The Design and Implementation of a Secure Instant Messaging Key Exchange Protocol

Chung-Huang Yang * Tzong-Yih Kuo **

* Institute of Information and Computer Education, NKNU Professor chyang@nknucc.nknu.edu.tw

** Institute of Information and Computer Education, NKNU Student dato@icemail.nknu.edu.tw

Abstract

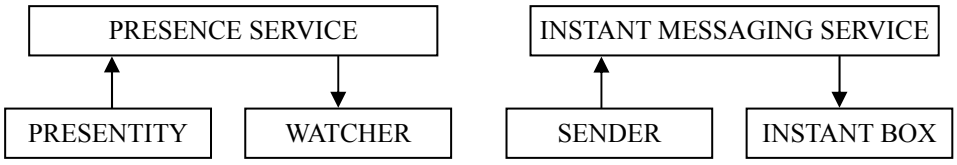
IM (Instant Messaging) is a useful communication and work coordination tool between individuals, groups, or businesses. Unfortunately, majority of the IM systems lack the needed perfect security mechanism capable of ensuring the security of Client-Client and Client-Server communications; in short, Confidentiality, Integrity and Authentication are not assured. In order to find a solution to this security problem, this paper designs the SIMPP (Secure Instant Messaging and Presence Protocol) based on a three-party PAKE (Password Authentication Key Exchange), thus ensuring the key exchange and communication security. Aside from using a password authentication and key exchange, the SIMPP offers the option of using a smart card instead of the password. In reality, the open source of the SIMPP has been amended to make it compatible with the XMPP (eXtensible Messaging and Presence Protocol)/Jabber Standard, the jabberd 2.0s10, wherein the cryptography libraries MIRACL 5.21 and the OpenSSL 0.9.8a implements the security function to resolve the IM security problem.

Keywords: Secure Instant Messaging, Smart Card, Key Exchange

The Design and Implementation of a Secure Instant Messaging Key Exchange Protocol

1. Introduction

Majority of the communications under the IM system are not secure [7][8]. For instance, in the MSN Messenger, Yahoo! Messenger, ICQ, and AOL environments, any user that has logged into the system may communicate in plaintext [1], communications are not properly protected. In year 2000, IETF proposed that the RFC 2778 defined the IM system to be composed of two types of services, which are the Presence Service and the Instant Messaging Service, as shown in Figure 1 for details [5].



(a) Presence Service Model (b) Instant Messaging Service Model

Figure 1: RFC 2778 Definition of the Two Types of IM System Service Models

Under the IM service models shown in Figure 1, communications between clients should pass through the Server. If the system could only assure Client-Server communication security, it would overlook the privacy and security considerations of the message in the Server after message has been transmitted between clients and from the Client to the Server [2]. One security problem is the means by which the system may assure the Confidentiality, Integrity and Authentication of all Client-Server and Client-Client communications. In solving this problem, the paper designed and implemented a secure IM protocol — SIMPP (Secure Instant Messaging and Presence Protocol), thus enhancing the key exchange and communication security of the IM system. In the XMPP/Jabber [5] protocol of the RFC 3920-3923 and JEPs (Jabber Enhancement Proposals) norm, we chose to use the ECC (Elliptic Curve Cryptosystem) [4] to implement the SIMPP.

2. Secure IM Mechanisms

In order to avoid administrator eavesdropping into Client-Client communication, Kikuchi et al designed a secure IM protocol founded on the Diffie-Hellman principle [2]. The [2] design is divided into two phases: (1) Registration; (2) Key Establishment in Peer-to-Peer and Key Establishment in Chat; as shown in Figure 2. As shown in the diagram, the third step in the Registration phase is Public Key Distribution [2], which is equivalent to the Public Key Distribution phase of IMKE [6] (Figure 3). In terms of efficiency, the problem with [2] is that when too many users frequently run the Registration and Key Establishment in Peer-to-Peer phases, it burdens the Server modulus computation.

