confidentiality, mutual authentication, data integrity and non-repudiation.

Keywords: *Protocol, client/server architecture, handshake, two-way authentication, mobile application*

1. Introduction

Short Message Service (SMS), created during the late 1980s to work with the Global System for Mobile Communications (GSM), allows the exchange of short messages over a mobile cellular network. The exchange is normally carried out between a mobile station and the network which includes base stations and a Short Message Service Center (SMSC), and between the network and another device capable of transmitting and receiving SMS messages. Since the first SMS message that was sent in December 1992, SMS usage has undergone an exponential growth. According to the International Telecommunication Union [1], the total number of SMS sent globally tripled between 2007 and 2010, from an estimated 1.8 trillion to a staggering 6.1 trillion. In other words, close to 200 000 text messages were sent every second. According to figures from Ofcom [2], SMS message traffic peaked in 2011 and has been declining ever since. The average number of SMS and MMS sent per person per month was 138 in 2013 and 117 in 2014 [2]. The rise of smartphones paved the way for consumers to communicate

Article History:

Received Date: Jul. 19, 2017
Accepted Date: Oct. 05, 2017
Available Online: Jan. 04, 2018

through a variety of outlets – from email and instant messaging to Over-the-Top (OTT) content messaging apps. Traditional SMS has been increasingly challenged by alternative messaging services such as Facebook Messenger, WhatsApp and Viber available on smart phones with data connection. However, despite the growth in other communication channels, SMS is still widely in use and remains one of the primary channels of communication, for example in areas with poor or no internet connectivity. This enable users receive messages quickly, making it ideal for sending short, time-sensitive content.

During its infancy, SMS was mainly used for the exchange of short messages between individuals. Its popularity resided in the fact that compared to making phone calls, SMS was cheaper and an ideal form of communication when privacy is concerned. SMS is the most effective way to reach users, with a 90 percent read rate in minutes [3]. When it comes to timeliness of delivery, SMS maintains the highest engagement rate in comparison to emails and OTT apps. Also SMS is available as long as one has a mobile phone and service plan. This makes its global reach soar as there are no pre-existing connections required, such as accepting friend requests or requiring two parties to download the same mobile application. SMS is a one-size-fits-all solution which makes anybody (with a mobile number) reachable. The technology for sending and receiving SMS is not reliant on high speed internet, essentially making anyone in modern society reachable.

Network interoperability was another feature which boosted the popularity of the technology. Over the years, SMS has evolved from a person-to-person communication tool to an information and control resource that has changed society and the way people make business [4]. SMS, due to its simplicity and relatively low cost, appeals to both individuals and businesses. For instance, SMS technology is widely used nowadays by companies for marketing purposes and is increasingly being used to perform financial transactions. Many banks now offer services such as balance inquiry or even money transfer via SMS messages and this gives them a serious edge over their competitors. SMS technology is also widely used in many data centric applications including railways enquiry, news alert, health care applications and to activate/deactivate different kinds of services.

However, as will be discussed in this paper, SMS technology has several vulnerabilities. The implementation of the aforementioned services involve the exchange of confidential information such as passwords, PINs or other sensitive data. Careless use of SMS can give rise to serious security issues such as frauds, especially when transfer of money is involved. In order to address the above vulnerabilities, this paper proposes a system in the form of a mobile application for the Android platform. The system, based on the SMSSec protocol, makes use of end-to-end encryption to allow the user to securely exchange SMS message with a large number of correspondents. After testing and analysis, it was found that the system ensures confidentiality, mutual authentication, data integrity and non-repudiation.

2. SMS Technology and its vulnerabilities

SMS messaging makes use of a separate signalling channel to transfer its payload which implies that SMS messages can be sent or received simultaneously with voice or data calls. However, due to constraints with the payload length of the signalling protocol, SMS messages content cannot normally exceed 160 seven-bit characters or 140 eight-bit characters depending on the encoding scheme being used. Larger content is generally sent using multiple messages.

The Norwegian engineers who invented SMS wanted a very simple messaging system that worked when users' mobile phones were turned off or out of signal range. A store-and-forward service was thus devised which implies that the message does not go directly to the intended recipient's cell phone but instead is stored in the SMSC (Short Message Service Center) and delivered when the recipient device announces its presence. The figure 1 depicts the different parts involved in the end-to-end delivery of an SMS message.

