Securing Instant Messaging Communications Using Limited-Used Session Keys

Supakorn Kungpisdan and Nitoon Moonviriyakit

Faculty of Information Science and Information Technology, Mahanakorn University of Technology
Bangkok, 10530, Thailand, supakorn@mut.ac.th

ABSTRACT – Instant Messaging (or IM for short) is becoming one of the most popular applications on the Internet. Individuals can communicate instantly through a user-friendly graphical user interface over fixed and wireless devices. However, several security issues occur, majorly privacy and authentication. A number of secure IM protocols have been proposed, but they still lack of necessary security properties and acceptable performance. In this paper, we introduce a new secure instant messaging protocol that not only satisfies necessary security properties. Moreover, our analysis shows that the proposed protocol has better transaction performance than existing protocols.

KEY WORDS – Instant messaging, information security, cryptography, cryptographic protocols, network security

1. Introduction

Instant Messaging (or IM for short) has become one of the most popular Internet-based applications due to its simple way to use to communicate among individuals. Several kinds of personal communications devices in today's markets such as mobile phones, laptops, or netbooks come with built-in IM applications. Nowadays, not only people use IM services for personal use, it also becomes one of the major communications methods in workplaces.

Unfortunately, the growing number of IM applications as well as IM users comes with several security issues. Firstly, IM applications transmit data in cleartex, but people still are not aware of sending confidential information through IM communications channels. This leads to several kinds of attacks, including sniffing and stealing confidential information. Secondly, sniffing the IM conversations leads to privacy problems. IM conversation can be easily monitored, especially in LAN-based environment using freely available packet sniffer programs such as Wireshark [1] or TCPdump [2]. Moreover, IM conversation cannot be authenticated. Thus, IM users can be victims of spoofing data by pretending to be a trusted user.

Clearly, an IM application that provides secure communications among IM users is required. Several secure IM protocols have been proposed. Mannan *et al.* [3] proposed a secure IM protocol based on the combination between symmetric and public-key cryptographic operations. The protocol ensures data confidentiality as it encrypts IM conversation using symmetric encryption and distributes session keys using public-key based session key distribution. However, the protocol is heavy-weight as a number of public-key cryptographic operations are required in each transaction. Moreover, Mannan *et al.* proposed a technique to update session keys. However, such keys need to be transmitted over the network that is possible to be intercepted by attackers. Furthermore, the authors did not mention about updating a session key between IM users after being used for a certain period of time. It can be argued that using the same session key for a long period of time increases an opportunity for an attacker to analyze the encrypted traffic and retrieve the session key. Yang *et al.* [4] proposed an elliptic-curve-based IM protocol by using elliptic-curve cryptography to create public keys and symmetric keys. However, the authors did not mention about key update process that leads to the same problem as that of Mannan *et al.*'s approach.

To overcome the above problems and limitations, we propose a new secure instant messaging protocol that improves several security properties and performance compared to the existing approaches [3, 4]. The proposed protocol ensures data confidentiality as well as the ability to identify the message sender. We solve the problem of reusing session keys by using each session key only once. We apply an offline session key generation and distribution technique to overcome the problem of transmitting updated session keys over the network. Each communicating party can generate a set of session keys shared with

another party on his/her local host without transmitting such keys over the network. As the new session keys are not transmitted over the network, it cannot be intercepted. The proposed protocol is based on the combination of symmetric and public-key operations, but it has better performance than the existing protocols.

This paper is organized as follows. Section 2 provides an overview of instant messaging applications. In section 3, we discuss problems and limitations of existing secure IM protocols. Section 4 describes an offline session key generation and distribution that is used in this paper. In section 5, our proposed secure IM protocol is introduced. Sections 6 and 7 analyze security and performance of the proposed protocol compared with existing protocols [3, 4]. Section 8 concludes the paper.