M. Mannan et al designed the IMKE (Instant Messaging Key Exchange) protocol [6] to

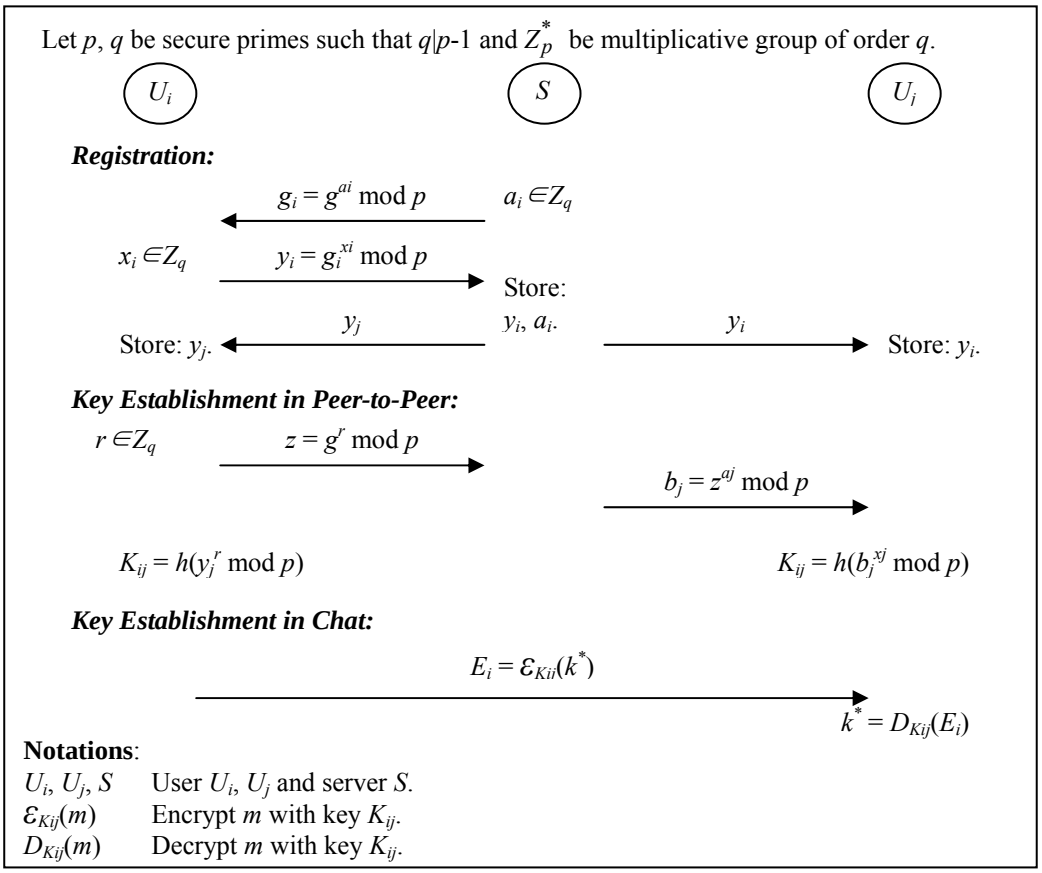


Figure 2: The Secure IM Protocol proposed by literati Kikuchi et al. [2]

ensure the Confidentiality, Integrity and Authentication of Client-Server and Client-Client communications. The IMKE designing process was conducted in three phases: (1) PAKE, (2) Public Key Distribution, and (3) Session Key Transport; the respective protocols are as shown in Figure 3. In terms of security, the IMKE has a better security than the Kikuchi.

3. SIMPP Security Objective

The paper sets the SIMPP security objective on the satisfaction of Client-Server and Client-Client communication security [6]:

G1. Even if an attacker manages to obtain the authentication data stored in the Server, attacker would not be able to launch an impersonation attack.

