A PKI-BASED END-TO-END SECURE INFRASTRUCTURE FOR MOBILE E-COMMERCE *

Tin-Wo Cheung and Samuel T. Chanson
Department of Computer Science
Hong Kong University of Science & Technology
Clear Water Bay, Kowloon, Hong Kong
{cswo, chanson}@cs.ust.hk

Abstract The popularity of handheld mobile devices, the move towards electronic commerce, and deployment of the public key infrastructure in many parts of the world have led to the development of electronic commerce on mobile devices. The main challenge is the limited computing capacity on these devices for PKI-based end-to-end secure transactions. This paper presents a new architecture and protocol for authentication and key exchange as well as the supporting infrastructure that is suitable for the mobile environment. The system requirements and our solutions in addressing these requirements in the restrictive environment are discussed. An evaluation of the system performance is also included. The system has been implemented and is supporting some real-life applications.

Keywords: wireless security, key exchange protocols, mobile e-commerce, PKI end-to-end security, smart card, GSM Short Message Service

* This work was supported by an ISF grant from the Hong Kong Innovation and Technology Commission (AF/153/99).

1. INTRODUCTION

1.1 Background

The rapid advances in wireless mobile communication technology and the growth of electronic commerce have naturally led to the development of electronic commercial services on the wireless medium through mobile phones. In Hong Kong as well as many parts of the world, the population of mobile phone users has more than doubled the number of Internet users. The popularity and high penetration rate of the mobile phone can be attributed to its convenience, ease of use, and low cost of ownership and operation. For business transactions conducted on electronic means, security is a major concern. Both the Internet and the wireless network are public networks and considered to be insecure, where messages can be eavesdropped, captured, modified and inserted by intruders. Intruders may also impersonate as legitimate parties for personal gain. Therefore, some mechanism is needed to guarantee the confidentiality, authenticity and integrity of the transmitted messages. Internationally, the Public Key Infrastructure (PKI) is accepted as an effective means to tackle the above security problem.

In January 2000, Hongkong Post [4] started its digital certification service and became the first recognized Certification Authority (CA) in Hong Kong. The Electronic Transactions Ordinance enacted in March the same year gives electronic signatures the same legal stature as handwritten signatures. Therefore, as in a number of other countries, a sound and legal public key infrastructure has been established in Hong Kong that is ready for commercial activities on the electronic medium.

1.2 Objective of Work

Our objective is to develop a PKI-based open infrastructure that supports end-to-end secure electronic transactions through mobile phones. We are not just looking for a theoretical solution; rather the solution must be practical, efficient, and implementable. Besides the security concerns, efficiency and availability of supporting hardware products are also important. The main challenge in this project is the limited capacity of the mobile device, especially the mobile phone [10, 12, 13, 14, 15, 17]. About the only computing facilities available on the mobile phone are those offered by the SIM card which has a very slow speed processor and little memory (of the order of 16 KB). The resources on current SIM cards are not sufficient to perform general PKI-based authentication. Moreover, the wireless network is error-prone and slow compared to wired networks. We have designed a set of key exchange and authentication protocols that can run on a thin client model.

The infrastructure consists of a number of servers offering supporting services for on-line purchase and payment in addition to authentication. Security is end-to-end rather than link-by-link.

The rest of the paper is organized as follows. Section 2 describes related works by others. The requirements of our system are given in Section 3 while Section 4 provides an overview of the system architecture. Section 5 presents our key exchange and authentication protocols, and Section 6 discusses some system implementation issues. System performance is presented in Section 7, and finally Section 8 concludes the paper.

2. RELATED WORKS

For secure digital communication, the communicating parties need to be sure of the identity of one another and the contents of their messages must not be tampered with in any way even if the communication network is insecure. An authentication mechanism is needed to satisfy the first requirement and a secret key exchange mechanism is needed to achieve the second goal. This is known as the *authenticated key exchange (AK)* problem, and a variation is called the *authenticated key exchange with key confirmation (AKC)* problem in [9]. Both problems are concerned with key agreement among the communicating parties. The *AK* protocol makes sure the key can be successfully exchanged only if the identities of the communication parties are genuine. However, it does not check if the exchanged key really works. The *AKC* protocol further adds messages to verify the key being established is correct. In this paper, we shall use the term *AKC* protocol to refer to the authentication and key exchange protocol in general.

2.1 Categories of AKC protocols

There is a vast amount of literature on AKC protocols. In a survey article [20], Clark et al. categorize AKC protocols according to criteria such as the cryptographic approach used, whether a trusted third party is present, and the number of protocol messages exchanged. In this section, we propose an alternative approach and categorize the published AKC protocols based on the methodologies used in authentication and key exchange. Most existing work deals with two communicating parties only.

2.1.1 Key exchange with Implicit Authentication

This class of protocols mainly apply the Diffie-Hellman [11] key exchange algorithm for exchanging keys between the communication parties. In this case, when two parties need to establish a session key, they exchange their

public keys. An identical data can be generated by each party by combining its own private key with the counterpart's public key (see Formula (A)). Since only these two parties are able to generate the secret data, proof of the ability to generate the data implicitly authenticates the participants.

The Beller-Chang-Yacobi [12] protocol is designed for the mobile environment, in which the client is a low-power portable mobile device. Let PK_j and SK_j be the public key and the corresponding private key of the base station respectively.