2. Overview of Instant Messaging Applications

Instant Messaging (or IM for short) is an Internet-based application whereby individuals can communicate mainly in a text-based conversation although nowadays, many IM applications offer the ability to send voice, graphics, or other file formats though them. People can register anonymously through an IM application and start inviting friends, chatting and conferencing through text-based conversation.

A general IM system is composed of three parties: a user *A*, a user *B*, and an IM server *S* as shown in Figure 1 below.

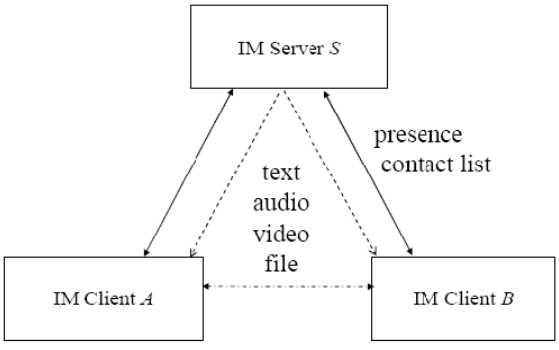

Figure 1: General instant messaging system

According to the above system, the users *A* and *B* need to register the IM service with the server *S*. After given a username and a password, each user can login to *S* and needs to maintain its presence with *S*. Each user can see the presence of others through his/her contact list. If *A* wants to communicate with *B*, *A* simply sends a request to communicate with *B* (aka an invitation to *B*) to *S*. *S* then sends this invitation from *A* to *B*. Then, *A* and *B* can establish direct communications between them. It can be seen that IM is a hybrid application in that IM users, *A* and *B*, need to establish client-server communications with the IM server *S*, whereas they communicate among them in a peer-to-peer manner.

In terms of security, it can be seen that identity and user privacy is not major security issue for general IM applications as we can see that an individual can register for an IM service for free. The provider only needs an email address to identify the user. Furthermore, IM messages are transmitted in cleartext. They can be easily intercepted by an attacker.

Nowadays, IM applications are receiving more acceptance as a communications method within an organization. Employees can collaborate or communicate with business partners through this channel. That means, several security issues needs to be discussed while designing a protocol for IM. As well as other Internet-based security protocols, in general, a secure IM protocol requires the following security properties:

- *Message Confidentiality*: messages transmitted though an IM protocol can be revealed only to an authorized party, which is an intended recipient.
- *Message Integrity*: IM messages should not be modified during the transmission.
- *Message authentication*: an IM message should contain evidence to identify its sender and intended recipient.
- *Non-repudiation of transactions*: the sender should not be able to deny the transaction made by him/her.

In addition to the security aspects discussed above, an IM protocol is considered acceptable if it has acceptable performance. The term acceptable performance means that each IM message should be transmitted from a sender to a recipient within limited time which is acceptable by both communicating parties. This concern is crucial as a protocol can be secure by implementing several security technologies, especially cryptographic techniques, to it, but this significantly adds overhead, especially delays, to messages transmitted in the protocol. One possible solution to this is to select appropriate security techniques that provide certain level of security while maintaining acceptable transaction performance. This approach is taken as the main focus of this paper. That is we intend to develop a secure IM protocol that satisfies both security and performance.

3. Existing Approaches to Secure Instant Messaging

A number of secure IM protocols have been proposed [3, 4]. This section discusses Mannan *et al.*'s protocol [3] and Yang *et al.*'s protocol [4] by focusing on their security and performance problems.

