A Protocol for Secure SMS Communication for Android OS

Smile Markovski[1], Aleksandra Kuzmanovska[1], and Milivoj Simeonovski[2]

[1] Faculty of Computer Science and Engeeniering,
Ss. Cyril and Methodius University Skopje
{smile.markovski, kuzmanovska.aleksandra}@gmail.com
[2] NLB Tutunska Banka
milivojs@gmail.com

Abstract. By its nature, the SMS communication is insecure and the message information can be viewed of many interested parties. Here we propose a protocol for secure SMS end-to-end communication between two mobile devices. Our protocol is based on a symmetric encryption of the message context and on secure key exchange. The security of our protocol is based on the usage of already secure cryptographic algorithms: AES, SHA-2, DH-EC, and others. We have considered several possible attacks on our secure protocol and we could conclude that it is resistant of them.

Keywords: End-to-end Security, Encryption, Mobile Communication, Secure SMS

1 Introduction

Short Messaging Service (SMS) is a communication service, originally developed as part of the Global System for Mobile Communications (GSM). Today it is one of the most widely used mobile services, with million messages exchanged on a daily basis.

Their present-day uses are far different from the initial idea. SMS has now become a popular means of communication by individuals and businesses. Banks worldwide are using SMS to conduct some of their banking services. People sometimes exchange confidential information such as passwords or sensitive data amongst each other. The mobile commerce is everyday growing.

SMS provides many conveniences in our everyday lives, but is it really secure?

SMS messages are sent via a store-and-forward mechanism to a Short Message Service Centre (SMSC), which will attempt to send the message to the recipient and possibly retry if the user is not reachable at a given moment. Transmission of the short messages between SMSC and phone is via the Signalling System Number 7 (SS7) within the unencrypted protocol allowing employees within the cellular providers network, which has access to SS7 network, to eavesdrop or modify SMS messages.

To protect our privacy, as well as to protect our confidential data, we propose a protocol that SMS messages from the source to the destination will be sent encrypted and, accordingly, only the final (end user) recipient can read it. If someone sniff the channel or wants to read the contents of the SMS messages directly from the base, he can only see the text with meaningless incomprehensible characters.

Android is a mobile phone platform developed by the Google-led Open Handset Alliance (OHA). The platform is popular amongst the community for its open source nature and adoption by telecommunication providers world-wide. This OS is based on Linux with middleware presented to developers and core applications. The platform itself focuses on applications, and much of the core phone functionality is implemented as applications in the same fashion used by third-party developers. SMS program is one of the core applications, but it does not support any security for SMS communication. SMS does not use TCP as the transport protocol, so it cannot rely on HTTPS to secure the transmission.

In our protocol, a secure channel is established between two Android devices, using SMS as the transmission medium. To ensure that the SMS remains confidential, a symmetric-based cipher is used to encrypt the message's content. The encryption can be made by any secure symmetric cipher, like AES. We emphasise that our paper is an extension of the ideas given in Hassinen and Markovski's paper [5], where the encryption was defined by using quasigroup transformations of messages. Their solution provides end-to-end confidentiality but they did not propose any method for exchanging the secret key and assumed that users can handle those themselves before communicating. Herein we developed complete secure protocol that work autonomously, including the secret key exchange and allows arbitrary many users. The communication is establishing online, provided that the users have installed the needed components of our protocol.

Nowadays, exists a lot of proposed solutions to secure the sms communication by using public key cryptography such as [6], [8],[9],[7],[3]. All the solutions are based on the server architecture and someone, the mobile operator or service provider, controls the servers. The servers in such architecture are controlling the cryptographic key generations such as key distributions and authenticate the users as well. If the servers are compromised, the whole security system will fail. However, public key cryptography also requires a lot of computational and storage resources to calculate and store public and private keys.

Our solution is based on non-server architecture due to its independency from third party (the mobile phone network operator or service provider). The user does not need to make any aditional agreement with the mobile operator or other third party and all the cryptographic operations are achieved on the users mobile phone.

2 Protocol for secure SMS Communication

We propose a novel protocol for secure SMS communication based on AES (Advanced Encryption Standard)[4] [2] for encryption/decryption and Diffie-

Hellman Elliptic Curve algorithm for key exchange [1][10]. For hashing purposes we use the SHA-2 algorithm.