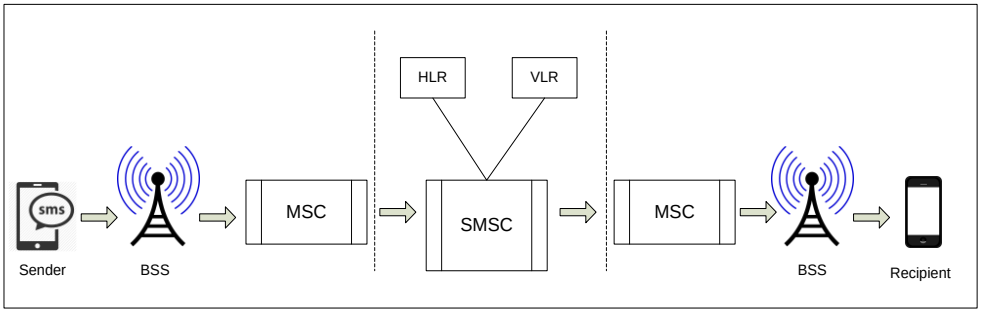


Figure 1: Parties involved in the end-to-end delivery of an SMS message

The BSS (Base Station System) is responsible for the transmission of data between mobile stations. The MSC (Mobile Switching Center) is responsible for the routing of data between networks. The SMSC (Short Message Service Center) is a combination of hardware and software responsible for the storing and forwarding of an SMS message between two devices. The HLR (Home Location Register) contains data relevant to network subscribers such as routing information and the VLR (Visitor Location Register) is used for maintaining the information of roaming subscribers.

The end-to-end delivery of SMS messages can be separated into two main parts [5]. In the first part, the SMS message is carried from the sending device to the BSS over the air interface. The BSS transfers the message to the MSC which then routes it to the appropriate SMSC via the Signaling System No.7 (SS7) channel. In the second part, the SMSC interrogates the HLR to find out the location of the recipient. If the latter is not reachable, the SMSC stores the message. When the HLR finally detects the recipient, it alerts the SMSC which then forwards the message to the serving MSC. The recipient finally receives the SMS message via its BSS.

Prior research [6], [7] has identified several attack scenarios related to 3G cellular networks and has also tried to present a new architecture for 3G security. Among the main threats identified were eavesdropping, masquerading, replay of messages, man-in-the-middle attacks and manipulation of messages. SMS technology is based on a cellular network, and is thus exposed to the same types of threats.

Operating a GSM network required a significant investment. Therefore, when GSM technology was implemented, network-based attacks against end devices were not much of a concern, so phones were not required to authenticate networks to which they were attached. Authentication in GSM is therefore a one-way process only. The network authenticates the user but the user does not authenticate the network. It is therefore possible for an attacker to perform a man-in-the-middle attack by using a 'fake' BSS equipment which has the same network code as the MS (Mobile Station) legitimate network. Provided that the MS is in the range of the 'fake' BSS, such an attack is quite simple to arrange nowadays using open-source software such as OpenBTS [8]. Often, in the case of m-commerce, the server (authentication source) is found in the Internet and does not form part of the GSM network. In such cases, different protocols such as TCP/IP and X.25 are used. The connections between the Internet and cellular network introduces new threats. For example, it is quite simple to do a man-in-the-middle attack against TCP/IP and intercept the communication.

In GSM, voice, data and signalling traffic sent via the air interface (e.g. between MS and BSS) is generally encrypted using either the A5/1 or A5/2 algorithms which are both GSM-specific algorithms. However, serious weaknesses in those ciphers have been identified [9]. They both have been submitted to cryptanalysis and it was found that the A5/1 cipher could be cracked in minutes using an attack called "biased birthday". The A5/2 is even weaker. An attacker can therefore easily intercept a message over the air interface, decrypt it and, read or modify it. On top of that, the choice to encrypt the data depends on the country's policies and approval. In some countries, encryption is not allowed.

Also, the SS7 network supports the transportation and SMS transmission through the core network which makes it an important target for attackers. Although the SS7 channel is already protected, the SMS content may be accessed by the mobile operator employers who can tamper with the integrity of the message.

By default, SMS messages are not encrypted during transmission. As a result, the message is stored as plain text at the SMSC before they are delivered to the intended recipient. It is therefore relatively easy for users in the SMSC to view and amend the message. Spying programs such as FlexiSpy enable intruders to automatically record all incoming and outgoing SMS messages.

One of the most serious threats on mobile operators is SMS spoofing. This occurs when a fraudster manipulates address information in order to impersonate a user [10]. SMS spoofing became even more common when mobile operators started to integrate their network communications with the internet. It is really easy to manipulate the message header and send an SMS from the internet without the recipient knowing that the message actually comes from the internet.

