

A Security Mechanism for Secure SMS Communication

HULISANI RATSHINANGA, JOHNNY LO AND JUDITH BISHOP
Computer Science Department, University of Pretoria, South Africa
hratshin, jlo, jbishop @cs.up.ac.za

Short Message Service (SMS) has grown in popularity over the years and it has become a common way of communication. SMS is usually used to transport unclassified information, but with the rise of mobile commerce it has become a popular tool for transmitting sensitive information between the business and its clients. By default SMS does not guarantee confidentiality and integrity to the message content. Therefore SMS is not totally secure and reliable. This affects the Wireless Messaging API (WMA) - an optional package for Java 2 Micro Edition that enables SMS messaging on Java-enabled cellular phones. This paper proposes a protocol that can be used to secure a SMS connection between a WMA client and SMS-based server.

Categories and Subject Descriptors: C.2.0 [Computer-Communication Networks]: General – *security and protection*; K.6.5 [Management of Computing and Information Systems]: Security and Protection – *Authentication*.

General Terms: Cryptography, Protocols, Mobile Devices, Short Message Service.

Additional Key Words and Phrases: Global System for Mobile Applications, Wireless Messaging API, Wireless Transport Layer Security.

1. INTRODUCTION

Short Message Service (SMS) can be defined as any text, voice, sound or image message sent over a public communications network, which can be stored in the network or in the recipient's terminal equipment until it is collected by the recipient [Hammonds 2003]. Although SMS was originally meant to notify users of their voicemail messages, it has now become a popular means of communication by individuals and businesses [Peersman et al 2000]. Banks worldwide, including in South Africa, are using SMS to conduct some of their banking services [Brown et al 2003]. For example, clients are able to query their bank balances via SMS.

When sensitive information is exchanged using SMS, it is crucial to protect the content from eavesdroppers. By default, SMS content is sent over the *Global System for Mobile communications* (GSM) network in clear text form, or in a predictable format [Lord 2003]. This allows an attacker with the right equipment to eavesdrop on the information that is being sent. Another problem with SMS is that the *originating address* (OA) field in the SMS header can be forged, thus allowing masquerading and replay attacks. Therefore SMS is not totally secure and cannot always be trusted. For example, there has been at least one case in the UK where SMS information has been abused by the operator employees [Lord 2003].

Due to the popularity of SMS, Java 2 Micro Edition (J2ME) provides an optional package called Wireless Messaging API (WMA) that enables mobile applications to send and receive wireless messages. J2ME is a Java platform that provides a standard environment for develop applications for mobile devices such as mobile phones and PDAs. Unfortunately, WMA 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.

There are two reasons why it is beneficial to implement SMS-based applications in Java. The first is the market potential [Lo and Bishop 2003]. Secondly, Java provides a richer way to present a SMS message. For example, instead of a plain text presentation, an individual can monitor personal shares in the form of a graph constructed from SMS messages. Therefore it is imperative to investigate a way to provide a security mechanism for SMS communication using WMA. At the current time of this writing, the version for WMA is 1.0 and has been implemented on most Java-enabled cellular phones.

In this paper, we propose a protocol to secure SMS communication between a client and a server using the WMA package.

2. BACKGROUND

In this section we present an overview of the network security threats, SMS message structure and software components that influence SMS communication in the WMA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, that the copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than SAICSIT or the ACM must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2004 SAICSIT

2.1 GSM network security issues

SMS messages are sent over the GSM network; however the GSM network does not provide important security services such as mutual authentication, end-to-end security, non-repudiation or user anonymity [Otto and Virtanen 2004]. This threatens the confidentiality, integrity and availability which are the key elements for providing a secure service for any type of communication over any network that uses either wired or wireless mediums.