The protocol is composed of three basic steps:

1. Authentication
2. Key agreement
3. Encryption/Decryption

3 PIN Authentication

By its nature, mobile devices are easy target for thieves. Therefore, authentication with personal identification number(PIN) has very important role in our protocol. Our protocol design includes PIN for authentication and it is required every time when protocol is in use. The PIN is a private key for encryption algorithm and encrypts all the data stored in the application database. If the user enters wrong PIN, application will be opened, but data will be displayed in unreadable format. The PIN has no relation with other elements in the protocol and it is used only for encryption and appropriate decryption of the content data.

The PIN is stored in the phones private memory, that is associated and managed by the application and it is stored for limited amount of time. After expiration, application went to inactive state and the PIN has to be typed again. A user can choose how many minutes the application will be active after the last activity.

The authentication process is independent and it consists of two phases:

1. The first phase is hashing the PIN using SHA-2 algorithm and the resulting string represents the users private key;
2. The second phase is encryption of the database content.

Using this private key and the encryption algorithm AES (or some other), the protocol encrypts all content data.

4 Key Agreement

For establishing secure, i.e. encrypted communication, parties in a communication process must exchange secret keys. Those keys are used by the symmetric algorithm.

The protocol needs the following two types of keys to be used:

- There are two types of Initial Session Keys used for establishing initial secure communication:
  - Initial Session Key for sending SMS messages,
  - Initial Session Key for receiving SMS messages.
- There are two types of Working Session Keys used for sending and receiving secure SMS messages:

- Working Session Key for sending SMS messages,
- Working Session Key for receiving SMS messages

We are introducing two session keys (for sending and receiving messages) to avoid possible collision, in a scenario when both parties sends messages at the same time.

4.1 Initial Session Key exchange

The Initial Session Key is in fact a secret Initial Vector (IV) used for starting the secure communication between two users. From this IV we derive two initial session keys, IV_0 and IV_1. We take $IV_0 = IV$ and IV_1 is the reverse order of IV. The length of IV is at least $n = 16$ characters, in order to be used for AES-128. (It can be set to any value $n \geq 10$, in order to obtain at least 80 bits of security). In the remaining text of the paper, when we mention IV we mean of both IV_0 and IV_1.

$$IV = x_1\ x_2\ x_3\ \ x_n, \text{ n} \geq 10,$$
$$IV_0 = x_1\ x_2\ x_3\ \ x_n, \text{ n} \geq 10,$$
$$IV_1 = x_n\ x_{n-1}\ x_{n-2}\ \ x_1, \text{ n} \geq 10.$$

For Initial Session Key exchange, the DH-EC algorithm is used. Elliptic curve algorithm is chosen because the protocol is primary designed for SMS messages and their limitation is 160 characters, i.e. 1280 bits.

The initial session key IV is defined to be the secret that is shared between two parties. The initiator of the communication sets $IV = IV_0$ as his primary key for sending encrypted messages and IV_1 as a key for decryption of received messages. Using the same IV, the other party in a communication scenario sets IV_0 as a session key for decryption and, appropriately, IV_1 for encryption.

In the proposed protocol, elliptic curve point is defined and it is taken to be a 256 bit number for the X coordinate and a 256 bit number for the Y coordinate. This is unvarying value recommended by NIST and it is used for multiplication with the private key [1]. The private key, i.e. the IV, is defined using random number generator.

The multiplication process and the derivation of the shared key is not explained in this paper, because we use the standard DH-EC algorithm [10]. After establishing the IV, for any two users, the protocol creates pairs of their phone numbers and the secret initial session keys. In such a way a fixed user can send/receive secure messages to/from several users. From one user, the initial session key is used only once for encryption and only once for decryption purposes. After the first application of IV_0 and/or IV_1, it will be changed by the corresponding working session key.

5 Working Session Key

The protocol is working with messages whose length is 160 characters. If a user wants to send a message M larger than 160 bits, it will be separated to several messages $M_1, M_2, \ldots, M_n$ ($M = M_1||M_2||\ldots||M_n$), each one of length ≤ 160 characters, and for every message M_i a corresponding working key will be generated. The original SMS message M will be padded to a message M_p of 160 characters, in the case when its length is shorter than that. The padding is done so that the original message with length of $l < 160$ characters is concatenated with the character of the ASCII code 199, and after that with $159 - l$ randomly generated characters. Hence, the communication between two users A and B will be by SMS of length 160 characters. When the user B will get an encrypted message from the user A, the user B will not see the random characters after decryption, because the application will not display them. We choose the ASCII code 199 because it cannot be typed from the mobile phone keypad, so the user A cannot type it by accident.