The last threat that will be discussed is replay attacks. A replay attack can be defined as an attack on a security protocol using replay of messages from a different context into the intended (or original and expected) context, thereby fooling the honest participant(s) into thinking they have successfully completed the protocol run. For example, if a user needs to send a password for authentication purposes, the attacker simply has to intercept the message and send it back at a later time. In such a case, even if the password sent is encrypted, the attacker doesn't need to decipher it but only has to send the password again as it is. Because of this, replay attacks are generally simple to perform.

3. Related work

Several protocols have been developed to address the security issues faced by SMS technology. Hassan Mathkour et al. [11] proposed a mobile based application named Secret Short Message Service (SSMS) which used symmetric key encryption and where the payload is encrypted before the message is sent and decrypted at the receiver's end. While this solution is efficient and cost effective, the major roadblock is to find a secure channel to distribute the secret keys. If this cannot be done properly, the security of the system is severely compromised. M.Toorani and A.Beheshti [12] proposed an application layer protocol to secure m-payment systems.

The protocol makes use of elliptic curve-based public key solution that uses public keys for the secret key establishment of a symmetric encryption and the public and private keys are directly stored on the SIM cards. A.Medani et al. [13] concluded that public-key infrastructure (PKI) provides high level security to protect SMS during transmission because it solves and avoids most of the issues related to SMS security. However, it was also recognized that the PKI process requires high mobile power capability and therefore decreases the mobile performance. J.Li-Chang Lo et al. [14] proposed an end-to-end protocol named SMSSec for securing SMS. The protocol uses both symmetric and asymmetric encryption and also includes the use of message authentication code (MAC) and hash functions. It was designed in such a way so as to be secure and at the same time computationally inexpensive. In SMSSec, asymmetric encryption is used during the handshake to ensure mutual authentication and to exchange secret keys. Symmetric encryption is then used to encrypt subsequent messages before sending. A solid security and efficiency analysis proved the protocol to be secure, reliable and efficient.

4. Design of proposed system

We have identified the vulnerabilities of the SMSSec protocol proposed by J.Li-Chang Lo et al., and we propose an amended version of the same protocol. It is to be noted that the vulnerabilities identified previously can be addressed at two different levels – the network level or the application level. The proposed system will work at the application level since it is under the developer's control.

This approach will relieve the network from having to implement the encryption mechanism and since there are different network operators, it will also ensure standardization.

Due to the nature of mobile devices such as cellular phones and PDA's, the application of cryptography at a software level can slow down the execution time if not implemented properly. Aside from the performance aspects, wrong implementation of cryptography can compromise the security of the system [15]. The challenge therefore resides in striving for the best combination between security and efficiency. Our aim is to come up with a system in the form of a mobile application which will allow a user to exchange secure SMS messages with a wide range of correspondents.

4.1 Protocol design

The SMSSec protocol proposed by J.Li-Chang Lo et al. has been designed for a client/server architecture where the client is required to enter a password each time it needs to initiate a session with the server. Thus, using such a protocol for a wide range of correspondents is not very practical since the user has to remember the password for each correspondent. In order to tackle this issue, the proposed system keeps a database of registered correspondents together with the password for each correspondent. It is imperative therefore in this case to ensure the security of the database as this can severely impact on the security of the whole system. Also, the handshake in SMSSec makes use of two messages. In the first message, the client basically encrypts an identification password and a random challenge and sends them to the server. The server decrypts the message and uses the password to authenticate the client. It then responds to the challenge via the second message. On receiving the second message, the client uses the response to authenticate the server and thus two-way authentication is achieved. However, in this system, both parties should be able to initiate a session at any time. Using the SMSSec handshake is therefore not sufficient for ensuring two-way authentication. To solve this problem, our proposed system uses three messages instead of two where challenges are sent by both parties. It is also to be noted that in SMSSec, all messages are sent using symmetric encryption, after the handshake. The secret key for the encryption is shared during the handshake itself. Our proposed system uses the same approach. Then figure 2 illustrates the various steps involved in the handshake for the proposed system.