2.2 SMS message structure

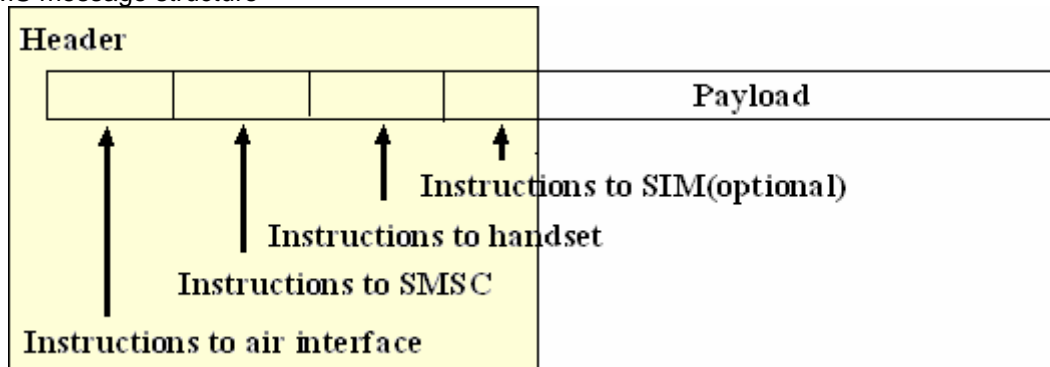


Figure 1: SMS message structure (adopted from [Clements 2003])

From figure 1, SMS is comprised of the header and the payload (which is the content of the message). The header provides four bytes to specify metadata and the size of the payload [Nokia Forum 2003]. The maximum length of the payload is usually 160 characters at 8-bit per character.

2.3 The WMA package

As mentioned in section 1, WMA is an optional package for J2ME that enables an application developer to develop an application that sends and receives an SMS. In figure 2, we illustrate the components within the WMA package.

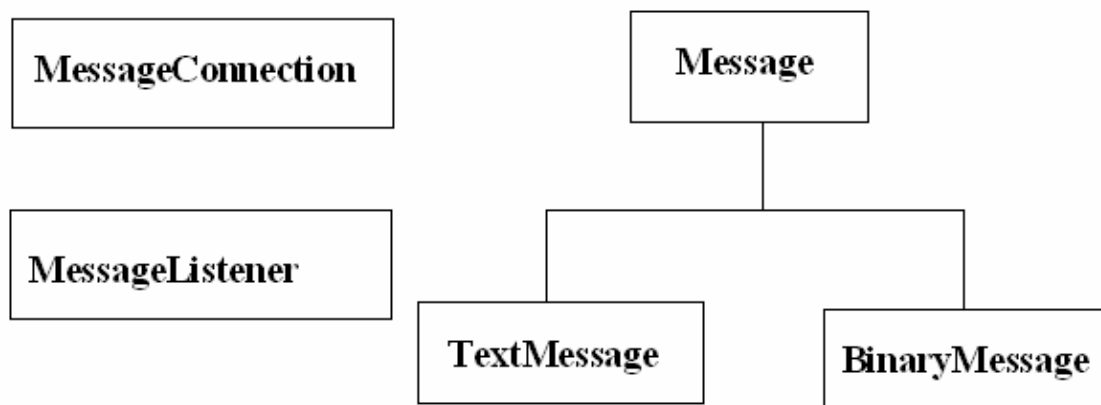


Figure 2: Components of the WMA package (adopted from [Ortiz 2002])

A brief summary of each of these components is given below [Ortiz 2002]:

- **Message:** Is the base for all types of messages communicated using the WMA - a *Message* is what is sent and received, produced and consumed.

- **TextMessage:** sub interface of *Message* that provides methods to set and get a text payload.
- **BinaryMessage:** sub interface of *Message* that provides methods to set and get a binary payload
- **MessageConnection:** sub interface for the *Generic Connection Framework* (GCF) that provides functionalities to create, send, and receive *Message* objects. This interface also defines two String constants, *BINARY_MESSAGE* and *TEXT_MESSAGE*, one of which is used to create the appropriate *Message* type.
- **MessageListener:** Implements the asynchronous notification of *Message* objects.