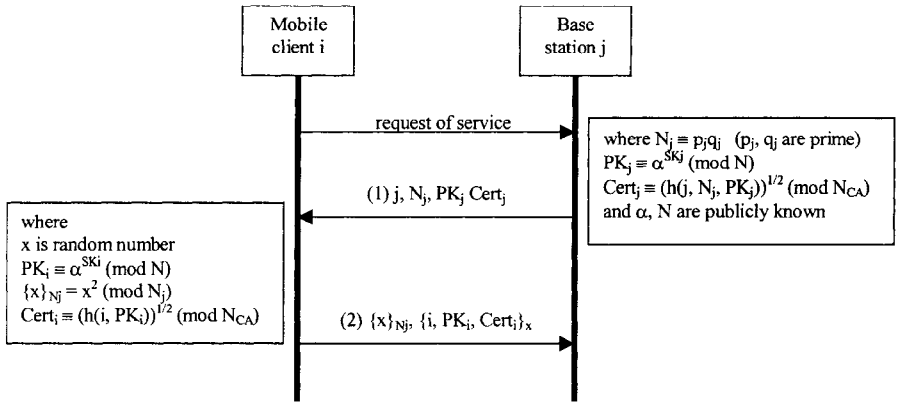

Figure 1. Beller-Chang-Yacobi Protocol

When both parties have successfully verified the certificates of their counterparts, they can calculate the value η using the Diffie-Hellman technique:

$$\eta \equiv (PK_j)^{SKi} \equiv (PK_i)^{SKj} \pmod N \quad \text{---- (A)}$$

A session key *sk* is calculated by symmetrically encrypting a random number x with key η, i.e., $sk = f(\eta, x)$.

The idea is similar to Diffie-Hellman in that a common session key can only be calculated by those with the corresponding private keys. The certificates ($Cert_i$ and $Cert_j$) in the protocol are for the purpose of ensuring the public keys used are valid.

Carlsen [13] and Zheng [14] have pointed out the weakness of this protocol due to the weak authenticity mechanism. The identities of both parties have not been authenticated even after message 2 has been received. This means either one can be an impersonator at this point without being detected.

2.1.2 Explicit Authentication by Challenge-Response with key exchange

The simplest way to authenticate a party is the "Challenge-Response" method. The responding party holds a secret, such as a secret symmetric key or a private key, that enables him to answer the challenge correctly. When some random data are embedded in the challenge and/or response messages, these data can be used to generate the session key after successful authentication.

The Aziz-Diffie protocol [15] is another certificate based AKC protocol. The difference between Beller-Chang-Yacobi and Aziz-Diffie is that the Aziz-Diffie protocol is based on the challenge-response mechanism. Therefore, we regard it as an authentication-oriented protocol with key-exchange functionality, while the Beller-Chang-Yacobi protocol is a key-exchange oriented protocol with implicit authentication functionality. The main design goal of Aziz-Diffie is to achieve both privacy of wireless data communication as well as mutual authentication between the communicating parties. The protocol can be summarized in the following diagram:

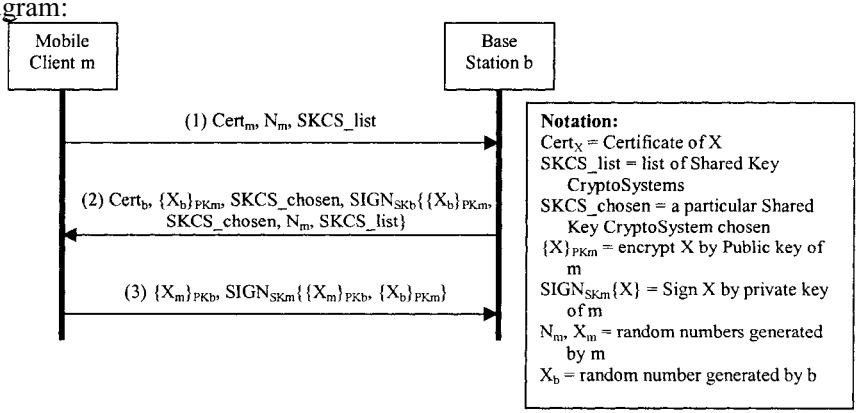

Figure 2. Aziz-Diffie Protocol

The mobile client initiates the protocol by sending N_m in message 1 as the first challenge to the base station. The signature in message 2 contains the element N_m which proves ownership of the private key of the base station b whose certificate $Cert_b$ is also passed to Client m in message 2. At the same time, the random number X_b in message 2 is encrypted by the public key of m as a challenge to the mobile client. In message 3, the mobile client responds to this challenge by decrypting it with its own private key and then encrypts it with the public key of the base station b. Since only the holder of the private key m can decrypt X_b correctly, the base station can authenticate the mobile

user after receiving message 3. As a result, mutual authentication is achieved. The session key is calculated by X_b XOR X_m.

The Aziz-Diffie protocol is a typical challenge-response AKC protocol. However, many other proposals take a hybrid approach. In [10, 16], both Diffie-Hellman factors and challenge-response pairs are involved in the protocols messages. Therefore, mutual authentication as well as key establishment are achieved simultaneously.

2.1.3 Explicit Authentication by synchronised data with key exchange

Another approach to explicitly authenticate a party is by using some kind of synchronized data. The principle is basically the same as that of the challenge-response approach. The difference is instead of random numbers, synchronized data such as timestamps from synchronized clocks or counters are used. It is important to ensure the freshness of the synchronized data to avoid replay attacks. In [14], timestamps are used. However, in reality, it is difficult to guarantee the clocks are synchronized.