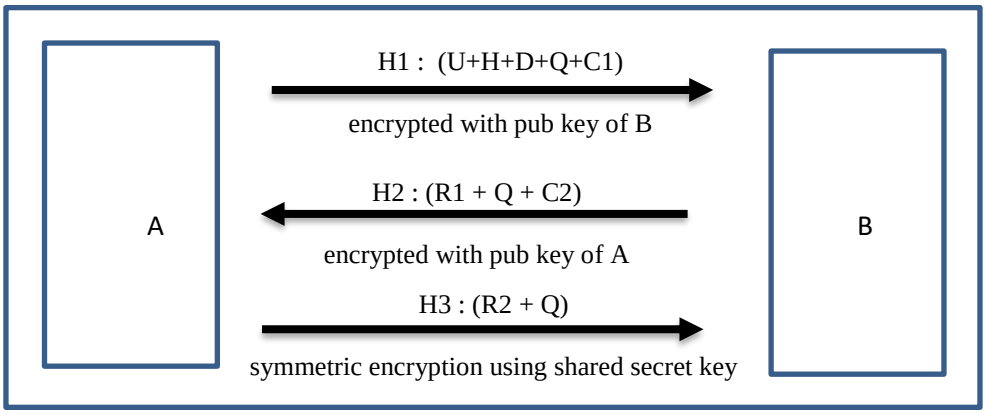


Figure 2: Steps involved in the handshake for the proposed system

4.1.1 H1 -1st message

The first message (H1) is sent by the device initiating the session. H1 is composed of U, H, D, Q and C1. The table 1 details the components of the first message, H1.

Table 1: Details of H1

U	Phone number of the initiating device
H	Constructed by hashing the password corresponding to the intended recipient together with the phone number (U) and a session identifier (Note here that the password is retrieved from the initiating device's database)
D	Generated using the password and a randomly generated salt. D is used to generate the secret key for symmetric encryption. It is important to note that the secret key is never transmitted directly.
Q	The session identifier. The purpose of Q is to prevent replay attacks. When one of the correspondents receives a message, it has to compare a stored value of Q with the value of Q received in the message. If there's a difference between the two, it implies that there has been a replay attack. The initial value of Q is 0 and it should be incremented accordingly when SMS messages are being received and sent.
C1	A random challenge generated by the initiating device. The challenge is basically a randomly generated integer. Its purpose is to authenticate the intended recipient. When the intended receiver receives H1, it decrypts it, and extracts the challenge. The recipient then increments the integer and sends it back. This proves that the intended recipient is authentic since the latter is the only one who is able to decrypt H1.

The initiating device then concatenates U,H,D,Q and C1, encrypts the whole of H1 using the public key of the intended recipient and sends it.

4.1.2 H2 - 2nd message

On receiving H1, the intended recipient decrypts it using its private key. It then retrieves the value of Q for the corresponding phone number of the initiating device from its own database and compares it to the value of Q received in H1. If both are identical, it implies that there has been no replay attack. Next, it retrieves the password corresponding to the phone number of the initiating device from its database and generates H on its side. If the H received in H1 is identical to the generated H, it implies that the device knows the password and is probably authentic. However, there might be the risk that another device, which has also previously registered with the intended recipient, is trying to impersonate the initiating device. Because of this, two-way authentication cannot be ensured by simply relying on H. To cater for this, the intended recipient also sends a challenge to the initiating device. Note that the password is mainly used to generate the secret key for later exchanging messages using symmetric encryption. It is a way of ensuring that only intended recipients can decrypt the message since they will be the only ones who will know the password. The second message (H2) is composed of R1, Q and C2. The table 2 shows details of H2.

Table 2: Details of H2

R1	the response to the challenge C1, which is generated by the intended recipient
Q	the session identifier - it should be incremented accordingly.
C2	the challenge generated by the intended recipient to authenticate the initiating device

The intended recipient then concatenates R1, Q and C2, encrypts the whole of H2 using the public key of the initiating device and sends it.

4.1.3 H3-3rd message

On receiving H2, the initiating device decrypts the message using its private key and verifies Q as done previously to check for replay attacks. It then verifies the intended recipient's response (R1). If it's correct, the initiating device can officially authenticate the intended recipient. Finally, it has to respond to the challenge sent by the intended recipient. The third message therefore consists of R2 and Q. Table 3 shows details of table H3.

Table 3: Details of H3

R2	the response to the intended recipient's challenge
Q	the incremented session identifier

The initiating device then concatenates R2 and Q and encrypts the whole of H3 using symmetric encryption. The key is obtained from the stored value of D and the password. On receiving H3, the intended recipient verifies if the response to the challenge is correct. If it is so, it can officially authenticate the initiating device and the handshake terminates.

4.1.4 Subsequent messages

It is to be noted that H1, H2 and H3 are generated automatically by the application and don't involve any input from the user apart from a confirmation to send the message. After the handshake, both parties can start exchanging messages. Subsequent messages are therefore a concatenation of the message written by the user and the value of Q, all of which is encrypted using symmetric encryption. Since both parties now know the password and have exchanged the value of D, they can both generate the secret key.