Mannan *et al.* [3] introduced an IM protocol called Instant Message Key Exchange (IMKE) that contains both session key exchange and message transmission. This protocol is composed of 3 phases: *PAKE and client-server communications*, *Public-key distribution*, and *Session key transport*. The idea of this protocol is that, while logging in to the system, a user *A* and a user *B* generate their own temporary public-private key pairs. Then they submit their public keys together with shared keys K_{AS} and K_{BS} to a server *S*. Note that the key K_{AS} is shared between *A* and *S* and the key K_{BS} is shared between *B* and *S*. Note also that K_{AS} (or K_{BS}) is used to generate a session key for encryption K_{AS}^{E} (or K_{BS}^{E}) and a session key for message authentication code (MAC) K_{AS}^{M} (or K_{BS}^{M}) to secure the communications between the users and *S*. If *A* wants to communicate with *B*, *A* sends a request to *S* asking for *B*'s public key. The server *S* then sends *A* *B*'s public key and sends *B* *A*'s public key. After receiving the public keys, *A* creates K_{AB}, a session key to be shared with *B*, encrypts K_{AB} with *B*'s public key and sends it to *B* in a peer-to-peer manner. Then, both and *A* and *B* create an encryption key K_{AB}^{E} and a MAC key K_{AB}^{M} from K_{AB}. After that, both *A* and *B* can communicate securely as follows:

A → B: $\{A\text{'sMessage}\}_{KAB}^{E}$, $MAC(A\text{'sMessage}, K_{AB}^{M})$
B → A: $\{B\text{'sMessage}\}_{KAB}^{E}$, $MAC(B\text{'sMessage}, K_{AB}^{M})$

Mannan *et al.* also proposed a mechanism to update a session key shared between *A* and *S* after the key K_{AS} is used for a certain period of time as follows:

A → S: $\{\{K_{AS1}\}_{Pub\text{-}S}\}_{KAS}^{E}$, $MAC(\{K_{AS1}\}_{Pub\text{-}S}, K_{AS}^{M})$
S → A: $\{\{K_{AS1}\}_{Pub\text{-}A}\}_{KAS}^{E}$, $MAC(\{K_{AS1}\}_{Pub\text{-}A}, K_{AS}^{M})$

Where $\{M\}_{Pub\text{-}R}$ stands for a message *M* encrypted with a user *R*'s public key, $\{M\}_{K}$ stands for a message *M* symmetrically encrypted with a shared key *K*, and *MAC(M, K)* stands for a MAC value of a message *M* with a shared key *K*.

However, Mannan *et al.* did not mention about how to update a shared key between users. It may possibly be assumed that the users *A* and *B* update the shared key K_{AB} by using the same method. According to the above assumptions, it can be argued that, although Mannan *et al.* proposed the key update method, but the new key requires to be transmitted over the network. That means the keys can be intercepted by an attacker. If the key K_{AS1} is intercepted, the attacker can easily impersonate as *A* and communicate with other users. Thus, a new session key generation and distribution is needed.

In addition, Yang *et al.* [4] proposed an elliptic-curve-based IM protocol by using elliptic-curve cryptography (ECC) to create public keys and symmetric keys. Employing ECC in their protocol allows better performance. As encryption using ECC with 160-bit key can be as strong as RSA encryption with 1024-bit key. Thus using the smaller size of encryption key leads to faster cryptographic operations. However, Yang *et al.* did not mention about key update mechanism that leads to the same problem as Mannan *et al.*'s approach [3].

In this paper, we apply an offline session key generation and distribution technique that eliminates the need to transmit the key itself over the network to the proposed IM protocol. This reduces an opportunity of an attacker to intercept the key. Moreover, we show that our proposed protocol is more lightweight compared to IMKE. This leads to better transaction performance compare to existing IM protocols [3, 4].

4. Session key generation and distribution for instant messaging transactions

Session key generation and distribution is one of the most widely discussed topics in symmetric cryptography. This is because a secret key needs to be shared between engaging parties in a secure manner. A number of session key generation and distribution techniques have been proposed [5, 6, 7, 8, 9, 10]. Among these techniques, they can be classified in two types: *online* and *offline* techniques. On one hand, online session key generation and distribution techniques require a new session key to be transmitted over the network. Although it is transmitted in an encrypted format, it is possible that the key can be compromised. On the other hand, in an offline session key generation and distribution technique, a new session key is not necessary to be transmitted in the network. Thus an attacker is not able to capture the session key on the wire.