The Lin-Harn [17] protocol employs a chained hash technique for generating the synchronized data. Before a session key can be established, a registration process is required where the mobile client generates a group of hash chains, signs the “tail” of each chain and then sends to the base station for registration. The client can prove its identity by revealing a previous image of the registered hash chain. Due to the non-reversible property of the hash function, the ability to present a previous image in the chain implies ownership (generator) of the chain. The verifier can verify a received image by hashing it a number of times and compare the result to the ‘tail’ image.

In the Lin-Harn protocol, the synchronized data as well as the challenge is the last revealed image in the hash chain. The verifier accepts an image only if it is a previous image of the last revealed one. The ability to provide a valid response to this challenge implies the ability to reverse the hashing operation to recover the previous images. Since only the generator of the hash chain knows the “head” image of the chain, the authenticity requirement can be satisfied. Note that in this protocol a single message is enough for authentication.

2.2 Standardized AKC protocols

In order to make use of the AKC protocols in real life, standardization of some generally accepted protocols is essential. Normally, the standardized protocols are independent of the cryptographic algorithm. Negotiation of parameters is generally required at the beginning of the protocol.

2.2.1 Secure Socket Layer (SSL) [1]/Transport Layer Security (TLS) [7]

SSL/TLS is a well-known standard that is generally used for secure Internet web browsing. Its handshake protocol is designed to address the AKC problem between the SSL server and SSL client. The authentication mechanism is based on the challenge-response approach discussed above.

There are two important concepts of SSL: *SSL session* and *SSL connection.* A *SSL session* associates a client and a server. It is created through a handshake protocol involving negotiation of version and algorithms to be used in the session. After the handshake protocol, both the client and the server will be able to calculate a secret data for the *SSL session* called the *master secret.* Every *SSL connection* is a peer-to-peer logical connection built on top of a *SSL session* for data exchange in the application layer. In a *SSL connection,* the two peers share a set of secrets derived from the *master secret* of the *SSL session.* This set of secrets is used for encrypting the exchanged data and creating the message authentication codes (MACs) used to ensure data integrity.

The main purpose of the *SSL connection* is to enhance the efficiency of the protocol by reusing the *master secret* of a *SSL session* in different *SSL connections* between the same client and server. The handshake protocol for *SSL connection* establishment contains fewer messages and involves no public key cryptographic operations which are CPU intensive.

SSL is suitable for wired Internet environment. However, for restrictive devices such as the mobile phone or smart card, the protocol is too heavy for implementation. Moreover, a *SSL session* can only be reused for connections between the same pair of client-server. If a client tends to communicate with different servers, a new *SSL session* must be establishment for each server.

2.2.2 Wireless Transport Layer Security (WTLS) [8]

WTLS is the security layer protocol for the Wireless Application Protocol (WAP) architecture to provide confidentiality, data integrity and authenticity of the communicating applications. The design of WTLS bears a close resemblance to the design of TLS 1.0. Because of the low bandwidth and unreliability of the wireless network, WTLS is concerned with datagram support, optimized handshake and a key refreshment scheme.

One limitation of WTLS is it only provides point-to-point security in most cases. Figure 3 shows a general configuration of a WAP network. WTLS provides a secure channel between the WAP user and the WAP gateway. But it does not protect the communication all the way to the Wireless Markup Language (WML) server. Typically, a *SSL session* is established between the WAP gateway and the WML server if data security is a concern. In reality, WAP gateways are maintained by mobile operators and WML servers are

owned by service providers. Therefore, WTLS normally only provides a point-to-point secure solution to the service provider (WML sever) and the end user. If end-to-end security is required, either the WAP gateway has to be maintained by a trusted party or the service provider must maintain the WAP gateway herself.

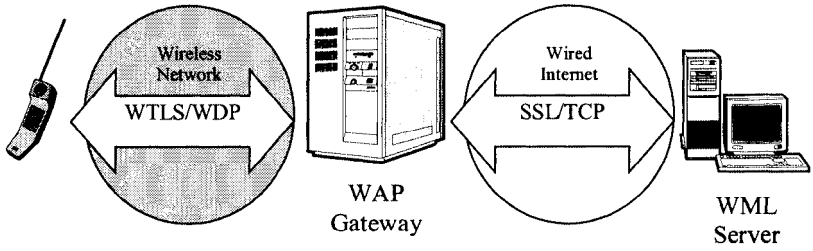

Figure 3. WAP Network configuration

2.3 Discussion

Different AKC protocols have been designed to tackle different issues, such as security of the protocol, restricted computational power of the client or privacy of the customers. However, few of the proposed solutions have considered utilizing the existing mobile system for implementing security adds-on, and as far as we know only the WTLS protocol has been implemented and deployed outside of the laboratory. The practical constraints and problems, therefore, have rarely been addressed in the literature. WTLS, however, does not provide end-to-end security between the user and the service provider.

In this work, the constraints of implementing an AKC protocol on the existing GSM mobile network using existing hardware products are discussed. A trusted third party is introduced in our AKC protocol to share the workload of certificate verification with the mobile client. Our design considerations are on security, restricted processing resources of the client, number of messages required and the size of each message packet.

3. SYSTEM REQUIREMENTS

In our system, users conduct mobile electronic transactions from a GSM dual-slot cellular phone. The phone has an additional smart card slot besides the GSM SIM card. The second smart card is chosen to be a Java Card [3] with 16K EEPROM and a cryptographic co-processor which can perform RSA cryptographic functions, triple-DES cipher, SHA-1 hashing and random

number generation. All AKC protocol processing as well as transaction operations are performed on the processor chip on-board the smart card. The GSM SIM card is only responsible for providing the wireless communication channel and invoking the applications on the second card. Message packets between the phone and the base station are transmitted through SMS (short message service) of GSM. A SMS packet has a size limitation of 140 bytes.