4.2 Database structure

Since the aim is to allow the user to exchange secure SMS messages with a wide range of correspondents, a database is required to store information pertaining to each correspondent. The protocol described in the previous section basically needs to store four pieces of information for each correspondent namely its phone number, a password, the value of D and the session identifier (Q). However, the system also needs to be able to keep track of the status of the communication for each correspondent. For example, the device might be waiting for a response from a challenge from correspondent X and on the other side has already authenticated correspondent Y. An additional field is therefore required in the database to keep track of that status. Also, for obvious security reasons, the phone number and password should not be kept as plain text but should be hashed. The salt used in D can be used as the secret value for the hashing function.

4.3 Choice of cryptographic algorithms

For symmetric encryption, two main options are available namely AES and Triple-DES. Although Triple-DES has been studied for many more years than AES, the proposed system will use AES because it always uses longer effective keys and is capable of key sizes up to 256 bits compared to Triple-DES with its 112-bit keys. Since SMS messages are short, they are susceptible to cryptanalytic attempts based on word frequencies. Thus, 256-bit keys will ensure better security. AES is also much faster than Triple-DES.

When public key cryptography is concerned, again there are two main options namely RSA and Elliptic Curve Cryptography (ECC). W.Chou [16] analysed the security and efficiency of ECC in constrained environments and put forward the fact that in a mobile environment, particular care should be taken when choosing public key cryptosystem since bandwidth, memory and battery life are limited. The properties of ECC make it very useful on small mobile devices and smartcards which are limited in terms of their CPU, power and network connectivity. Nonetheless, the proposed system will

make use of RSA with the OAEP padding as suggested in SMSSec for two main reasons. Firstly, if the computing power available on most Android devices nowadays is considered, the difference in performance should not be significant even if RSA uses much larger keys. RSA is thus the best option because of its reliability. Secondly, there are a lot of patent-related uncertainty which exist around ECC. Various aspects of the algorithm have been patented by different companies around the world. In fact, it is one of the main factors which affects its popularity. Finally, the proposed system will make use of HMAC with SHA256 hash function as recommended by J.Li-Chang Lo et al. in SMSSec.

4.4 Choice of platform

Android is the leading mobile platform in the world [17]. While in its early days Android phones were essentially available to consumers shopping within the top price brackets, it is now possible to find and Android phone for nearly any budget. At the time of writing, 97.8 percent of Android devices ran version 2.3.3 (Gingerbread) of Android or above [18]. Thus, choosing to implement our system for Android v2.3.3 and above will ensure that our aim of allowing a user to communicate with a wide range of correspondents is achieved.

5. Implementation

5.1 Choice of cryptographic API

In order to apply the various aspects of mobile security identified previously at the application level, programmatic access is needed to cryptographic algorithms. Many cryptographic packages created for mobile devices exist but they are most of the time a subset of the cryptographic packages for desktops. Mobile cryptography packages have lightweight API methods to keep the size small and as a result, mobile toolkits do not contain many useful methods to support the utilization and initialization of cryptographic algorithms. Mobile cryptographic packages are also more complex to use than their desktop counterparts.

For a mobile cryptography package to provide a safe level of security and efficiency it should meet five major requirements namely speed, size, comprehensive algorithm support, sensible API and easy key identification and serialization. Android's package javax.crypto provides the classes and interfaces for cryptographic applications implementing algorithms for both symmetric and asymmetric encryption and decryption. Stream ciphers and block ciphers are both supported. Authentication mechanisms based on MAC such as HMAC are also available. However, the javax.crypto API does lack some features such as X509 certificate generation. To compensate for this, the Android platform also comes with a customized version of the Bouncy Castle crypto API, which is a free and open-source JCE provider. Unfortunately, the Bouncy Castle version shipped with Android is a cut-down and crippled one which makes installing an updated version of the libraries difficult due to classloader conflicts. To work around this issue, Spongy Castle was developed by making a couple of small changes to Bouncy Castle so that it works on Android.

The proposed system mainly makes use of the javax.crypto API since it supports almost all of the cryptographic algorithms required. Spongy Castle is required only for the retrieving and building of private and public keys for use with RSA.

5.2 Application structure

Figure 4 shows the overall structure of the application, the various classes involved and how those different classes interact.

