Mobile Payment Gateway

Amol Kulkarni,Sachin Kulkarni,Amey Kulkarni,Ashish Ghirnikar and Prof.S.G.Pukale
(Computer Engineering Department, VIT Pune, Maharashtra, India.)

***Abstract*—**Mobile payment is a processing of payment for goods or services with a mobile device such as a mobile phone, Personal Digital Assistant (PDA) etc [1]. Many other ways of performing money transactions like credit cards, debit cards etc. experience severe drawbacks like counterfeiting, skimming.

The main motto behind mobile payment is mobility control is in the hands of mobile user instead of a third party as in case of credit cards[8]. This system provides end to end security and prevents the above mentioned frauds.

AES, SHA and RSA offer end to end security. Confidentiality has been implemented through AES,RSA and SHA provides integrity. Every transaction has a unique session key.

***Keywords*—**AES,RSA,SHA.

I. Introduction

NOw a days transactions are made through credit cards, debit cards etc for payments.

One has to swipe the card for making payment at shopping centre. In case of debit cards one has to enter the secret pin no for the payment. Thus the secret information required for accessing the account of the customer is present on the card, only common security measure on such cards is a signature panel, but signatures are relatively easy to forge. Many merchants will demand to see a photo like a driver's license, to verify the identity of the purchaser, and some credit cards include the holder's photo on the card itself [3].

The existing system doesn't have any special facility to verify the card holder's identity. A common countermeasure currently used is that the user has to enter some identifying information, such as the user's ZIP or postal code. This method may deter casual theft of through card found alone, but if the card holder's wallet is stolen, it may be trivial for the thief to deduce the information by looking at other items in the wallet. For instance, a US driver license commonly has the holder's home address and ZIP code. So there is a need of a system which is highly secure and easy to use. We propose a unique solution called as MPG system.

The introduction section I.A gives brief overview of the system. The section II narrates system design and implementation. The security features have been highlighted in section III. Finally conclusion is in section IV

A. Overview

Brief flow description of MPG is given in the above figure.

1 The mobile user and the merchant logs into the system. So authentication of end users must be done first before commencement of any transaction.
2 The mobile user enters and sends the purchase order information along with merchant's identity to MPG.

Fig. 1. Overview of system

3 The mobile payment gateway sends this order information to the merchant.
4 Merchant give the confirmation regarding the transaction to MPG.
5 The server on receiving the confirmation performs the actual financial transaction.
6 After the transaction is over both parties will get the confirmation message.

II. Design and Implementtaion

In this system we propose a general Java 2 Platform Micro Edition (J2ME)[5] application-layer encryption methodology that uses the Advanced Encryption Standard (AES) Rijndael symmetric block cipher algorithm to provide confidentiality. The integrity between the mobile users and the banking server is provided through SHA1. Our main objective is to provide a highly secure environment that is simple to use and deploy, and does not require any change in the infrastructure or protocols of the wireless network. Application-layer encryption falls in this category since it does not rely on new protocols at the lower layers; the application itself handles all the security-related functions.

The application we designed and implemented provides a prototype solution for securing sensitive data over the wireless network irrespective of the underlying transport protocol used for transporting this data. The only requirement is that of having a MIDP-compliant device.

This section presents a discussion of the design starting with the client environment and moving on to the server environment, the ciphering algorithm, and the session encryption/decryption key management process.

A. The Client Environment

1) Mobile User: The client application follows MIDP 1.0 specifications, which was developed using the J2ME wireless toolkit 2.5 provided by Sun [6]. The application was tested on Nokia N-72 Java cellphone (GSM with GPRS enabled).

The client-side application has following classes.

- passwordMIDlet class, which is the main application class.
- AESEngine class and its utility classes which provide the encryption/ decryption operations and services.
- SHA1Digest class which performs the hashing operations
- Keys class, which is responsible for decrypting the encryption/decryption session keys generated by the server. The Bouncy Castle APIs used to perform encryption/decryption [9].

2) Merchant: So basic classes are:

- AESEngine for encryption/decryption.
- SHA1Digest for integrity check are implemented. To add more security X.509 certification is used between merchant and server. So merchant can verify the authenticity of server.
- Keys class will generate of unique session key between merchant and server, which will be transferred to the server using public key cryptography.
- The Bouncy Castle APIs are used for performing encryption/decryption

B. The Server Environment

To benefit from a pure Java solution, we implemented the server-side application according to J2SE specifications [7]. The server-side application consists of socket connections in addition to the encryption classes and utilities.

We used J2SE server along with JDBC connectivity to access the data from database.

The main server-side application components are :

- Authentication which is responsible for authenticating clients
- Key generation, which is responsible for generating the random session keys.
- Certification which is necessary for the self authentication, and the key storage.
- The Bouncy Castle APIs are used for performing encryption/decryption.

