International Journal of Engineering Sciences & Emerging Technologies, Feb 2012.
ISSN: 2231 – 6604 doi: 10.7323/ijeset/v1_i2_1 Volume 1, Issue 2, pp: 1-7 ©IJESET

A REVIEW OF SECURE SMS BASED M-COMMERCE

Ashish Ranjan[1], Rajashekara Murthy S[2], Ramakanth Kumar P[3]

[1]Student & [2]Assistant Professor, Department of CSE,
RV College of Engineering, Bangalore -560059, INDIA
[3]Professor, Department of ISE,
RV College of Engineering, Bangalore -560059, INDIA

ABSTRACT

Mobile devices have revolutionised the world. They are now being used for M-Commerce applications. Their affordability, high availability, and usage convenience has made SMS based M-Commerce to take over e-commerce. Nowadays it is used as an alternative for online transaction systems. The SMS facility available in mobile phones is used for communication between client and server modules. The regular encryption and decryption algorithms cannot be used as mobile devices have limited memory and processing power. Keeping these limitations in mind a new framework is proposed. It involves suitable encryption and decryption algorithms and a frame structure. The proposed framework is implemented for a banking application. The user sends the SMS after encryption to the bank server through his mobile phone. The bank server is programmed to poll the modem for messages. On the arrival of a request, the bank server decrypts and authenticates the message before servicing it. The serviced reply is again secured by encryption before the transaction, details are sent back to the client as another SMS using the modem. The client decrypts the received messages and displays appropriate results to the user.

KEYWORDS

M-Commerce, SMS, Perl, TEA, MD5 authentication

I. INTRODUCTION

M-commerce is an emerging field with many developments taking place and it has taken over e-commerce in many ways today. SMS based M-Commerce comes as an alternative to the online transaction systems that require the user to login using a computer. With M-Commerce the user is allowed to transact at his convenience from anywhere and at anytime.

SMS by itself is an unsecured mode of communication as the message can be obtained and examined at the service centres. The existing M-Commerce systems that use the SMS service for communicating with their clients are unsecured and hence only generic information like special offers, notices etc are sent. To provide security to this system peer to peer encryption of the messages is required. Encryption can be Public key or Private Key [1]. Public key cryptography provides us with the advantage of providing authentication along with security. However when used on a device with various limitations like memory and processing power it suffers from various disadvantages. The encryption procedure is more resource consuming i.e., both in terms of memory and processor cycle requirements [2]. Also the generated cipher text is often too long to fit into the 160 character SMS message owing to the very large keys used to provide security (1024 bits recommended for RSA) [3,14]. Only by using some compression technique like ZIP the size of the message can be reduced which is a computing overhead.

The use of a private key block cipher reduces the resource requirements and also the amount of cipher text produced. From the various available block ciphers TEA [4, 5] is chosen as it is designed to

reduce the memory footprint and maximize speed MD5 is used for authentication as it produces a standard 128 bit output for any given length of input text. Thus reducing the overall length of the request message

The remainder of the paper is organized as follows. In Section 2 - "Challenges and related work", the major challenges involved in the application development for mobile devices are given. In Section 3 "System design", the proposed frame structure for efficient communication between the server and client modules is provided. Section 3 also discusses idem potency problem and proposes a solution. Section 4 - "Security analysis" discusses the algorithms used for providing security and authentication. Section 5 - "Implementation for bank application" explains the implementation of the proposed system for the banking related application [16]. Section 6 - "Conclusion" summarises the observations made with the banking application implementation and concludes the paper by listing the other areas where the proposed system can be useful and possible further enhancements. Section 7 lists the papers and other materials referenced.

II. CHALLENGES AND RELATED WORK

The major challenges faced in the development of the banking application are - choice of suitable encryption algorithm considering the processing power and memory limitations of the mobile device, choice of a suitable authentication procedure, development of appropriate key renewal and distribution procedures to build a design capable of handling idem potency, capability to limit the whole request message to a maximum of 160 characters in 7 – bit encoding format and development of an application that uses the messaging facility of a mobile device independent of the manufacturer's specifications. A detailed opportunities and challenges are listed in GS1_mobile_com [6]. A hybrid compression encryption technique to secure the SMS data is proposed by Tarek M Mahmoud [7]. The computational cost overhead that the security protocols and algorithms impose on mobile phones is analyzed by Anita & Nupur Prakash [8]. Johnny Li-Chang Lo etal, proposed a two-phase protocol with the first handshake using asymmetric cryptography which occurs only once, and a more efficient symmetric nth handshake which is used more dominantly [9].