4. OVERVIEW OF SYSTEM ARCHITECTURE

Due to the scarce resource for both memory space and computational power, the mobile equipment (ME) is incapable of verifying a X.509 digital certificate to authenticate a service provider. We have developed a server called the User Authentication Server (UAS) to act as a trusted third party to assist the mobile client to authenticate and exchange keys with the service provider, which is named the PKI End-to-end Secure Module (PESM). The following diagram shows the system architecture. Each component will now be described below.

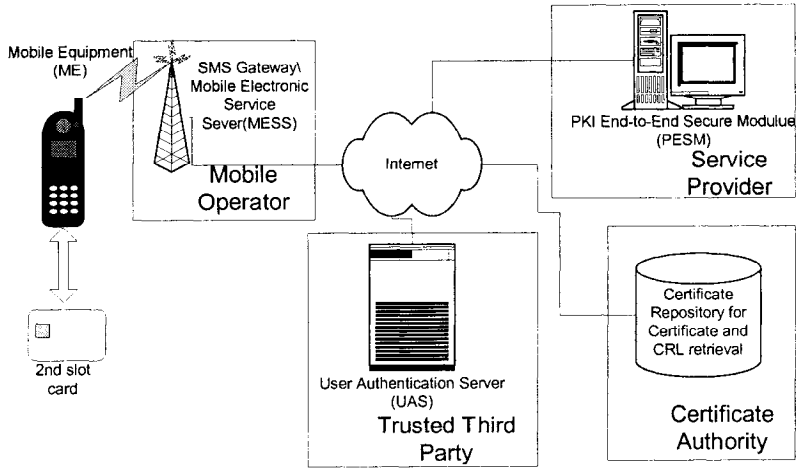

Figure 4. Overview of the system architecture

4.1 SMS Gateway and Mobile Electronic Service Server (MESS)

The SMS Gateway and MESS together act as an interface between the wireless and wired networks. MESS interprets the header of message packets and routes the packets to the proper MEs and servers. It is unable to read the message contents since they are encrypted at source.

4.2 Mobile client

The mobile client is a portable device which in our case is a dual slot GSM phone and a smart card with cryptographic functionality. Each user is required to have his own digital certificate issued by a valid Certification Authority (CA). The corresponding private key is stored in the user's second slot smart card. Moreover, we require the UAS's public key be pre-loaded on the card as well. In subsequent sections, the mobile client is abbreviated as ME (Mobile Equipment) for simplicity.

4.3 User Authentication Server (UAS)

The UAS is a centralized server that should be operated by a trusted third party. Its role is to help the ME to authenticate the party it is communicating with. First, mutual authentication is performed between the ME and UAS. Then, the UAS authenticates the PESM on behalf of the ME. Following that, a PESM session key is exchanged between the ME and PESM to establish an end-to-end secure communication channel.

4.4 PKI End-to-end Secure Module (PESM)

PESM is a server operated by the service provider. It is responsible for ensuring security at the application level, includes authentication, confidentiality, integrity and non-repudiation. For authentication, it performs the handshake protocol to authenticate the UAS or optionally authenticate the mobile client and establishes a session key. For confidentiality, it encrypts and decrypts messages sent and received from the mobile client using the established session key. Furthermore, it verifies the Message Authentication Code (MAC) of each message to guarantee integrity. For non-repudiation, it verifies the digital signature of a message if it is present.

4.5 Certificate repository

The certificate repository is a service provided by the CA which allows the public to access the issued digital certificates. It is usually implemented by a LDAP server on which object records can be searched by subjects. The UAS and PESMs will access this server from time to time to retrieve digital certificates for verification purposes.

5. PROTOCOLS

5.1 Design Criteria

Our AKC protocol is based on a 3-tiered model involving the ME, UAS and PESM. With the assistance of the UAS, the ME and PESM establish a session key between them for encryption and MAC calculation. Prior to key establishment, authentication is required between the UAS and ME, and then the UAS and PESM. For certain applications, the PESM may want to authenticate the client (ME) in order to determine the privileges allowed. If this is not necessary, the PESM normally does not need to authenticate the client. The ME and UAS, however, must authenticate the PESM to ensure they are talking to the right service provider.

The communication channel between the ME and MESS is furnished by the GSM SMS (Short Message Service) service. SMS is a datagram based packet switching channel in which message packets can be corrupted, delayed or lost. The maximum size of each message packet is 140 bytes and there is a significant delay associated with each SMS packet transmission. Furthermore, the slow processing speed of the ME and the limited memory space for program code and data on the ME's second slot smart (of the order of 16K Bytes) should also been taken into account in the design of the protocol. Therefore, the protocol should satisfy the following requirements:

- The size of each message packet on the wireless network must be less than 140 bytes
- The number of messages to be transmitted on the wireless network should be minimized
- The process on the client side must be simplified in order to minimize the code size and processing time on the second slot smart card
- The protocol must be able to tolerate the errors on the wireless transmission channel

5.2 Description of Protocols

Our protocol is divided into 2 phrases, namely: *UAS Session Establishment* and *PESM Session Establishment.* This is illustrated in the following diagram (Figure 5).