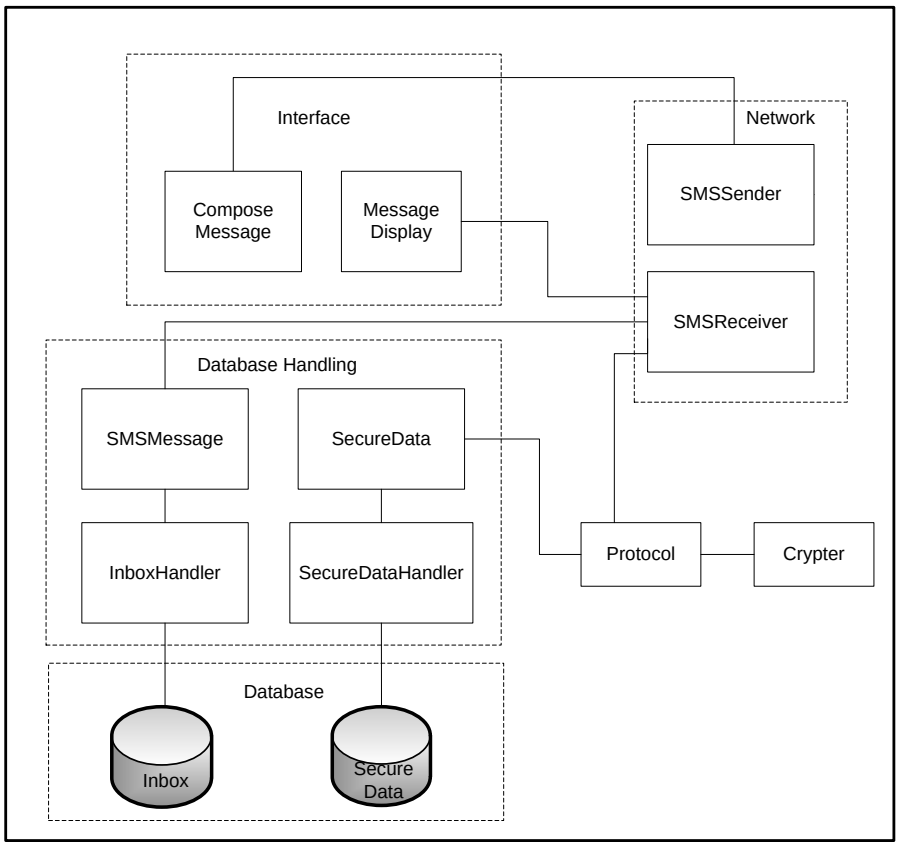


Figure 4: Overall structure of the proposed application

In order to ease the implementation of the application and to allow the protocol to be reused in other applications or on other platforms, particular care has to be taken to keep the actual protocol and the other parts such as database access separately. J.Li-Chang Lo et al. made some suggestions for the implementation of the SMSSec protocol. Our structure is based partly on those suggestions.

A separate class is used to handle the implementation of the processes involved in the protocol such as handshake and exchange of keys. It is however recommended to use a separate class for the implementation of all cryptographic algorithms. Here, the Crypter class is used to perform all cryptographic functions such as hashing, encryption and decryption. Two different classes are used for interacting with the network interface; one for the sending of messages and another one for the receiving of messages. In Android, sending of SMS messages can be implemented using the SMSManager class. In order for an application to be able to receive incoming SMS messages, there should be a class which implements the BroadcastReceiver class.

Security of the persistent storage is a priority because external access to the database values would seriously compromise the security of the whole system. No other applications should be able to access the storage system. The Android platform basically provides full support for SQLite databases. The databases can only be accessed from classes inside the application and cannot be accessed from other applications. This approach therefore ensures the required level of security.

6. Testing and analysis

The application was tested on the following three Android devices:

1. Samsung Galaxy Pocket S5300 (Android v2.3)
2. Samsing I9100 Galaxy S2 (Android v4.0.4)
3. Sony Xperia U (Android v2.3)

Before testing, the databases were pre-populated programmatically to simulate the registering process, that is, each device had in its database the number and password corresponding to the two other devices. During the testing, one device was used as the initiating device and the other was used as the intended recipient. Table 5 shows a few scenarios which have been tested to verify that the application behaves as expected.

Table 5: Scenarios to test application

Scenario	Result
1: A device tries to send an encrypted message without handshake.	The application notifies the user that it first has to perform the handshake.
2: The initiating device encrypts H1 with the recipient's public key and sends it to the intended recipient.	The intended recipient receives H1 and decrypts it successfully.
3: The initiating device encrypts H1 with a recipient's public key and sends it to another recipient.	Garbage values are obtained at the receiving end since the recipient is unable to decrypt H1 as it does not have the correct private key.
4: The initiating device sends H1 with the wrong password.	The receiving device notifies the user that the password authentication failed and terminates the handshake.
5: The intended recipient sends H2 with a wrong value of Q.	The initiating device notifies the user that there is a problem with the intended recipient and terminates the handshake.
6: After handshake has been successfully performed, a device sends an encrypted message to an intended recipient.	The message is successfully recovered at the receiving end.
7: A device tries to send an encrypted message to another device but the receiving device has the wrong value of D.	Garbage values are obtained at the receiving end since the recipient cannot generate the correct key to decrypt the message.