III. SYSTEM DESIGN

The application is divided as client module developed using Java ME [10,11] and the server developed using PERL [12] as shown in figure 1.

The client application should be capable of

1. Sending SMS messages from the phone
2. Encrypting and decrypting the messages appropriately
3. Authenticating the messages
4. Should have a provision for persistent storage of data in a secure manner

The server application should be capable of the following -

1. Fetching the request from the modem to the computer
2. Capable of running the same decryption and authentication algorithms as the client in similar manner irrespective of the word size of the processors used
3. Replying back to the client request on a specific port so that the client application receives the message instead of the default phone inbox

3.1 MESSAGE FRAME STRUCTURE

A frame structure to the request and reply messages is essential to parse the messages easily. The field that can be used in a request message include

ACCOUNT NUMBER || TID || SID || Enc (TID || PIN || REQ-TYPE || REQ-DETAILS) || MAC-VALUE

The choice of encrypting the account number is left to the implementation. However one must note that by increasing the number of fields in the encryption string the chances of message length crossing

160 characters increases.TID specifies the Transaction ID. This is a unique transaction ID for each new request. It helps us solve idem potency.

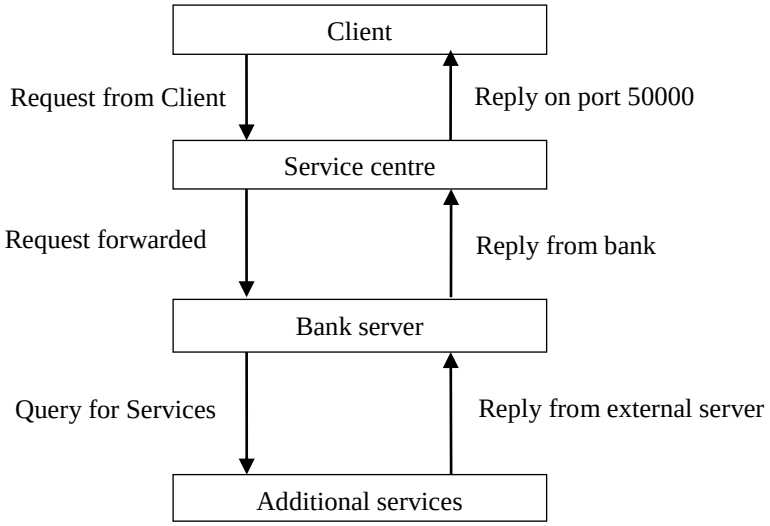

Figure 1. Flow of control between different entities in the application

Considering the fact that more than one bank client should be allowed to use the software on the same phone to transact, instead of maintaining an encryption key for each account it is easier to maintain a key for each installation of the software. Thus SID in the request field specifies the server which key has to be used for decryption of the request message.

The reply message could have the following fields

Enc (RESULT || NEW-TID || NEW-KEY || TRANSC-DETAILS)

The RESULT field is a single character which can take values '0' or '1' based on success or failure of the transaction request. More error codes can be used that can specify the error like invalid account number, insufficient balance, authentication failed etc.

NEW-TID and NEW-KEY are used for the next transaction that the client makes from his phone.

3.2 DESIGNING THE CLIENT SIDE MODULES

A Java ME application is developed to accept the user information. The information is then processed by this application to format the message into a specific request format. The request is then encrypted and authentication value appended. This request is then sent as a SMS request to the server.
Java ME provides us with the SMS messaging APIs in the WMA [13] (Wireless Messaging APIs) set. These APIs provide us with the option of sending and receiving SMS messages on specific port number. The client's push registry stores the information about these messages. Every time the phone receives a message on this particular port, the corresponding application is run by the phone. Java ME also provides us with a set of APIs to build useful user interfaces. It also provides us with the RMS (Record Management system) APIs to store data persistently on a mobile device.
The records that are created using RMS, are used to store NEW-KEY, NEW-TID, and any transaction history that the client would want to save. RMS provides us with the setAuthMode () facility using which the access to the records can be restricted to the midlet that we develop.