G2. The PAKE process is not susceptible to offline dictionary attack and guessing attack.

G3. The PAKE process blocks off replay attacks.

G4. Confidentiality, Integrity and Authentication of Client-Server communication are assured.

G5. Confidentiality, Integrity and Authentication of Client-Client communication are assured.

G6. The SIMPP establishes Session-Key Security, Known-Key Security, Mutual Authentication.

4. Design SIMPP

The SIMPP contains three phases: (1) Registration (§4.1); (2) PAKE and Client-Server Communication (§4.2); and (3) Client-Client Key Exchange and Communication (§4.3). The

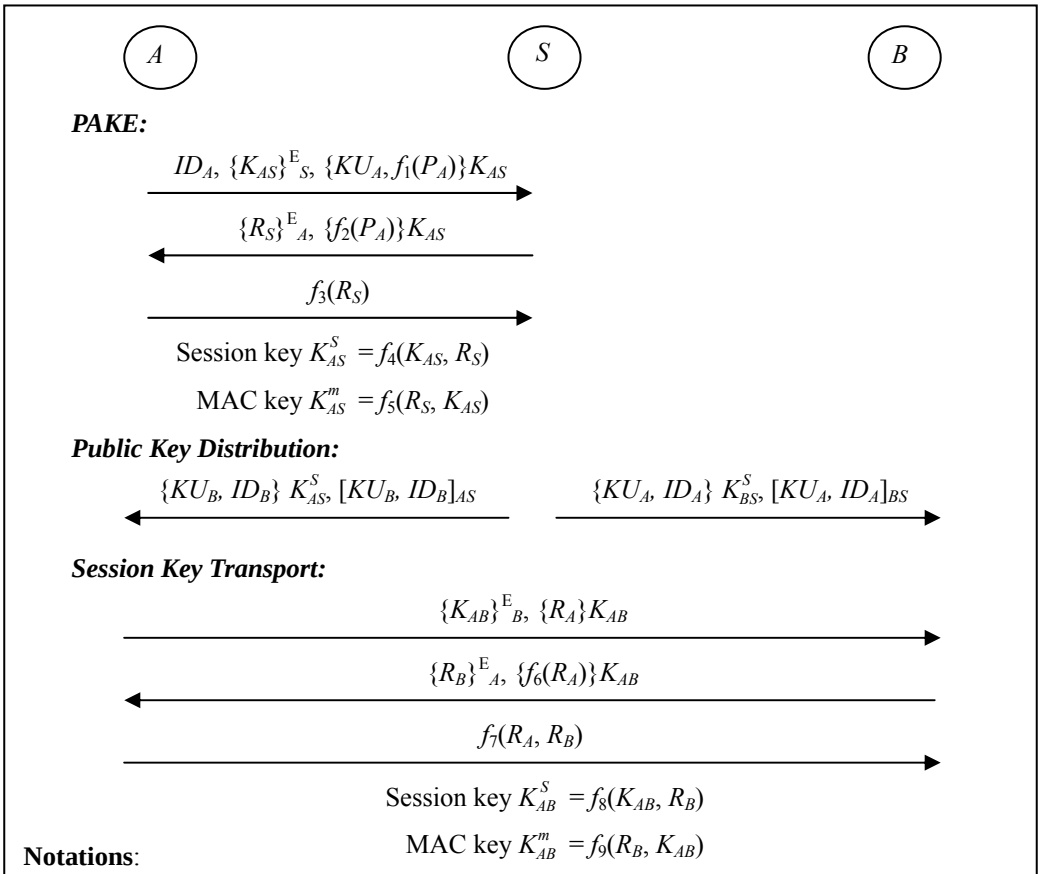


Figure 3: The IMKE Protocol proposed by literati Mannan et al. [6]