5.1 Establishment of new session key

After each successfully exchanged message M', the new session key, called a working session key, will be generated from M' (Figure 1). Let's suppose that user A sends the message M to the user B. The user A already has its working session key $W_{AB,0}$ for sending encrypted messages to the user B and $W_{AB,1}$ for decrypting the messages obtained from B. $W_{AB,1}$ is the reverse order of $W_{AB,0}$. The user B also has its working session key $W_{BA,0}$ ($=W_{AB,1}$) for sending encrypted message to the user A and $W_{BA,1}(= W_{AB,0}$) for decrypting the message obtained from A.

After sending the encrypted message M', new session key will be produced for the user A as follows. By using the hash function SHA-2, a 256 bits hash value $h(M')$ will be produced. Then, the last 128 bits of $h(M')$ will define the new session key $W'_{AB,0}$ for encryption purposes of the user A, and the reverse order, $W'_{AB,1}$, of $W'_{AB,0}$ will be used for decryption purposes. (Here we suppose that AES-128 will be used, so we take 128 bits from the hash value.) The user B will generate the key for the new session after decryption of the obtained message, i.e. after getting the message M'. He will produce a 256 bits hash value $h(M')$ and the last 128 bits will define the new key $W'_{BA,1}$ for decryption of the next message obtained from A, while the reverse order, $W'_{BA,0}$, of $W'_{BA,1}$ will define the new session key for encryption of the next message to A.

After completion of this phase, both users have a new temporary working session key. This working session key will be used for encryption/decryption purposes of the next message only. In such a way, the working session key will be changed when the next message will be exchanged between them.

6 Attacks on Our Protocol

Our crypto system for secure SMS messages contains three types of secret keys: An IV, working session key and a PIN. The IV is used only ones, the PIN is permanent, while the working session key is temporary and it is changed after each exchanged message.