According to the above discussion, an offline session key generation and distribution was chosen to secure our proposed secure instant messaging protocol. Several offline session key generation techniques have been proposed [5, 6, 7, 8, 9, 10]. Among them, Kungpisdan *et al.* [7] introduced a session key generation technique that not only it is secure against key compromise attacks, but it can also operate

purely offline. According to this technique, the longer the technique is used, the more secure the internet transaction will be. Kungpisdan *et al.* argued that this technique can be applied to any kind of transactions.

Kungpisdan et al.'s Approach

Assume that Alice and Bob share $\{K_{AB}, DK, m\}$, where K_{AB} is a long-term key, DK is called a *distributed* key, and m is a random number. m is used to specify the number of keys that will be generated. m also varies randomly among different pairs of parties. $conc(M_1, M_2, M_3)$ represents the concatenation of the message M_1, M_2, and M_3, respectively. Then, the key generation process is shown in Figure 2.

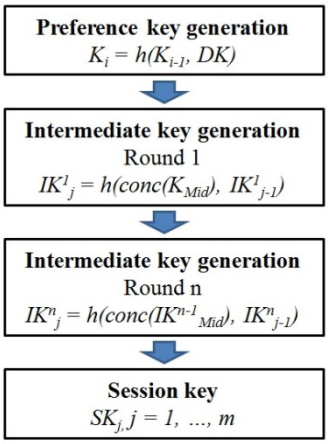

Figure 2: Session Key Generation

After sharing $\{K_{AB}, DK, m\}$, Alice and Bob generate a set of *preference keys* K_i, where $i = 1, \ldots, m$, as follows: $\boldsymbol{K_i = h(K_{i-1}, DK)}$, where $K_0 = K_{AB}$. The set of K_i will be used as a source to regenerate session keys if needed. After generating the set of K_i, K_{AB} and DK can be removed from the system.

Then both Alice and Bob create sets of *intermediate keys* in order to increase the difficulty for cryptanalysis. In other words, it increases difficulty to trace back to the preference key if the session key is compromised. In each round, a new set of intermediate keys is created. The higher number of round is performed, the greater security the system is. The intermediate key generation is performed as follows: $\boldsymbol{IK^x_j = h(conc(IK^{x-1}_{Mid}), IK^x_{j-1})}$, where x specifies the round number, j specifies the number of intermediate keys that is generated, $j = 1, \ldots, m$. IK^{x-1}_{Mid} stands for the set of $\{IK^{x-1}_{Mid1}, IK^{x-1}_{Mid2}, IK^{x-1}_{Mid3}\}$. $IK^x_{Mid1} = mid(IK^x_1, IK^x_{rm})$ and rm is the remaining number of intermediate keys in the set of IK^x_j. $IK^x_{Mid2} = mid(IK^x_{Mid1}, IK^x_{rm})$. $IK^x_{Mid3} = mid(IK^x_1, IK^x_{Mid2})$. $IK^1_{Mid1} = K_{Mid1}$, $IK^1_{Mid1} = K_{Mid2}$, and $IK^1_{Mid1} = K_{Mid3}$. The generation of K_{Mid1}, K_{Mid2}, and K_{Mid3} is the same as that of IK^x_{Mid1}, IK^x_{Mid2}, IK^x_{Mid3}, respectively. $IK^x_{j-1} = \phi$. The output of the last round of intermediate key generation is considered as session keys SK_j, where $j = 1, \ldots, m$, which is shown below: $IK^n_1 = SK_1$, $IK^n_2 = SK_2, \ldots, IK^n_m = SK_m$. Alice and Bob then can use SK_j as a credential to secure transactions e.g. as an encryption key or as an input to message authentication code.