C. The Encryption Algorithm

The encryption algorithm that we used is the AES Rijndael algorithm [9]. AES Rijndael is an iterated block cipher, meaning that the initial input block and cipher key undergo multiple transformation cycles before producing the output. The algorithm can operate over a variablelength block using variable-length keys; a 128-, 192-, or 256-bit key can be used to encrypt data blocks that are 128, 192, or 256 bits long, and all nine combinations of key and block length are possible. The algorithm is written so that block length and/or key length can easily be extended in multiples of 32 bits, and the system is specifically designed for efficient implementation in hardware or software on a range of processors.

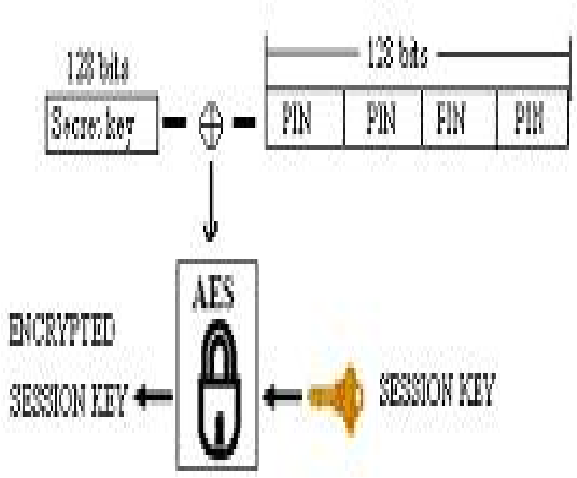

Fig. 2. Cipher Key Management

In our application we used a block size of 16 bytes processed with 128-bit keys: this proved to be the best combination for operation on J2ME devices due to the speed and memory limitations of such devices.

D. Cipher Key Management

Securing the communication between the client and server is our primary concern. For this reason we implemented a session-key management mechanism where the encryption/decryption keys are randomly generated for every client session. This mechanism is addressed as follows:

The client and the server uses different 128-bit session key for each session. At the start of every client session, the server randomly generates this session key and stores it in the client's specific entry in the database. The server then encrypts these session key by the key formed by x-oring 128 bit pin and 128 bit secret key shared between client and server. This encrypted session key is transmitted to the client where it is stored in local variables there.

Another important issue that must be addressed is securing the storage of the shared secret on the client and server. On the server this shared secret is stored in the database server, which we assume to be secured by the database management system and other computer security policies. Securing the shared secret on the client machine is somehow more challenging. To protect the shared secret on the client: it is stored encrypted, in the Keys Java class in the application's JAR file. The shared secret key is encrypted by the client's pin code (128 bits), since AES requires that the key length is 128 bits.

To prevent the attacks like spoofing, origin repudiation, alteration of message etc. it is necessary to maintain data as well as origin integrity. This is done by computing and sending the digest along with every message. The digest is computed separately on other side and compared with the received one to ensure the integrity. SHA1 algorithm with 160 bit output is used for this purpose.

At the time of subscription to the mobile banking service, it is the responsibility of the service manager to encrypt the shared secret with the client's pin code when the application is distributed and to store it on the mobile phone. This way, even if the phone is stolen and the application code is unpacked, the intruder will not be able to retrieve the actual value of this shared secret and can not make much use of the encrypted value. It is only when the client logs on to the system at run time that this shared secret is decrypted by the client pin code and used, together with the pin code, to decrypt the encrypted session key which is then used in the actual data encryption/decryption process.

1) Certification and Server authentication : Initially bank will fill up the details and will generate a X.509 certificate. Then it will request a Certificate authority to sign this certificate using authority's private key. Then after verifying details authority will sign the certificate and then bank server will store this signed certificate. So now server certificate will contain the public key of the server.

When merchant connects to the server, server will transfer his certificate to the merchant. As the public key of certificate authority is known to everybody at merchant side the certificate is verified. And if the verification is successful then now merchant has the valid public key of the server. That means server has proven his authenticity to the merchant [10].

2) Merchant side key generation: As merchant needs to confirm the transaction request, message needs to secure. So as there are no constraints on merchant side, merchant generates the unique session key between him and server. As the merchant has the valid public key of server so he can send the session key encrypted using public key of user so that only server can decrypt the same using private key and will store it and will use the session key for the further communication with merchant.