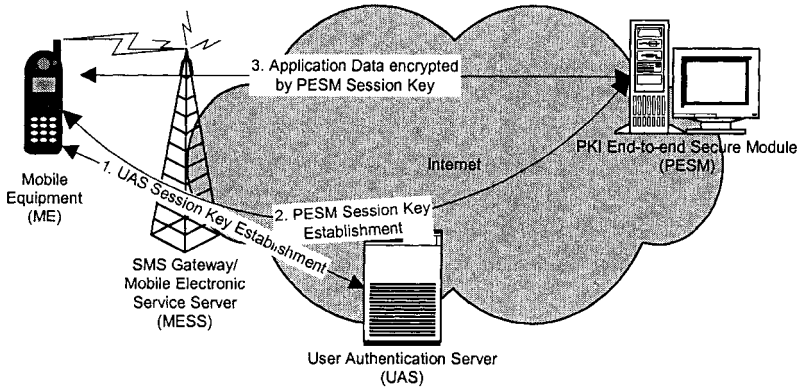

Figure 5. Steps of Key establishment

We use the following notations in describing our protocol:

Table 1. Symbols used in protocol description

Symbol	Description
ID_P	A unique identifier of entity P
$E_P\{x\}$	Encrypt x by P's public key
$Cert_P$	Digital certificate of P
$ES_{KEY}\{x\}$	Encrypt by symmetric key block cipher in CBC mode (3DES) with the key "KEY"
$hash\{x\}$	Hash the value x
$f(x)$	Some kind of one-way function for session key diversification
$N \in_R \{0, 1\}^k$	Randomly generate k bits of binary data N
$A \mid\mid B$ or A,B	A concatenates with B

5.2.1 UAS Session establishment

The session is established between the ME and UAS using a two-pass AKC protocol based on a general challenge-response authentication mechanism (see Figure 6).

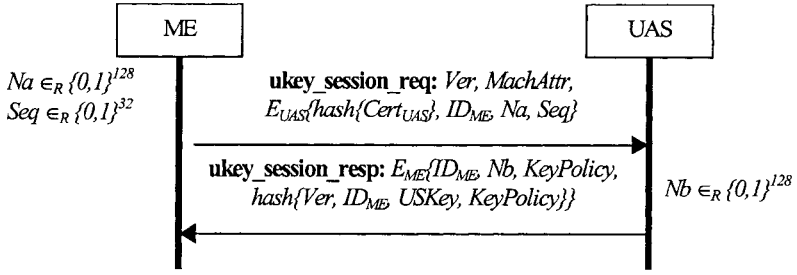

Figure 6 UAS Session Establishment

Table 2. Definition of terms used in UAS Session Establishment protocol

Items	Description
Ver	Version of the protocol
MachAttr	Configuration attribute of the ME (eg. Language)
Na	A random number generated by ME
Seq	A random number generated by ME as the starting sequence number of this session
Nb	A random number generated by UAS
USKey	UAS Session Key calculated from $f(Na \| \| Nb)$
KeyPolicy	A value defining the lifetime of *USKey*

The ME initiates the establishment of a secure session with the UAS by performing the following operations:

1. Randomly generates *Na* and *Seq*.
2. Encrypts ($hash\{Cert_{UAS}\}$, ID_{ME}, *Na, Seq*) using UAS's public key.
3. Composes and sends *ukey_session_req* to UAS.

When UAS receives the *ukey_session_req* message, it should

1. Decrypt the message using its own private key.
2. Check if $hash\{Cert_{UAS}\}$ is the fingerprint of its current certificate. If the check fails, the protocol cannot be continued since the ME does not have the correct public key of the UAS.
3. Randomly generate *Nb* and calculate *USKey* (UAS session key) from $f(Na \| \| Nb)$.
4. Determine the lifetime of the session key and specify it in the value of *KeyPolicy.*
5. Compose and send *ukey_session_resp* to ME.

On receiving the *ukey_session_resp* message the ME verifies the validity of the message by generating its own value of $Hash\{Ver, ID_{ME}, USKey, KeyPolicy\}$ and comparing it with the one in the message

Since only the valid UAS can decrypt *ukey_session_req* to get the value of *Na,* ME can authenticate UAS by checking the-correctness of the *ukey_session_req* message. If the message is correct, ME accepts *USKey* and *KeyPolicy.*

In the above protocol, only one-way authentication of UAS is achieved. Adversaries can impersonate the ME by creating its own *ukey_session_req* message. Therefore, the UAS does not accept this newly established session yet. Instead, it stores the state parameters (i.e., *USKey, KeyPolicy* and *Seq*) of the session as a pending state and switch to the current state only after the ME has further authenticated itself in the *PESM Session Establishment* protocol that follows.

5.2.2 PESM Session Establishment

After the UAS session has been established (either in pending state or current state), the ME may start the *PESM Session Establishment* protocol in order to establish a secure communication session with the service provider. A request is sent by the ME to the UAS specifying which PESM it would like to talk with. The UAS then communicates with the target PESM on behalf of the ME.

Before the UAS can start the key exchange protocol with a PESM, it may have to interact with the PESM to find out the key exchange mode required and exchange the related certificates. If the information is already known then this step can be skipped.

The PESM can choose from two authentication modes of session key establishment when it receives an enquiry: Server Authentication and Client Authentication. Server Authentication means the PESM does not need to authenticate the ME. Otherwise, Client Authentication mode is used.