Apart from the above said benefits Java ME makes the application portable on to any phone with Java runtime. Also the word size used is independent of the processor word size. Java ME provides us with a good level of abstraction with its rich set of APIs making it easy for the programmer to build this application.

Figure 2. Screen shot when the client side application is run on the emulator

However due to the limitations in the use of APIs (restricted Java API set) the programming of the encryption and authentication algorithms proves to be a challenge.

3.3 Designing The Server Side Modules

The server side is implemented using PERL and MYSQL. For receiving the SMS on the server side a standard mobile phone modem is used, which is connected via USB to the server. For accessing the messages on the phone and also for sending the messages, the server makes use of the modem AT commands. The Perl extensions Device: Modem and Device::GSM are used to send AT commands to the phone.

The messages retrieved using the AT commands are validated and all the new requests are added to the database. Before adding the request the decryption of the request is carried out using the client's key, also the MD5 value is calculated for the message and stored to the database, which is further used to authenticate the request message. The added request is then serviced and the reply is generated which contains the details of the transactions, the new key and also the new transaction ID Message format

RESULT || NEW TID || BALANCE || NEW KEY

This message is encrypted using the TEA encryption. Once the message is encrypted, it is to be sent to the client application on a particular port. This is achieved by sending the PDU SMS instead of a text SMS. The message is formatted in the form that it can be sent as a PDU and also the UDH (User Data Header) is added to it. This message is finally sent to the client using the modem connected to server with the AT commands and Device: Modem extension.

3.4 The Additional Services

Any additional services like the purchase of e-tickets; mobile recharge etc. can be implemented with this system.

The server can check for the type of the additional service requested by the client and accordingly make a HTTP request to the server providing that service. The reply from the additional server can be appended to the normal reply, then encrypted and sent to the client using the same procedure used for

normal transactions.

3.5 THE KEY DISTRIBUTION PROCEDURE

Since we are using the private key encryption methods renewal of a key is a problem. Maintaining a key distribution centre is an expensive and impractical option.
We can go for a random string generation function that generates the same new key when run on both the client and the server with a specific input. However synchronization between these runs on the client and the server side is a major problem considering the fact that the request or the reply messages can be lost due to the unreliability in the SMS messaging system.
We propose a synchronization mechanism in the following section using which keys can be distributed.

3.6 THE IDEM POTENCY PROBLEM

To solve the problem of servicing repeated requests due to failure in loss of request or reply messages a transaction ID (TID) is used. Initially when the software is installed on a phone a default value is given. On every successful transaction the TID is updated by server sending the newly calculated TID in the reply message.
In case the request is lost, the client re-requests with the same TID. If the reply message is lost the client again request with the same TID. Now the server identifies that the client has made a request with the old TID as the acknowledgement from the server has not received him. The server simply replies back to the request without making any modifications to the database. The TID can also be used to foil any replay attacks as the TID is used in the encryption string while generating request. Along with the NEW-TID a NEW-KEY is also sent to the client. This is equivalent to renewing the client's key in a synchronous manner.