It can be clearly seen that the session key was generated purely offline. Each engaging party can create a set of session keys used to secure communications between them without the needs to transfer credentials over the network. As a new session key is not transferred over the network. Thus, it will never be intercepted. Thus, this technique is expected to increase security of symmetric-key cryptosystems including the proposed secure instant messaging protocol.

5. The Proposed Protocol

In this section, we introduce a new **S**ecure **I**nstant **M**essaging **P**rotocol (called SIMP) that overcomes the problems and limitations of existing secure IM protocols [3, 4]. The following notations are defined for the proposed protocol:

- $\{A, B\}$ is the set of communicating users, whereas S denotes an instant messaging server.
- ID_A is the identity of A.
- $\{DK, K_{AS}, m\}$ is the of key distribution parameters of session key generation and distribution.
- P_A is a password shared between A and S.
- SK_{ABj}, where $j = 1, \ldots, m$, stand for session keys shared between the users A and B.
- n is a nonce to prevent replay.
- $\{m\}_K$ is a symmetrically encrypted message of a message m with a key K.
- $h(m)$ is a hash value of a message m.
- $h(m, K)$ is hashed message authentication code (MAC) of a message m and a key K.
- $\{Pub\text{-}A, Pri\text{-}A\}$ is the set of public and private keys of A, respectively.
- $\{m\}_{Pub\text{-}A}$ is a message m encrypted with a public key of a user A.
- $\{m\}_{Pri\text{-}A}$ is a message m signed with a private key of a user A.

The proposed protocol is composed of 3 sub-protocols as show below:

Registration Protocol

In this protocol, every user, including A and B, needs to register to the IM server S for the first time. S transmits the necessary information to A as follows:

R1: A → S: ID_A, P_A
R2: S → A: $DK_{AS}, K_{AS}, m, h(ID_A, P_A, DK_{AS}, K_{AS}, m_1)$

The above protocol is performed over a secure SSL tunnel. *S* stores $h(ID_A, P_A)$ in its database for user authentication. After receiving $\{DK_{AS}, K_{AS}, m_1\}$, *A* and *S* generate a set of session keys SK_j, where $j = 1, \ldots, m$, according to the session key generation technique presented in the previous section. Each session key will be later used for encryption in the next section.

User Login Protocol

After the registration is successful, *A* creates one-time private key *Pri-A*, public key *Pub-A*, and a nonce n_1. Then, *A* logins to *S* as follows:

L1: A → S: $\{h(ID_A, P_A, n_1), n_1, Pub\text{-}A\}_{SKj}$

After receiving the above message, *S* decrypts the message and compares the hashed password $h(ID_A, P_A, n_1)$ with the one it has. If they are matched, *S* can ensure that *A* originates this request. To authenticate the user *A*, *S* responses with the following message:

L2: S → A: $\{n_2, h(n_1, n_2)\}_{SKj}$

A decrypts the message by using SK_j to retrieve n_2. *A* calculates the hashed value $h(n_1, n_2)$ and compared with the received hash value. If they are matched, *A* successfully logins to *S*. Note that every user, including *B*, performs exactly the same steps to login to *S*.

Key Exchange and User-to-User Communications Protocol

After every user logins to the IM server, *S* stores public keys of each user. When a user wants to communicate with another user, he/she can perform the following steps:

E1: A → S: $Request, ID_B$
E2: S →A: $\{ID_B, Pub\text{-}B, h(SK_{BSj})\}_{SKASj}$
E3: S → B: $\{ID_A, Pub\text{-}A, h(SK_{ASj})\}_{SKBSj}$

It can be seen that both *A* and *B* receive the public keys of each other. Moreover, adding $h(SK_{BSj})$ to the message sent to *A* and adding $h(SK_{ASj})$ in the step **E3** prevent *A* and *B* from generating the above messages by themselves. Then *A* sends the following message to *B*:

E4: A → B: $\{n_3, K_{AB}, DK_{AB}, m_2\}_{Pub\text{-}B}$, $\{h(K_{AB}, DK_{AB}, m_2)\}_{Pri\text{-}A}$