Table 1: Definition of the notations used in Design SIMPP

A, B, S	User A, B and server S.
ID_A, pw_A	A's ID and password.
R_S	Random string generated by S.
KU_A, KR_A	A's public key KU_A and private key KR_A for long-term.
KU_A', KR_A'	A's public key KU_A' and private key KR_A' for short-term.
$\{data\}K_{AS}$	Encrypt *data* with symmetric key K_{AS}.
$\{data\}^{E}_{S}$	Encrypt *data* with S's public key KU_S.
$[data]^{S}_{A}$	Sign digital signature of *data* with A's private key KR_A.
$[data]K_{AS}$	MAC(Message Authentication Code) of *data* with key K_{AS}.
h	One-way cryptographic hash functions such as SHA-256.

protocol design uses the notations and definitions shown in Table 1. The paper uses the ECC [4] to implement SIMPP. We set G as the ECC point of the SIMPP. The Server selects the big integer $KR_S \geq 224$ bits for its private key and assesses public key point $KU_S = KR_S \bullet G$. The Server and all its clients uses the same G and KU_S.

4.1 Registration

The SIMPP design uses the asymmetric (or verifier-based) model [3]; thus once the user completes the Registration phase, the Server stores the long-term public key the user uses for identity authentication instead of the password plaintext. Steps of the SIMPP Registration procedure are as shown in Figure 4:

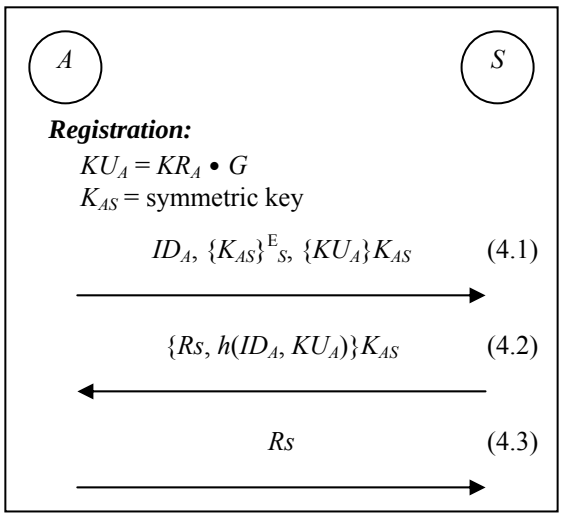


Figure 4: Steps of the Registration Process

1. A selects the ID ID_A. Where A opts for (1) password authentication: Select password pw_A and set the long-term private key $KR_A = h(pw_A)$. Where A opts for (2) smart card authentication: A randomly produced random number $KR_A \geq 224$ bits serves as the private key. The long-term public key $KU_A = KR_A \bullet G$ is assessed, then the randomly produced symmetric key $K_{AS} \geq 128$ bits is relayed (4.1) to the S.
2. Once the S receives (4.1), the K_{AS} is drawn out through the KR_S. Thereafter, the KU_A is drawn through the K_{AS}. The random letter chain produced $R_S \geq 128$ bits is fed back (4.2) to A.
3. Through K_{AS} decryption (4.2), A calculates to determine whether the $h(ID_A, KU_A)$ is consistent with the hash value in (4.2). If numbers are found to be inconsistent, then link is aborted; otherwise value is fed back (4.3) to S.
4. S calculates and reconciles the value received from (4.3) to determine whether value is consistent with R_S. If value is not consistent, it will abort link; otherwise it will store the ID_A and KU_A. If A chooses to use smart card authentication, the ID_A and KR_A are stored into the smart card.