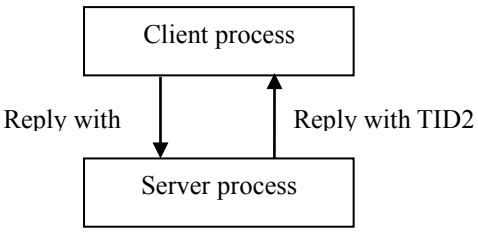

Figure 3. TID updating in a successful transaction

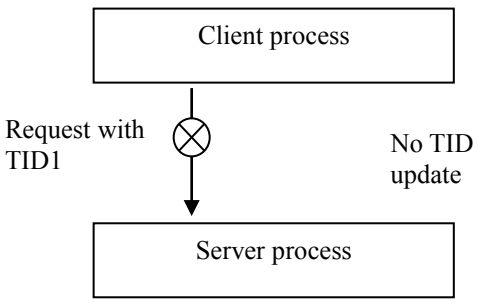

Figure 4. TID updating in case of request failure

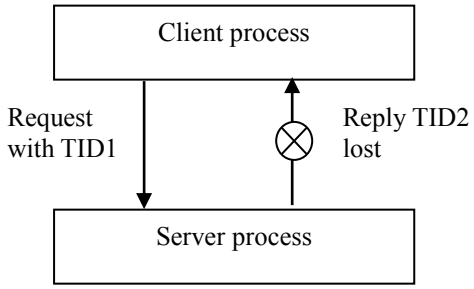

Figure 5. TID updating in case of reply failure

As we note the transaction IDs are maintained and distributed by the server.

IV. SECURITY ANALYSIS

The algorithms used to secure and authenticate the application are TEA – Tiny Encryption Algorithm and MD5 – Message Digest 5 [15]

4.1 TEA

TEA operates on 64-bit blocks and uses a 128-bit key. It has a Feistel structure with a suggested 64 rounds, typically implemented in pairs termed cycles. It has an extremely simple key schedule, mixing all of the key material in exactly the same way for each cycle [4].
TEA is highly resistive to the cipher text only, known plain text and chosen plain text attacks. Plain texts with one bit difference do not find any similarities in the cipher texts. TEA produced approximately 32 bit differences in the cipher text for one bit difference in the plain text just after six rounds.
The basic encoded routine in 'C' is:

```
while (n-->0) {
    sum += delta;
    y += (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
    z += (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
}
```

where: n = number of rounds,
delta = a constant to make sure that the sub keys are different,
y = first 32 bits of the message,
z = remaining 32 bits of the message,
k[0-3] = 32 bit key(total 128 bits key).

The basic decode routine is:

```
while (n-->0) {
     z - = (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;
     y -= (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
     sum -= delta ;
}
```

However, it is susceptible to two attacks, which are:
Related key attack and Equivalent key attack. To overcome these attacks enhanced versions of the TEA such as XTEA and XXTEA are designed.
Other popular symmetric key ciphers are Triple-DES, AES and Blowfish. TEA is chosen as it is designed to reduce the memory footprint and maximize speed.

4.2 MD5

MD5 is a hashing algorithm mainly used for the purpose of authentication. It takes as input a message of arbitrary length and produces as output a 128-bit message digest. The input is processed in 512-bit blocks.

The MD5 algorithm has the property that every bit of the hash code is a function of every bit in the input. The complex repetition of the basic functions(F,G,H,I)produces results that are well mixed, that it is unlikely that two messages chosen at random having similarly regularities, will have the same hash code. In the implementation of MD5, with the input we include the pin number of the client so the hash code generated is particular to the client.

The advantage of MD5 over other hashing algorithm is that the output is 128-bit and other algorithms like SHA-1generate 160-bit hash code. This is a greater advantage because the SMS length is 160 characters. Other advantage is that the execution speed is faster because of 128-bit buffers rather than the 160-bit buffers.

The equation that produces hash code is:

CV(0)=IV

CV(q+1)=SUM32[CVq, RFI (Yq, RFH (Yq, RFG (Yq , RFF (Yq , CVq))))]

MD5=CV(L-1)

where:

IV = intial value of the 160 bit buffer.
Yq = the qth 512-bit block of the message.
L = the number of blocks in the message.
CVq = chaining variable processed with the q^{th} block of the message.
RFx = round function using primitive logical function x.
MD = final message digest value.
SUM32 = Addition modulo 2^32 performed.

V. IMPLEMENTATION FOR BANK APPLICATION

Client side was programmed in J2ME with CLDC 1.1 and MIDP 2.0 configuration. JAR file generated after building the application was used to install the application on various mobile phones of different make (Nokia, Motorola and SonyErricsion) to test the functionality. The application functioned as expected by encrypting the user request, authenticating it and sending it as a text message to the respective server modem.

The server module was coded in Perl. The phone connected for receiving messages was Motorola (Moto ROKR E6) and phone used to send replies was Nokia 5310. The server module was able to fetch inbox and decrypt the messages appropriately. The same Java class files used for encryption and authentication on the client's phone were used by the server module. The server was able to reply back to the client's request on the expected port by sending the reply as a binary message.