From our observations, we notice that none of these components include any encryption mechanism to protect the confidentiality content of a SMS message during transit. Also, the WMA 1.0 specification [Sun Microsystems 2002] stipulates that for security purposes, an application must be granted a permission to send and receive message. Unfortunately, granting permissions for these operations does not ensure confidentiality. Therefore, developers must supply their own security information and accompanying mechanism if they wish to secure the content.

In order for the server to be able to communicate with the SMS client, it requires the relevant software support for this particular kind of communication. As of this writing, the Java 2 Standard Edition (J2SE) platform does not support SMS communication, however work is in progress to integrate this feature [Yuan 2002]. If no Java platforms can support SMS on the server side, a third party library is needed for example, *JSMS API* [Yuan 2002].

3. PROPOSED PROTOCOL

3.1 Design requirements

There are two main challenges that we encountered during our protocol design. The first is the limited CPU strength and physical memory size on a small mobile device. In our protocol, we made use of cryptography to provide confidentiality and integrity of the SMS message content. Cryptography can be computationally expensive to perform especially when public key cryptographic functions are used. We do not like to sacrifice security over efficiency so careful judgment needs to be made on the choice and utilization of cryptographic algorithms on small mobile devices.

The second challenge is the SMS message structure and its length. The only part of the SMS message structure we can manipulate is the payload (see figure 1). This means the encryption process and digest could add extra bytes, which correspondingly reduces payload capacity [Nokia Forum 2003].

Based on the security threats defined in section 2.1 and the limitations mentioned in the above two paragraphs, our security mechanism should satisfy the following requirements outlined in [Lam et al 2003]:

- secure,
- easy to implement,
- low computation needs,
- no storage of secret cryptographic keys,
- achieve entity authentication,
- achieve key exchange.

3.2 Design Specification

Our proposed security mechanism is in the form of a protocol that uses public and symmetric key cryptography and password authentication strategy. The following denotations are used for the explanation of the protocol design and they will be referred to throughout the paper:

- S: denotes the server;
- C: denotes the mobile client;
- PK_{pub} : denotes public key of the server that is digitally signed by a certificate authority (CA);
- PK_{pri} : denotes private key of the server;
- R_c : denotes a 64-bit random challenge response generated by C;
- R_s : denotes a 64-bit random challenge response generated by S;
- SK: is the secret symmetric session key shared by S and C;
- SQ: is a 32-bit sequence number (nonce) generated by C starting at 1;
- Slt: is a 128-bit salt value generated by C. This valued is used by C and S to generate SK;
- DT_C : is the confidential SMS message sent from C to S;
- DT_S : is the confidential SMS message sent from S to C;
- MD: denotes the message digest appended to the Req and Rep;
- $E_{PK_{pub}}[X]$: denotes the encryption of data X using the S public key;
- $E_{SK}[X]$: denotes the encryption of data X using SK;

- Username: This is the login name of C that is registered with the S ;
- PIN: Personal Identification Number shared by C and S ;
- \parallel : symbolizes concatenation;
- n : symbolizes the message number.

Our protocol P is divided into two parts, namely the handshake (HS) and transaction (TS) thus forming the equation:

$$P = HS + TS \quad (EQ1)$$

The HS form the key exchange and authentication part of P and TS forms the total transactions done using an encrypted SMS communication. These transactions include the confidential SMS messages that are exchanged between C and S . We can further define HS as:

$$\begin{aligned} \mathbf{M1: C} &\rightarrow \mathbf{S: E_{PK_{pub}}[Username \parallel Slt \parallel SQ \parallel R_c]} \\ \mathbf{M2: S} &\rightarrow \mathbf{C: E_{SK}[R_c \parallel R_s \parallel SQ]} \text{ where } SQ_n > (SQ_{n-1} + 1) \\ \mathbf{M3: C} &\rightarrow \mathbf{S: E_{SK}[R_c \parallel R_s \parallel SQ]} \text{ where } SQ_n > (SQ_{n-1} + 1) \end{aligned}$$