4.2 PAKE and C-S Communication

The steps in the PAKE and Client-Server Communication phase of the SIMPP are as shown in Figure 5:

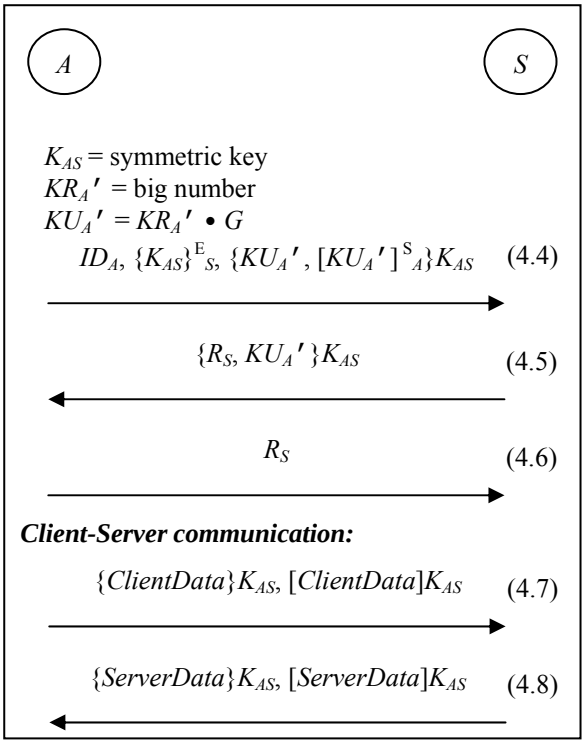


Figure 5: The PAKE and Client-Server Communication Steps

1. Where A opts for (1) password authentication: Enter the ID_A and pw_A and set the long-term private key $KR_A = h(pw_A)$. Where A opts for the (2) smart card authentication: insert smart card to allow ID_A and KR_A reading. A randomly produces the random number $KR_A' \geq 160$ bits, $KR_A' \neq KR_A$, for its short-term private key and calculates the short-term public key $KU_A' = KR_A' \bullet G$. A uses the KR_A to sign the digital signature $[KU_A']^S_A$ on the KU_A'. The randomly produced symmetric key $K_{AS} \geq 128$ bits is then relayed (4.4) to S.
2. Once the S receives the (4.4), the K_{AS} is drawn out through the KR_S. Thereafter, the KU_A' is drawn through the K_{AS}. S verify the $[KU_A']^S_A$ based on the KU_A obtained through the ID_A to determine accuracy. If it is not inaccurate, it will abort link; otherwise S feeds back (4.5) to A.
3. A decrypts the (4.5) through the K_{AS} and compares the KU_A' to determine whether it is consistent with the value received in (4.5). If value is not consistent, it will abort link; otherwise it will feed back (4.6) to S.
4. S reconciles the R_S to determine whether value is consistent with value received in the (4.6). If value is not consistent, it will abort link; otherwise, it will permit A to log in and temporarily store KU_A' into S.
5. Once A has successfully logged in, K_{AS} serves as the short-term session key between A and S. The communication between A and S during the session, as shown in (4.7), and (4.8), is

encrypted through K_{AS} and signed with Message Authentication Code (MAC).

4.3 C-C Key Exchange and Communication

Steps of the SIMPP Client-Client Key Exchange and Communication phase are as shown in Figure 6. SIMPP sends the short-term public key [6] through Presence Service.