It can be seen that *B*, previously received *Pub-A* from **E3**, is able to identify the sender of the message, that is *A*, from $\{h(K_{AB}, DK_{AB}, m_2)\}_{Pri\text{-}A}$. *B* can retrieve $\{K_{AB}, DK_{AB}, m_2\}$ from the message encrypted with *B*'s public key. Then both *A* and *B* can generate a set of session keys SK_{ABj}, where $j = 1, \ldots, m_2$, by using the key generation and distribution technique discussed in section 4. In order to ensure that *A* can generate the same set of session keys, *B* sends the following message:

E5: B → A: $h(n_3, SK_{ABj})$

A can compare the above hash value as he/she knows SK_{ABj}. At this stage, both *A* and *B* share the same set of session keys and then can use these keys to encrypt messages sent between them as follow:

E6: A → B: $\{A's\_Message, n_4\}_{SKABj}$, $h(n_4, A's\_Message, SK_{ABj})$
E7: B → A: $\{B's\_Message, n_5\}_{SKABj}$, $h(n_5, B's\_Message, SK_{ABj})$

The message exchanges between *A* and *B* are encrypted using the session keys shared between them. Also, the MAC attached to these messages ensures integrity of the messages. Also, the nonce n_4 and n_5 ensure that the messages are fresh.

6. Security Analysis

In this section, we analyze important security for IM transactions as follows: message authentication, message confidentiality, non-repudiation of transactions, and message integrity.

6.1 Message Authentication

Message authentication ensures the originator of a message. The proposed protocol ensures user authentication in that only the user who possesses a shared key can communicate with the other user. Consider the message in step **E2**:

E2: S →A: $\{ID_B, Pub\text{-}B, h(SK_{BSj})\}_{SKASj}$

It can be seen that *S* and *A* share the session key SK_{ASj}, but *A* cannot generate this message by himself/herself because *A* cannot generate $h(SK_{BSj})$ in that the session key SK_{BSj} is shared between *B* and *S*. Only *S* knows both SK_{ASj} and SK_{BSj}; thus this guarantees that *S* generated this message.

6.2 Message Confidentiality

Confidentiality ensures that the data can only be revealed to authorized parties. The proposed protocol

satisfies the confidentiality by applying symmetric and asymmetric encryption to messages in each step.

6.3 Non-repudiation of Transactions

Non-repudiation of transactions is the property that we can ensure that each party cannot deny the transaction he/she has performed. Generally, non-repudiation can be achieved by digital signature to a message. However, a symmetric-cryptographic message can satisfy non-repudiation property by collecting evidence from each message that demonstrates the actions that each party has performed in each transaction. To illustrate that our proposed protocol satisfies non-repudiation property, consider the message in the step **E2:**

E2: **S →A:** $\{ID_B, Pub\text{-}B, h(SK_{BSj})\}_{SKASj}$

It can be seen that *S* cannot deny that it did not originate this message as the possession of both SK_{ASj} and SK_{BSj} demonstrates clearly that only *S* can generate this message.

6.4 Message Integrity

Message integrity is the property that ensures that the message is not modified during the transactions. Moreover, if the message is modified, the receiver should be able to prove the integrity of the message. It can be seen that, in the proposed protocol, each message contains MAC value that guarantees that the recipient of the message can compare to see if the received message can generate the same MAC value as that was transmitted within the message.

7. Performance Analysis

In order to show that the proposed protocol has better performance than existing protocols, we compare performance of the proposed protocol with Mannan *et al.*'s protocol [3] and Yang *et al.*'s protocol [4] by comparing the number of cryptographic operations applied to each protocol techniques. Table 1 shows our comparison.