and TS as:

$$\begin{aligned} \mathbf{M4: C} &\rightarrow \mathbf{S: E_{SK}[DT_C \parallel SQ_n]} \text{ where } SQ_n > SQ_{n-1} + 1 \\ \mathbf{M5: S} &\rightarrow \mathbf{C: E_{SK}[DT_S \parallel SQ_n]} \text{ where } SQ_n > SQ_{n-1} + 1 \\ &\cdot \\ &\cdot \\ &\cdot \\ \mathbf{Mn} &\text{ where } n \text{ is the message number} \end{aligned}$$

Thus forming the equations:

$$HS = M1 + M2 \quad (EQ2)$$

$$TS = M_{n-1} \text{ where } n \text{ excludes HS message numbers} \quad (EQ3)$$

In terms of the cryptographic algorithms, we recommend 1024-bit RSA for the public key algorithm due to its popularity in many e-commerce applications, 128-bit AES/CTR for symmetric key and block cipher mode algorithm and SHA-1 for the hash algorithm. The reader must take note that the following description is based on a single protocol run.

M1: For the initial handshake message, C encapsulates and encrypts the following using the public key of S : the *Username*, a salt value *Slt*, a sequence number *SQ* and a random challenge R_c . The purpose of the R_c is to assure the freshness of a protocol run [Lam and Golmann 1992; Lam 1994]. The SK is known to C by hashing the *Slt*, *PIN* and the *Username* to produce a 128-bit key. The encrypted $M1$ is sent in binary format from C to S . After $M1$ is sent, the value of SQ on C is incremented by 1.

M2: Once S receives $M1$, it decrypts $M1$ using the private key. If the decryption is successful, S will first do a check to see if the *Username* is currently used. If it is, it discards $M1$ and terminates the handshake; otherwise, it generates a SK (in the same way as C) and R_s . The received SQ from $M1$ is incremented by 1. S composes $M2$ by encrypting R_c , R_s and the updated SQ . $M2$ is then sent in binary format from S to C . After $M2$ is sent, the value of SQ on S is incremented by 1.

M3: C decrypts $M2$ using the SK . If decryption is successful, C verifies the R_c and compares the SQ that it updated since the last send of $M1$ with the SQ received in $M2$. If SQ in $M2$ has incremented by 1 and R_c matches with the R_c it generated earlier for $M1$, this proves to C that S is authentic. If either of the conditions fails, then C will discard $M2$ and terminate the handshake. If S is proven authentic, C composes $M3$ by encrypting R_c , R_s and an incremented SQ to S using SK and is sent to S in binary format. Once S receives $M3$, it decrypts $M3$ and if it is successful, S verifies the R_s and compares the SQ that it updated since the last send of $M2$. If the SQ in $M3$ has incremented by 1 and R_s matches with the R_s it generated earlier for $M2$, this proves to S that C is authentic. If either of the conditions fails, S will discard $M3$ and terminates the handshake. If C is proven authentic to S , the handshake completes.

M4, M5 and Mn: These messages are the transaction data transmitted between C and S . At this stage, both C and S share a SK between them. Both C and S must ensure the SQ in the current message is greater by 1 from the previous message that was either sent or received. This is to ensure that no replay attacks have occurred during the entire communication using the SK shared between C and S .

A timeout will occur if either C or S does not receive messages for a full HS or TS . The duration of the timeout is still to be determined.

Comparing to other protocols like HTTPS, our protocol is more lightweight and does not add too much overhead to the payload of a SMS message.