E. FLOW OF SYSTEM

The actual flow of the system is as follows:

1 MPG server to CA : Certification request.
2 CA signs bank's certificate using private key of itself and sends certificate to MPG.
3 Merchant to MPG server connection request.
4 MPG server to Merchant: Server's X.509 certificate.
5 Merchant verifies certificate using public key of authority and gets the public key of server as Kspub.
6 Merchant generates the session key Ksession1.Merchant to MPG : uname+EKspub[H(uname+pass+TS+nonce)+ Ksession1]
7 Server will decrypt message using its private key and will authenticate merchant and if authentication is successful. It will send.
MPG to Merchant: EKsession1 [nonce+1]
8 Mobile User to MPG server: Uname,MID, TS, nonce, H(Uname, Pass, MID, TS,nonce).
9 MPG will authenticate customer and will generate session key Ksession2. MPG to Mobile User : EKsession2 (TID, Merchant Info.) + EPIN+Kshared (Ksession2).
10 Client enters pin no to decrypt the shared secret key. And then using pin+kshared session key(Ksession2) will be decrypted. Mobile User to MPG : EKsession2 [Amt, MID] + EKsession2[H(Amt,MID]+ TID.
11 MPG to Merchant : EKsession1 (Amt, Customer Info., TID).
12 Merchant to MPG: EKsession1 (Confirmation, TID)
13 MPG to Customer: EKsession2 (Transaction successful + TID).
14 MPG to Merchant: EKsession2 (Transaction successful + TID).

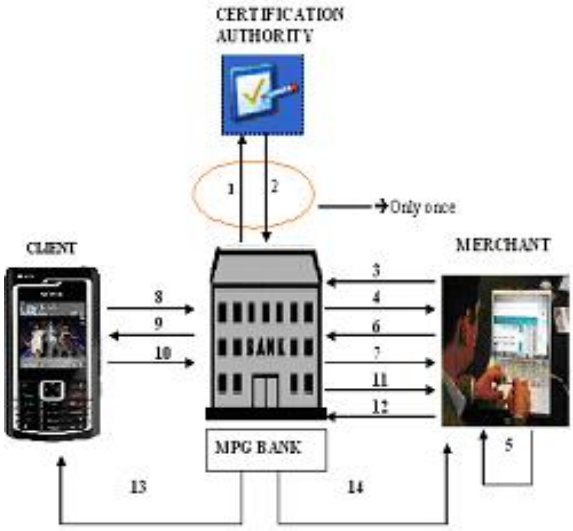

Fig. 3. Flow Of System

Steps 1, 2 will be executed once just to get certificate from authority.

III. SECURITY FEATURES

Confidentiality: The confidentiality of all the messages is maintained using the session keys which will be unique per transaction. The session keys for mobile user and merchant are different. The session keys are exchanged securely. The session key is obtained only form a combination of pin and a shared secret key in case of mobile customer. RSA is used to exchange session key between merchant and mpg server.AES-128 is used to encrypt data on mobile customer and server side. Assuming that one could build a machine that could recover a DES key in a second (i.e., try 255 keys per second), then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a 128-bit AES key [2].

Integrity: The integrity of the message is maintained using a digest which is attached with every message. The digest is

Fig. 4. Sample Output

computed at the destination and is compared with that attached to the message. If it is not valid then discard the message since its integrity is violated.

Authentication: The nonce and password scheme is used for mutual authentication between mobile user and mpg server. RSA certification is used in merchant and MPG.

Man in the middle attack: This attack is encountered using unique session key for every transaction. There are different session keys for mobile user and merchant communication. The session keys are securely exchanged using a shared secret key at mobile customer side and using RSA certificate infrastructure. There is a limited period for every transaction that is conducted and in the case of timeout the transaction is immediately aborted. Thus it is difficult to sniff any information.

Replication attack: This attack is encountered using the timestamp in the message. The replicated message will contain an older timestamp. Hence a replicated message will be easily caught and will be discarded.

IV. Conclusion

There are frauds involved in credit card payment. These frauds threaten user for payment at the hands of merchant.

The solution to this insecure credit card payment system is a secure mobile payment gateway. Care is taken off all data that flows in the system. The data that is sent is always encrypted with the control of payment totally in the hands of mobile user. End to end security is the key feature of the project. In addition the mobility is provided to the user with his own handset. GPRS is the technology that provides user a fast and reliable solution.

The merchant application enforces security with involvement of digital certificates in the authentication and authorization process. Thus the paper narrates a secure mobile payment solution.

References

[1] Mobile payment http://en.wikipedia.org/wiki/Mobile payment
[2] G. Keating, "Performance Analysis of AES Candidates on the 6805 CPU Core," http://members.ozemail.com.au/geoffk/aes 6805/paper.pdf, April 1999.
[3] M. Soriano and D. Ponce, "A Security and Usability Proposal for Mobile Electronic Commerce," IEEE Communications, August 2002.
[4] V. Gupta and S. Gupta, "Securing the Wireless Internet," IEEE Communications, December 2001.
[5] G. Lawton, "Moving Java into Mobile Phones, "IEEE Computer, June 2002
[6] Sun Microsystems, J2ME Wireless Toolkit 2.5 User's Guide, June 2002.
[7] Sun Microsystems, J2SE Platform Specification v1.4, August 2006
[8] M. Soriano and D. Ponce, "A Security and Usability Proposal for Mobile electronic Commerce," IEEE Communications, August 2006.
[9] Beginning Cryptography with Java by David hooks Wrox publication.
[10] E-commerce book by Chen