6.1 Security analysis

6.1.1 Attacks during handshake

In the first scenario, it is assumed that an attacker tries to impersonate the initiating device by modifying the header of the message so as to pretend that the message originates from a legitimate device. Also it is assumed that the attacker successfully sends H1. On receiving H1, the recipient decrypts it and sharing of the secret key is complete. However, the recipient then sends H2, which is encrypted with the legitimate device's public key. On receiving H2, the attacker will not be able to decrypt it since it doesn't have the private key of the legitimate device. Consequently, it cannot respond to the challenge sent in H2 and therefore will not be authenticated by the recipient. Two-way authentication will fail and the protocol will terminate. Even if the attacker tries to impersonate the recipient, two-way authentication will fail, since the latter does not possess the private key of the intended recipient.

6.1.2 Man-in-the-middle attack

In the proposed system, messages transmitted are encrypted at the source using symmetric encryption and are decrypted only at the receiving end. After the handshake, only the two

communicating devices know the secret key. Therefore, if a man-in-the-middle attack is performed, the attacker might be able to intercept the message but will not be able to decrypt and read it since he doesn't know the secret key. Regardless of the point during the whole transmission at which the message is intercepted, its content will remain confidential.

6.1.3 Replay attacks

In the following scenario, it is assumed that the proposed system is being used for an m-payment service which allows customers to pay their bills by sending an SMS containing the amount and a specific password. Assuming that the customer has already been registered for the service and has already performed the handshake, an attacker might intercept the encrypted message. Same could be sent at a later time, which would imply that the customer would pay twice. However, in such a case, the intended recipient, which will most probably be a server delivering the service, will be able to detect that it is a replayed message since the value of Q that it will receive in the message will not match with the value of Q stored in its database. The message will thus be disregarded.

6.1.4 SMS Spoofing

With the proposed system, SMS spoofing is also not possible since two-way authentication is performed during the handshake. An attacker might be able to impersonate one of the corresponding devices after the handshake but even in this case, it will be impossible for the latter to actually communicate with the other device. Firstly because the value of Q that the attacker will have to send in the message will not correspond to the value of Q expected by the other device. Secondly, even if the value of Q does correspond, the attacker will not be able to properly encrypt or decrypt messages since he doesn't know the value of D which is needed to generate the secret key.

6.1.5 Android vulnerabilities

However, vulnerabilities have been identified originating from the Android platform. According to the Android's security model, an application cannot have access to another application's data. However, a problem arises when the user gains root access. With root access, an application gains access to the whole system and consequently can also access data from other applications. Rooting therefore poses a serious threat to the security of the proposed system.

6.2 Efficiency analysis

For analysing the performance in terms of speed of execution, the following device was used to obtain a result representative of a larger population.

Model: GT-S5300 (Samsung Galaxy Pocket), Android version: v2.3.6 (Gingerbread), CPU: 832 MHz ARM 11, RAM: 289 MB

The two most computationally expensive functions of the system are symmetric and assymetric encryption and decryption. Table 6 presents the running time for symmetric and assymetric encryption and decryption on the abovementioned device.

Table 6: Running time for symmetric and assymetric encryption and decryption

	Encryption time/ms	Decryption time/ms
AES_CTR (256-bit key)	2ms	5ms
RSAES_OAEP (2048-bit key)	12ms	105ms

The Table 7 presents the average duration of the various steps involved in the handshake i.e. H1, H2 and H3.

Table 7: Average duration of H1,, H2 and H3 in ms

Message	Duration/ms
H1	2015
H2	136
H3	138

The total duration of the handshake is approximately 2.3s which is perfectly acceptable since it is performed only once.

6.2.1 Cost

Using RSA encryption with 2048-bit keys makes the communication safer but has the drawback that the encrypted output exceeds 140 bytes. It therefore implies that two SMS messages will be required to send the encrypted data. This will be the case for H1 and H2 during the handshake. SMS messages encrypted using AES_CTR will have the same length as the plain text since with the CTR mode, no padding is required. Therefore, on the overall, a total of five SMS messages are required to complete the handshake - two for H1, two for H2 and one for H3. The number of SMS messages to send the subsequent messages will of course depend on the length of the plain text message.