Server Authentication

If Server Authentication mode is selected, the protocol runs as follows:

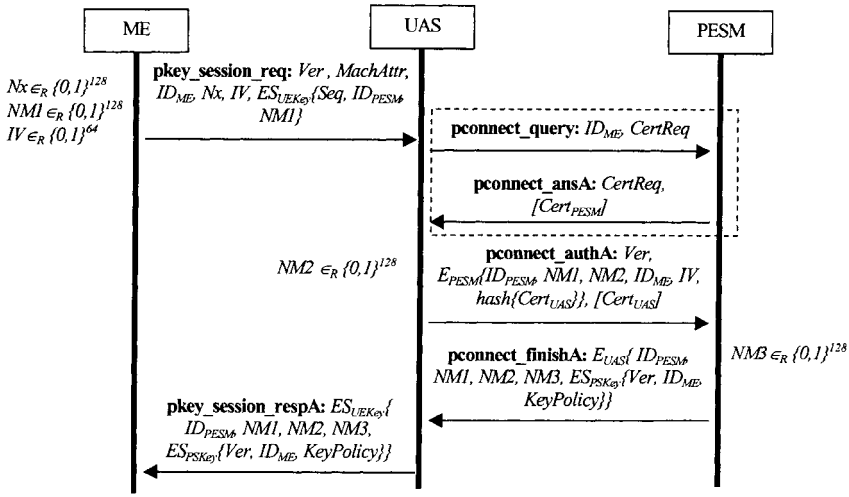

Figure 7. Protocol for PESM Session establishment in Server authentication mode

Table 3. Definition of terms used in PESM Session Establishment protocol

Items	Description
Nx	A random number generated by ME
Seq	Sequence number of this session
NM1	A random number generated by ME

NM2	A random number generated by UAS
NM3	A random number generated by PESM
IV	Initialization Vector for CBC mode encryption
UEKey	UAS Encryption Key which is calculated from *f(USKey || Nx)*
PSKey	PESM Session Key which is calculated from *f(NM1||NM2||NM3)*
KeyPolicy	A value define the lifetime of the *PSKey*
CertReq	A list of CAs which is recognized by the sender. If this list is empty, it means certificate is not requested.

The ME initiates the protocol with the following actions:

1. Randomly generates *NM1* and *Nx* and calculates *UEKey* from *f(USKey||Nx).*
2. Increments *Seq.*
3. Encrypts (*Seq, ID_{PESM}, NM1*) using *UEKey.*
4. Composes and sends *pkey_session_req to* UAS.

When UAS receives the *pkey_session_req* message, it

1. Computes *UEKey* using the received *Nx* and its own *USKey.*
2. Decrypts the message using *UEKey.*
3. Checks if the value of *Seq* is valid. UAS will only accept *Seq* if it is larger than the last accepted *Seq* but falls within a certain predefined range. This mechanism is to avoid intruder's attack by replaying the *pkey_session_req* message.
4. If this message is valid, UAS switches session state from 'pending' to 'current'.
5. If UAS has no information about the mode of authentication or the certificate of PESM, a *pconnect_query* message is sent to the PESM. After receiving the *pconnect_ansA* response, UAS checks if the PESM's certificate was issued by one of the CAs listed in the non-empty *CertReq.* If it does not check out, the session cannot be established.
6. Randomly generates *NM2.*
7. Encrypts the elements in the *pconnect_authA* message using PESM's public key and sends the message to PESM.

On receiving the *pconnect_authA* message, the PESM

1. Decrypts the message using its private key.
2. Checks if the UAS certificate fingerprint in the message matches that of the certificate.
3. Randomly generates *NM3* and computes the *PSKey* by *f(NM1||NM2||NM3).*
4. Determines the lifetime of the *PSKey* and assigns the value of *KeyPolicy.*
5. Encrypts (*Ver, SRN, KeyPolicy)* using *PSKey.*
6. Composes the *pconnect_finishA* message, encrypts it using the public key of UAS and sends to UAS.

On receiving the message *pconnect_FinishA,* the UAS can authenticate the PESM by checking if the values of *NM1* and *NM2* are the same as what were sent in the *pconnect_authA* message. This is, again, a simple challenge-response mechanism since the values of *NM1* and *NM2* can only be obtained by the holder of PESM's private key. After authenticating the identity of PSEM, the UAS forwards the needed1 data to ME needed to calculate the PESM session key.

Note that the message type of the response by PESM indicates the mode of authentication. A *pconnect_ansA* message means server authentication is required. This step may be skipped if UAS already has this information.

Client Authentication

If the PESM requires client authentication, a *pconnect_ansB* message is sent instead of *pconnect_ansA.* The only difference in message packets contents between these two modes is that the public key of ME is used to encrypt the message sent from PESM. As a consequence, only the ME is able to calculate *PSKey.* The protocol runs as follows:

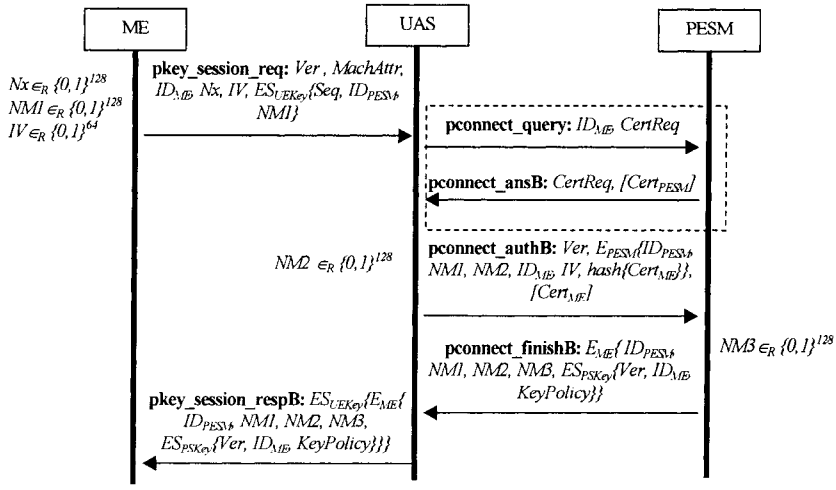

Figure 8. Protocol for PESM Session Establishment in Server authentication mode

5.3 Discussion

All authentication mechanisms in the protocols are achieved by the "challenge-response" approach. In both the *UAS and PESM Session* protocols, two messages are exchanged on the wireless network, which is the minimal number for challenge-response. However, this can achieve one-way