4. SECURITY ANALYSIS

4.1 Application of public key cryptography

The two main concerns of public key cryptography is the authenticity of the public key and private key management. Anyone can generate and forge a public key and falsely claim who they are. In order to prevent this, the public key is digitally signed by a certificate authority (CA). All Java applications are deployed on mobile phones in the form of a compressed Java Archive (jar) file. Ideally this jar file must be signed by the originating party and should be downloaded securely onto the mobile phone via HTTPS or Wireless Transport Layer Security (WTLS). We recommend this practice in order to support our protocol since the public key used in P will protected within the jar file along with the other class files. This prevents tampering of the authentic public key by an opponent.

If the server is compromised and the private key is retrieved by an opponent, the whole protocol will fail because the opponent can act as the legitimate party. Therefore an effective key management on S is crucial.

4.2 Security advantages of the protocol

A main advantage to our protocol is that we don't include the PIN or the SK in $M1$. The intention is to not expose the PIN even if public key cryptography is used. The main idea is to get both C and S to generate SK by using the PIN they share between them. An attack on the encrypted $M1$ would not be fruitful if the opponent will not retrieve the PIN . However this might lead to dictionary or brute-force attacks on the PIN . In P , the PIN is protected by a 128-bit Sl value during key generation thus making such attacks more difficult. In general, the PIN should be at least 8 characters consisting of alphabetic and numeric characters. Due to the fact that SK is generated, it is not stored on the mobile phone and in this way SK is not compromised if the mobile phone is stolen.

Each message number is incremented by the SQ to ensure that no messages are replayed. We also excluded the usage of timestamps in P because not all mobile phones will have the correct time coinciding with S .

4.3 Attacking the protocol

In order to attack the protocol, we take on an assumed role that either side could be an opponent. For the first attack, we assume S is the opponent and C is legitimate. S captures $M1$ but cannot decrypt it because it does not have the private key. What S can do is that it can try to fool C by sending a presumed $M2$. When C receives this bogus $M2$ it will discard it and the protocol terminates. Supposedly, opponent S captures a valid $M2$ from a legitimate S and sends it to C . Now C is able to decrypt $M2$, but opponent S is unable to decrypt $M3$ when C replies because opponent S cannot generate a valid SK .

For the second attack, we assume the opponent is C and further assume C has the $Username$. C can generate $M1$ and send it to S and because S possesses the correct private key it can decrypt it and send $M2$ to C . However, C cannot generate the correct SK because it does not know the PIN ; therefore it cannot send a legitimate $M3$ to S . C sends a bogus $M3$ to S which will be discarded by S and the protocol will terminate. Supposedly C captures all three of the HS messages from a previous legitimate protocol run. C can send a valid $M1$ to S and S happily replies to C with a $M2$. Now C can only reply with the $M3$ it captured from the previous legitimate protocol run. However, when S decrypts $M3$, it will find that R_S is different to the R_S generated for this current protocol run; therefore S will terminate the protocol.

5. EFFICIENCY ANALYSIS

In this paper, we present an efficiency analysis in terms of the overhead incurred on the SMS payload when our protocol is applied at each stage of HS and TS . For $M1$, RSA 1024-bit encryption requires its content to be less than 1024 bits. SQ , Sl and RC results in 224-bit in total, this leaves the $Username$ with 800-bits which is sufficient.

Both $M2$ and $M3$ are encrypted with AES/CTR. Due to the fact that CTR does not include padding it therefore excludes this overhead. This means that the data is comprised of R_S , R_C and SQ which are 160-bits in total. For TS messages, the total overhead is from the SQ which amounts to 32-bits. This excludes the DT_C and DT_S , which are the actual payloads.

The downside to this protocol is the cost of the computations performed on C to encrypt $M1$. However due to the security benefits explained in section 4.2 it is justified to shift the computation on C . At S , it is required to generate SK . This might cause a bottleneck on S if it needs to handle over a substantial amount of clients per protocol run. This problem can be alleviated by using hardware accelerators [Lam et al 2003] to compute SK .