7. Recommendations for future work

The presented work shows the implementation of a system, in the form of a mobile application for the Android platform, to secure the transmission of SMS messages between devices. Unlike previously implemented systems, the proposed system ensures not only confidentiality but also mutual authentication, data integrity and non-repudiation. It also provides scalability as it allows a user to communicate securely with a large number of correspondents. However, the overall security of the system heavily relies on the ability of the private and public keys to be distributed securely. Further work is therefore required to find ways in which the use of PKI with the proposed system can be optimized in order to ensure a higher level of security. For instance, a protocol which would work jointly with the proposed system and ensure proper utilization of digital certificates and of a third-party validation authority could be designed.

During security analysis, it has been seen that rooting can be a serious threat to the proposed system. A solution needs to be found in order to tackle this issue. One way to do this would be to store a hashed version of the data in the database. Also, research could be done at the Android API level in order to prevent access to the database from other applications even if the other application has root access. During efficiency analysis, it was found that five SMS messages are required to complete the handshake. SMS compression techniques could be implemented in the protocol to reduces this number and consequently reduce the cost of using the proposed system. Finally, to make the proposed system even more accessible, research into implementing the proposed system on iOS should be carried out.

Conclusion

SMS is an integral part of mobile communication. However, in order to leverage SMS capabilities in a business environment, its vulnerabilities have to be addressed. This paper proposes a system in the form of a mobile application for the Android platform. The system, based on the SMSSec protocol, makes use of end-to-end encryption to allow the user to securely exchange SMS message with a large number of correspondents. After testing and analysis, it was found that the system ensures

confidentiality, mutual authentication, data integrity and non-repudiation. The system has been developed in a modularized way where implementation of the protocol and other parts have been kept separately. In order to make the proposed system even more accessible, the system can also be implemented for the iOS platform.

References

[1] International Telecommunication Union, Market Information and Statistics Division, Telecommunication Development Bureau, 2011

[2] Ofcom, Facts & Figures. http://media.ofcom.org.uk/facts. Accessed 05 Feb 2016.

[3] Future of Communications, Nexmo (2016). Why are people still using SMS in 2015? http://thenextweb.com/future-of-communications. Accessed 20 June 2016

[4] ERICSSON (2012). Twenty years of Short Message Service (SMS): how text messaging helped to change the world. http://www.ericsson.com/news/121130-twenty-years-of-short-message-service_244159017_c

Accessed 07 Aug 2015.

[5] Peersman, G., Griffiths, P., Spear, H., Cvetkovic, S., Smythe, C.(2000). A tutorial overview of the short message service within GSM. Computing & Control Engineering Journal, 11(2),79-89.

[6] Chi-Chun Lo; Yu-Jen Chen(1999). Secure communication mechanisms for GSM networks. IEEE Transactions on Consumer Electronics, 45(4), 1074-1080.

[7] Howard, P.; Walker, M.; Wright, T.(2001). Towards a coherent approach to third generation system security. Second International Conference on 3G Mobile Communication Technologies, 477, 21-27.

[8] Messaging, Malware and Mobile Anti-Abuse Working Group (M3AAWG), 2001. Best Practices to Address Online and Mobile Threats.

[9] E.Barkan et al. (2008). Instant Ciphertex-Only Cryptanalysis of GSM Encrypted Communication. Journal of Cryptology, 21(3), 392-429.

[10] Openmind Networks (2008). The Threat of SMS Spoofing: Prevent Revenue Loss by Securing the Network against Fraudulent Attack.

[11] H.Mathkour et al. (2011). A Secured Cryptographic Messaging System. International Conference on Machine Learning and Computing.

[12] Toorani, M.; Beheshti, A., (2008). SSMS - A secure SMS messaging protocol for the m-payment systems. IEEE Symposium on Computers and Communications.

[13] A.Medani et al., (2011). Review of Mobile Short Message Service Security Issues and Techniques Towards the Solution. Scientific Research and Essays, 6(6), 1147–1165.

[14] J.Li-Chang Lo et al., (2008). SMSSec: And end-to-end protocol for secure SMS. Computers and Security 27, 154-167.

[15] J.Li-Chang Lo and J.Bishop, (2003). Component-based Interchangeable Cryptographic Architecture for Securing Wireless Connectivity in Java Applications.

[16] W. Chou, (2003). Elliptic curve cryptography and its application to mobile devices. Federal Information Processing Standards Publications, Prentice Hall.

[17] Smartphone Statistics and Market Share, IDC Worldwide Mobile Phone Tracker, (2012). http://www.email-marketing-reports.com/wireless-mobile/smartphone-statistics.htm. Accessed 07 Aug 2015.

[18] The Statistics Portal. http://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os. Accessed 20 June 2016.