authentication only. For mutual authentication, at least 3 messages are needed. Our approach of achieving mutual authentication with the minimum number of messages is to combine the last response with the message in the next protocol. In other words, completion of the authentication procedure is postponed until the next protocol. For example, in the *UAS Session Establishment* protocol, authentication of ME is not achieved until the start of the following *PESM Session Establishment* protocol. The same is done in the *PESM Session Establishment* protocol. Because of this, PESM and UAS should put the unauthenticated session in the pending state until confirmation can be made.

Another interesting property of this protocol is that since every protocol is a 2-pass protocol, the servers (i.e., UAS in the *UAS Session Establishment* protocol and PESM in the *PESM Session Establishment* protocol) are stateless. This is particularly important from the efficiency point of view since the servers do not need a waiting state.

6. IMPLEMENTATION ISSUES

This AKC protocol is designed to shift as much computational load as possible from the mobile device (ME) to the servers without compromising security. The infrastructure consists of a number of components as described in Section 4. In this section, some practical issues on the implementation of these components are discussed. There are 4 major entities involved in a transaction: mobile client, mobile operator, service provider and trusted third party.

As mentioned earlier, the mobile client is composed of three components: a dual slot GSM mobile phone, a GSM Subscriber Identity Module (SIM) which is a smart card with SIM Application Toolkit [21] enabled, and a second slot smart card. We have defined a set of protocols between the SIM and second slot cards so that they can communication with each other. The workflow is controlled by the second slot card. The SIM and mobile phone together provide a user interface and communication channel for the applications on the second slot card. In our implementation, the second slot smart card is a Java Card [3] which is compliant with the Java Card 2.1 standards. The card has a crypto co-processor on-board that supports public key cryptographic functions in hardware. The Java card was chosen for its flexibility and ease of programming; and the crypto engine is needed for speed in executing the RSA operations. We have tried three different brands of this type of card to test the generality of the infrastructure. The code size of the protocol on the card is about 6~7 Kbytes.

For the mobile operator, a prototype of the MESS (see Figure 4) is implemented. This is a server that interfaces the wireless network with the wired network. It is physically connected to a SMS gateway and also the

Internet. A set of message headers is defined for the MESS to route the associated messages from the SMS gateway to the appropriate destinations.

Because of the 140-byte limitation of the SMS message packet, all messages in our protocol are designed to satisfy this constraint. However, for the few cases when an application data message exceeds this limit, it will be broken down into two or more SMS packets. MESS is responsible to combine the fragmented SMS packets back to a single message according to the information in the headers.

7. PERFORMANCE

We have measured the various segments of time needed to execute the protocols on the mobile device in order to analyze the performance of the protocols and determine the system's bottleneck.

First we examine the time required for transmitting a SMS message (see Table 4). This time mainly depends on the performance of the mobile handset and the GSM SIM card. Different products can exhibit very different performance. The purpose of the study is not to compare the service provided by different mobile operators or to compare the performance of different mobile phone and SIM card products. We are only interested in gaining some ideas on the amount of time needed by each process involved in the transmission of a short message.

Table 4. Timing for SMS transmission

Operation	**Time (in sec)**
Fixed overhead for sending each SMS packet which is independent of packet size (depends only on the handset)	4.43 ~ 6.39
Transmission delay of 140 bytes of data	1.75
Processing delay of 140 of bytes data (handset dependent)	0.22 ~ 0.28
Total amount of time for a handset to send a 140-byte SMS packet (1.75+0.22+4.43) ~ (1.75+0.28+6.39)	6.4 ~ 8.42

As can be seen, the bulk of the time is in transmitting the SMS packet onto the wireless network from the handset no matter how small the size of the packet. Therefore, it is obvious the priority is to minimize the number of SMS messages used by the protocols rather than minimizing the packet size.

Another factor affecting the execution time of the protocol is the processing delay of the second slot smart card. Since the processing speed of the processor chip on the smart card chip is very slow compared with that of a PC, it may induce a significant delay on protocol execution. Again the processing time is highly dependent on the smart card product. We have compared three brands of Java Card with cryptographic function (see Table 5).