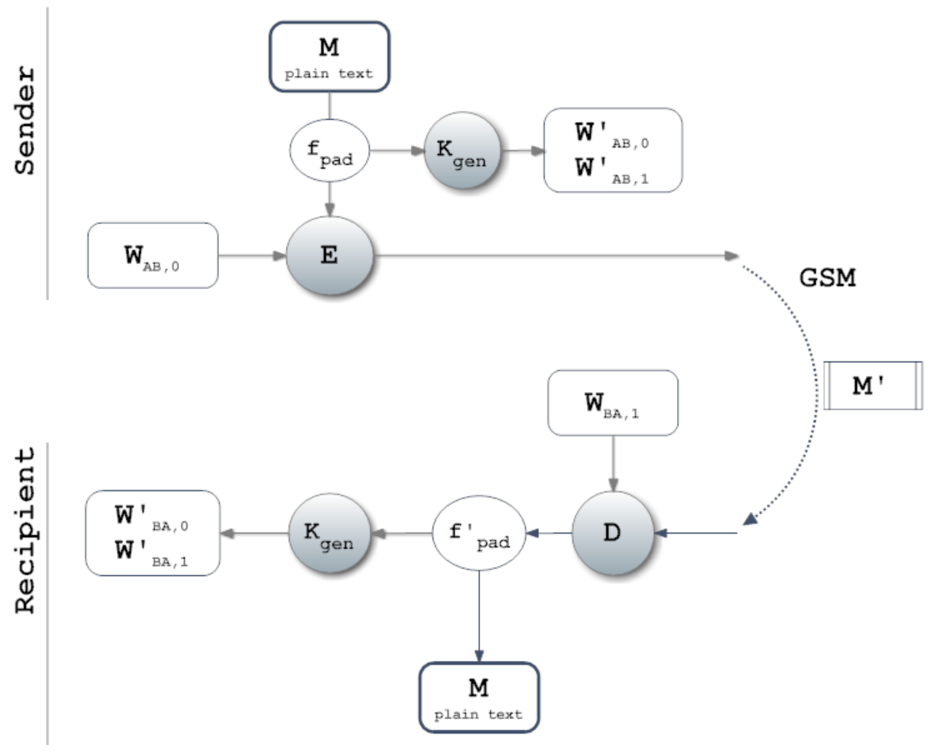

Fig. 1. Secure transmission and key generation

The initial session key, IV, is most vulnerable and special attention for its security is given. It should be exchanged between two parties by public channel and that is why the Diffie-Helman with Elliptic Curve Algorithm is used. Hence, the security of IV exchange is based on the security of DH-ECC. So, it is vulnerable of man in the middle attacks. To avoid the man in the middle attacks, a digital signature should be applied. This will restrict the usability of our protocol, but have to be applied in the case when higher security is needed, like bank transactions. The working session keys are generated from encrypted messages that are also sent through insecure public channel. To generate the working session key, the attacker has to know the previously generated working session key, since at first he/she should decrypt the encrypted message. By a recursive procedure, this means that the attacker should know the IV.

For the encryption and the decryption functions we use the standard 128 bits AES, and the security is based on the security of AES. We find that in such a way protection against linear, differential and other types of attacks, is achieved.

All secret data of our protocol are stored in the application database in encrypted form. If an attacker somehow can get the database from the phone memory of a user A, then the security of messages is based on the security of encryption algorithm. That will not affect the security of the other users of our secure protocol.

7 Implementation of the Protocol for Android OS

We have developed an application CyptoSMS that is a fully functional messaging application, and it provides us with high level of security when sending and receiving SMS text messages. CryptoSMS has been developed for the Android platform and it works on all phones and PDA's that have Android version 2.1 and higher. A list of such devices grows constantly with new models appearing. In order to be able to use the application, both the sending and receiving devices must have the application installed. The messages are sent as binary messages and the device make decision whether an incoming message is for CryptoSMS or not.

SMS receiver is part of the application that is working as a background process with task to listen for the incoming messages. The receiver catches all incoming messages and checks if the message is for CryptoSMS application. If this check fails, receiver leave the message to be received by the default phone application, otherwise receiving is interrupted and the message is processed by the application.

In our implementation for android, we use SQLite database to store all of the data. The database content is encrypted by a symmetric algorithm for encryption and SHA-2 hashing algorithm for the PIN authentication.

8 Conclusion

We have proposed a new protocol for secure SMS communications between two parties with dynamical working key generation. It is based on standard secure algorithms and hence it has the needed security. In fact, depending on the needs of the users, there are three possible levels of security, and they depend on the Initial Session Key exchange. For easy establishment of a fast secure communication between two users by public channel, the DH-EC algorithm can be used. For more secure communication by public channel, a digital signature should be applied. Finally, by using a private channel for Initial Session Key exchange, the best security can be obtained.

We proposed above usage of standard cryptographic algorithms, AES, SHA-2, DH-EC, but the protocol can be implemented by using other cryptographic algorithms as well.

References

1. Recommended elliptic curves for federal government use. National Institute of Standards and Technology (1999), available at http://csrc.nist.gov/encryption

2. Specification for the advanced encryption standard (aes). Federal Information Processing Standards Publication 197 (2001), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
3. Albuja, J.P., Carrera, E.V.: Trusted sms communication on mobile devices. Proceedings of the IEEE pp. 165–170 (2002), http://profesores.usfq.edu.ec/vinicioc/papers/wtr09.pdf
4. Daemen, J., Rijmen, V.: The design of Rijndael:AES — the Advanced Encryption Standard. Springer-Verlag (2002)
5. Hassinen, M., Markovski, S.: Secure sms messaging using quasigroup encryption and java sms api. In: SPLST. pp. 187– (2003)
6. Herlin, H.: Method for secure communication in a telecommunications system (2000)
7. Lisonek, D., Drahanský, M.: Sms encryption for mobile communication. In: Proceedings of the 2008 International Conference on Security Technology. pp. 198–201. SECTECH '08, IEEE Computer Society, Washington, DC, USA (2008), http://dx.doi.org/10.1109/SecTech.2008.48
8. Luo, T.: Method for sending a secure message in a telecommunications system (1999)
9. RATSHINANGA, H., LO, J., BISHOP, J.: A security mechanism for secure sms communication. In: Conference of the South African Institute of Computer Scientists and Information Technologists
10. Research, C.: Standards for efficient cryptography, SEC 1: Elliptic curve cryptography (Sep 2000)