6. FUTURE WORK

The protocol description in section 3.2 is done informally. For future work relating to proving the correctness of P we will use formal logic. The actual deployment of this protocol is not yet done in a real world environment. Further work is needed for its implementation and deployment. Not all security protocols are 100% secure, so we cannot guarantee our protocol is perfect either. Therefore further security analysis needs be worked upon.

We hope to broaden our efficiency analysis to include the duration it takes for the HS to complete as well as for some TS messages to be sent and received. In section 3.1, we mentioned there is a limitation of 160 characters per SMS message, if we multiply this by 8, this means we can send 1280-bits per SMS message. Assuming this is correct; we can send all messages within the HS in 3 SMS messages. However this is still need to be verified.

The protocol presented in this paper is designed for a client and server communication. We would like to design a protocol to secure a peer to peer SMS communication.

7. CONCLUSION

In this paper, we have shown that SMS communication is not that secure. Due to the lack of security in WMA, we presented a protocol to ensure the confidentiality and integrity of the SMS communication. The security and efficiency of this protocol is analyzed. The conclusion from the security analysis is positive; however the downside is a heavier requirement in computation for both the server and client. Future work includes tackling proof of correctness, improved efficiency analysis and extension to peer to peer communication.

8. REFERENCES

- BROWN, I., CAJEE, Z., DAVIES, D., AND STROEBEL, S. 2003. Cell phone banking: predictors of adoption in South Africa – an exploratory study. *International Journal of Information Management*, volume 23, issue 5, 381 – 394.
- CLEMENTS, T. 2003. SMS – Short but Sweet. Sun Microsystems: <http://developers.sun.com/techtomics/mobility/midp/articles/sms/>
- HAMMONDS, M. B. 2003. Spam – the meat of the problem. *Computer Law & Security Report* vol 19, issue 5, 388 – 391.
- ITANI, W., AND KAYSSI, A. 2004. J2ME application- layer end –to- security for m-commerce. *Journal of Network and Computer Applications*. Volume 27, issue 1, 13 -32.
- LO, J., AND BISHOP, J. 2003. Component-based Interchangeable Cryptographic Architecture for Securing Wireless Connectivity in Java Applications. *Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists*, Johannesburg, South Africa, 301 -307
- LAM, K.Y., CHUNG, S., GU, M., AND SUN, J. 2003. Lightweight security for mobile commerce transactions. *Computer Communications* 26, 2052 - 2060.
- LAM, K.Y., GOLMANN, D. 1992. Freshness assurance of authentication protocols. *Proceedings of the second European Symposium on Research in Computer Science*, Toulouse, France, LNCS 648.
- LAM, K.Y. 1995. Replay tolerance of authentication protocols. *Journal of computer communications*, volume 18, issue 12, 988-992.
- LORD, S. 2003. Trouble at Telco: When GSM Goes Bad. *Network Security*, issue 1, 10 – 12.
- NOKIA FORUM. 2003. *A Brief Introduction to Secure SMS Messaging in MIDP*. http://www.forum.nokia.com/seap/Developing_MIDP_ChiamPohGuan.pdf
- ORTIZ, E. 2002. *The wireless Messaging API*. Sun Microsystems: <http://developers.sun.com/techtomics/mobility/midp/articles/wma/index.html>
- OTTO, K., AND VIRTANEN, T. 2004. MIDP 2.0 Security Enhancements. *IEEE international conference on System Sciences*. Big Island, Hawaii, January 2004, HI, 287 -294.
- PEERSMAN, C., CVETKOVIC, S., GRIFFITHS, P., AND SPEAR, H. 2000. The Global System for Mobile Communications Short Message Service. *IEEE Personal Communications*. volume 7, issue 3, 15 – 23.
- SUN MICROSYSTEMS. 2002. The WMA Specification. <http://java.sun.com/products/wma/>
- YUAN, M.J. 2002. *Access SMS using the Wireless Messaging API and other packages*. <http://www-06.ibm.com/developerworks/wireless/library/wi-p2pmsg/>