VI. CONCLUSION

A new framework that works efficiently under the limitations of mobile phones is proposed and implemented for a bank application. The proposed frame work overcomes the major drawbacks observed in the existing M-commerce systems while providing a cost effective and secure way of banking with absolute user convenience. The frame structure of this application can be extended to provide various other facilities like a college server using which students can know their marks and other information on their cell phones or a search server that replies to various client queries.

Some further enhancements that we propose for this application are - to use public key cryptography for the exchange of session keys and DSS standard for the purpose of authentication, use of SMS service centres for handling more requests reliably, building an application that overcomes the limitation of requiring java runtime on the client’s mobile

REFERENCES

[1] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, "Handbook of Applied Cryptography", CRC Press
[2] G. Le Bodic, "Mobile Messaging Technologies and Services SMS, EMS and MMS", 2nd Ed. John Wiley & Sons Ltd, (2005).
[3] John Timmer, "768-bit RSA cracked, 1024-bit safe" in Digital Security Essentials : arstechnica.com on January 7, 2010
[4] Vikram Reddy Andem, "A Cryptanalysis Of The Tiny Encryption Algorithm", Master's thesis, The University of Alabama, Tuscaloosa, 2003
[5] Wheeler D. J., Needham R. M., "TEA, a tiny encryption algorithm", Lecture Notes in Computer Science - Fast Software Encryption: Second International Workshop (1994), pp. 363–366.
[6] "Mobile Commerce: opportunities and challenges - A GS1 Mobile Com" White Paper February 2008 Edition
[7] Tarek M Mahmoud, Bahgat A. Abdel-latef, Awny A. Ahmed & Ahmed M Mahfouz, "Hybrid Compression Encryption Technique for Securing SMS", International Journal of Computer Science and Security (IJCSS), Volume (3): Issue (6) P473-P481
[8] Anita & Nupur Prakash, "Performance Analysis of Mobile Security Protocols: Encryption and Authentication", International Journal of Security, Volume (1) : Issue (1), June 2007
[9] J. Li-Chang Lo, J. Bishop and J. Eloff. "SMSSec: an end-to-end protocol for secure SMS", Computers & Security, 27(5-6):154-167, 2007
[10] James Keogh, James Edward Keogh, "J2ME: The Complete Reference", McGraw Hill/Osborne, 2003.
[11] John W. Muchow, "Core J2ME Technology & MIDP", Prentice Hall PTR, 2002.
[12] Randal L Schwartz, "Learning Perl", O'Reilly & Associates, Inc, 1997.
[13] C. Enrique Ortiz, "The Wireless Messaging API", Article, December 2002
[14] David Pointcheval, RSA Laboratories' CryptoBytes, "How to Encrypt Properly with RSA", Volume 5, No.1, Winter/Spring 2002, pp. 9-19.
[15] J. Touch, "Report on MD5 Performance", RFC 1810, June 1995.
[16] Anantha Murthy H S, Rajashekara Murthy S, Sheela Ganesh Thorenoor, "Communication enabled Business Process for Banking Industry", Vol. 2, Proceedings of National Conference on Recent trends in Computer Technology (RTCT 2011), April 2011

Authors

Ashish Ranjan is currently pursuing his bachelor degree in computer science and engineering. He is currently in 7th semester (4th year). He has developed a pedagogical tool for binary tree properties that is being used by teachers in teaching the subject 'Data structures' to undergraduate students.

Rajashekara Murthy S obtained his Master's Degree in Computer Engineering from VTU and Bachelor's degree in Computer Science & Engineering from Bangalore University. His research interests are Natural Language Processing and Algorithms. He has guided more than 15 undergraduate projects and 5 post graduate projects.

Ramakanth Kumar, P was awarded Doctorate from Mangalore University, has teaching experience of around 14 years in academics and Industry. His area of research is on Artificial Intelligence, Pattern recognition. He has to his credits 03 National Journals, 02 International Journals, 12 Conferences and 15 Research Publications. He is guiding 14 M.Tech students and 03 PhD students.