Table 5. Timing for smart card operations

Operation	**Time (in sec)**		
Invocation time of the second slot card by the handset	1.3		
Brand of Java Card	**A**	**B**	**C**
RSA Private Key Decryption (Key length = 1024 bits)	0.58	0.43	0.14
RSA Public Key Encryption (Key length = 1024 bits)	0.19	0.10	0.02
SHA-1 hash on 64 bytes of data	0.16	0.29	0.08
Triple-DES encryption in CBC mode on 64 bytes of data	0.28	0.04	0.008
Triple-DES encryption in ECB mode on 64 bytes of data	0.21	0.04	0.008
Random number generation (64 bytes)	0.43	0.007	0.008

For the *UAS Session Establishment* protocol, the overall processing time spent by the mobile client is about 18~22s (two SMS messages and four second slot card invocations with the associated processing time on the smart card). On the other hand, the processing time incurred by the UAS, which is implemented on a Pentium III 550MHz PC, is only about 0.1s and is negligible compared to that of the mobile client. The total time to establish a UAS session is the sum of the processing times and the queuing delay at the SMS gateway which varies from time to time depending on the traffic intensity. Since the major part of the processing time is spent on SMS transmission and invocation of the second slot card, the delay time for the *PESM Session Establishment* protocol is roughly the same as that for the *UAS Session Establishment protocol.*

We can see that performing an AKC protocol on the GSM mobile wireless is unacceptable for time critical applications. If the mobile phone starts with the *UAS Session Establishment* protocol and then the *PESM Session Establishment* protocol and lastly the application data, it would likely take about 1 minute to finish the process. A way to reduce this time is to reuse the session keys which have been previously established. In our protocol, both the UAS session key and PESM session key are kept for a period of time determined by the *KeyPolicy* value assigned by the UAS and PESM. Normally, a session key can be used for hundreds of times before refreshment since the time needed to break a 128-bit session key using the traditional method is of the order of tens of years. In this case, the AKC protocols will not need to be executed most of the time before a transaction. The application data is directly encrypted by the established session key stored in memory on the second slot smart card and transmitted. The time needed for the transaction is about 9 seconds if no digital signature is required or 18 seconds otherwise.

8. CONCLUSIONS

In this paper AKC protocols are categorized according to their authentication and key exchange approaches. We then presented a new set of AKC protocols that are practical and optimized for the GSM wireless network with SMS as the carrier. The protocols involve three parties - the mobile client, the service provider, and a trusted third party- and provide PKI-based end-to-end security between the client and the service provider. It is one of the few systems that has been implemented and an analysis of its performance has been given.

To evaluate the applicability of the protocols to electronic commerce activities, we have implemented a payment application on the infrastructure. The payment system makes use of the secure features of the protocols to protect the credit card number transmitted from the second slot smart card on the mobile client to the payment server of the merchant's acquirer bank. The application is being used in real-life for purchase and payment. The customers have found the system convenient to use and practical. The performance is acceptable and so far no security problem has been reported.

REFERENCES

[1] Secure Socket Layer (SSL) Version 3.0, http://home.netscape.com/eng/s13

[2] "Wireless Application Protocol Architecture Specification Version 30-Apr-1998", Wireless Application Protocol Forum, Ltd., http://www.wapforum.org

[3] Java Card API 2.1, Sun Microsystems Inc, http://java.sun.com/products/javacard

[4] Hongkong Post e-Cert, http://www.hongkongpost.gov.hk

[5] "Wireless Transaction Protocol Specification Version 05-Nov-1999", Wireless Application Protocol Forum,Ltd., http://www.wapforum.org

[6] "Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM-ME) Interface (GSM 11.14 ver 7.1.0 Release 1998)", European Telecommunications Standards Institute

[7] T. Dierks and C. Allen, "The TLS Protocols Version 1.0", RFC 2246, 1999, ftp://ftp.isi.edu/in-notes/rfc2246.txt

[8] "Wireless Application Protocol Wireless Transport Layer Security Specification Version 18-Feb-2000" Wireless Application Protocol Forum, Ltd., http://www.wapforum.org

[9] S. Blake-Wilson, D. Johnson and A. Menezes, "Key Agreement Protocols and their Security Analysis", Sixth IMA International Conference on Cryptography and Coding, December 1997.

[10] K. H. Lee and S. J. Moon, "AKA Protocols for Mobile Communications", Proceedings of the 5th Australasian Conference on Information Security and Privacy, ACISP 2000, pp. 400-411, 2000

[11] W. Diffie and M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, IT-22(6): pp. 644-654, November 1976

[12] M. J. Beller, L. -F. Chang and Y. Yacobi, "Privacy and Authentication on a Portable Communication System", IEEE Journal on Selected Areas in Communications, vol. 11, pp.82l-829, August 1993.

[13] U. Carlsen, “Optimal Privacy and Authentication on a Portable Communications System”, ACM Operating Systems Review, 28(3), 1994, pp. 16-23

[14] Y. Zheng, “An Authentication and Security Protocol for Mobile Computing”, Proceedings of IFIP, pp. 249-257, Sep. 1996.

[15] Aziz and W. Diffie, “Privacy and Authentication for Wireless Local Area Networks”, IEEE Personal Communications, vol. 1, pp. 25-31, 1994

[16] H. Lim, P. J. Lee, “Several practical protocols for authentication and key exchange”, Information Processing Letters, vol 53, pp. 91-96, 1995

[17] H.-Y. Lin and L. Harn, “Authentication Protocols with Nonrepudiation Services in Personal Communication Systems”, IEEE Communications Letters, vol. 3, no. 8, pp. 236-238, Aug. 1999

[18] “GSM System Security Study”, RACAL Research Ltd, http://jya.com/gsm061088.htm, 1998

[19] R. Needham and M. Schroeder, “Using Encryption for Authentication in Large Networks of Computers”, Communications of the ACM, 21(12), December 1978.

[20] J. Clark and J. Jacob, “A Survey of Authentication Protocol Literature: Version 1.0”, http://www.cs.york.ac.uk/jac/papers/drareview.ps.gz , 17 November, 1997

[21] “Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM-ME) interface (3GPP TS 11.14 version 8.5.0)”, ETSI, 1999