From the table 1, it can be seen that the proposed protocol has better performance comparing to other protocols due to lower number of cryptographic operations. Moreover, the proposed protocol deploys lightweight cryptographic operations e.g. symmetric-key and MAC operations, whereas IMKE and SIMPP rely heavily on public-key cryptographic operations. Note that lower number of cryptographic operations and lightweight operations lead to better transaction performance to a security protocol.

Note also that, even though the proposed protocol relies heavily on symmetric cryptographic operations that are generally known that their security is weaker than the public-key ones. However, the proposed protocol overcomes this limitation by not reusing session keys. This helps strengthen the IM transactions.

Table 1: The number of cryptographic operations applied to SIMP, IMKE, and SIMPP, respectively

Cryptographic Operation		SIMP	IMKE	SIMPP
Public-key Encryption	S	-	1	-
	C1	1	2	-
	C2	1	2	-
Public-key Decryption	S	-	1	-
	C1	1	2	-
	C2	1	2	-
Signature generation	S	-	-	-
	C1	1	-	2
	C2	1	-	2
Signature verification	S	-	-	2
	C1	1	-	1
	C2	1	-	1
Symmetric operations	S	6	3	10
	C1	4	6	7
	C2	4	6	7
Hash operations	S	2	5	6
	C1	1	9	3
	C2	1	9	3
Keyed-hash operations	S	4	-	-
	C1	2	-	-
	C2	2		-
MAC operations	S	2	1	2
	C1	3	2	3
	C2	3	2	3

8. Conclusion

In this paper, we addressed problems and limitations of existing secure instant messaging protocols, IMKE [3] and SIMPP [4], by focusing on several important security properties for Internet transactions. We found that both of the existing IM protocols lack of the ability to perform secure session key update which is one of the most crucial mechanisms concerned in our context. Then, we introduce SIMP, a new Secure Instant Messaging Protocol, that not only provides secure communications among engaging parties, but it also offers better security properties and transaction performance compared to the existing protocols.

As our future works, we plan to implement the proposed protocol to demonstrate its capability to perform as a real-world application. Moreover, we intend to develop a peer-to-peer payment system based on IM applications. This would allow individuals to purchase products or services, or transfer money though an IM application.

References

[1] Wireshark, http://www.wireshark.org, last access February, 2010.

[2] TCPdump, http://www.tcpdump.org, last access February, 2010.

[3] M. Mannan and P. C. van Oorschot, "Secure Public Instant Messaging", Proceedings of Financial Cryptography and Data Security 2006.

[4] C. H. Yang, T. Y. Kuo, T Ahn, and C. P. Lee, "Design and Implementation of a Secure Instant Messaging Based on Elliptic-Curve Cryptography", Journal of Computers, Vol. 18(4), January 2008.

[5] O. Dandash *et al.*, "Fraudulent Internet Banking Payments Prevention using Dynamic Key," Journal of Networks, Vol.3(1), Academy Publisher, pp. 25-34, 2008

[6] S. Kungpisdan, P.D. Le, and B. Srinivasan, "A Limited-Used Key Generation Scheme for Internet Transactions", Lecture Notes in Computer Science, Vol. 3325, 2005.

[7] S. Kungpisdan and S. Metheekul, "A Secure Offline Key Generation With Protection Against Key Compromise" in the World Multi-conference on Systematics, Cybernetics, and Informatics 2009, Orlando, Florida, July. 2009, pp. 63-67.

[8] Li, Y. and Zhang, X., 2004. A Security-enhanced One-time Payment Scheme for Credit Card. *Proc. of the Int'l Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications*,

[9] S. Kungpisdan, B. Srinivasan, and P.D. Le, "Lightweight Mobile Credit-card Payment Protocol", Lecture Notes in Computer Science, Vol. 2904, pp. 295-308, 2003.

[10] A. D. Rubin and R.N. Wright, "Off-line Generation of Limited-Use Credit Card Numbers", Lecture Notes in Computer Science, Vol. 2339, pp. 196-209, 2002.