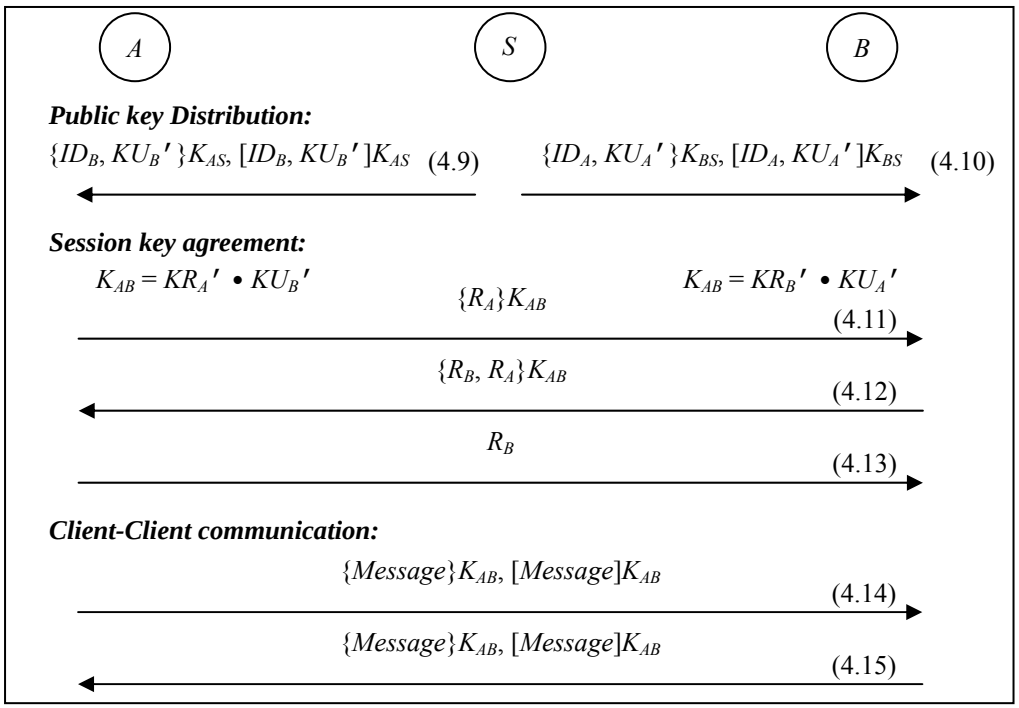


Figure 6: The Client-Client Key Exchange and Communication Steps

1. *A* and *B* uses ECDH (Elliptic Curve Diffie-Hellman) to calculate the common curve point K_{AB}: According to *A's* calculation, $K_{AB} = KR_A' \bullet KU_B'$; whereas in *B's* calculation, $K_{AB} = KR_B' \bullet KU_A'$. In which case, the K_{AB} coordinate values become the symmetric key.
2. Before Client *A* may communicate with user *B*, *A's* randomly generated string of $R_A \geq 128$ bits should be sent (4.11) to *B*.
3. *A* then verifies the R_A to determine whether it is consistent with what (4.12) received. If strings are consistent, then it is fed back (4.12) to *B*. If the strings are not consistent, then *A* rejects the communication request from *B*.
4. After *B* verifies the R_B to determine whether it is consistent with what (4.13) received. If the strings are not consistent, then *B* rejects the communication request from *A*; otherwise, it shall accept communication with *A*. The indirect (communication is relayed through *S*) or direct (P2P) communication between *A* and *B* is as indicated in (4.14) and (4.15); communication is encrypted through K_{AB} encryption and signed with MAC.

5. Analysis and Comparative Study

In a comparison of the security functions, we made a comparative study on the security functions of the (1) Registration, (2) PAKE, (3) Public Key Distribution, and (4) Client-Client Key Exchange and Communication phases of the Kikuchi, IMKE and SIMPP protocols. On

Table 2: A Comparative Study of the Server and Client Computation Costs

Compare / Phases / Protocols		Round	Server Computation Costs								
			Asymmetric cryptosystem					Symmetric cryptosystem			Hash
			E	*D*	*V*	*M*	*K*	*E*	*D*	*C*	
Registration	Kikuchi	1				1					
	SIMPP	3		1				1	1		1
PAKE	IMKE	3	1	1				1	1		3
	SIMPP	3		1	1			1	1		
	SIMPP [a]	3		2	1			2	2		1
Public Key Distribution	Kikuchi	1									
	IMKE	1						1		1	
	SIMPP	1						1		1	
C-C Key Exhcnage	Kikuchi	1				1					
	SIMPP	3									
	IMKE	3									
Compare / Phases / Protocols		Round	Client Computation Cost								
			Asymmetric cryptosystem					Symmetric cryptosystem			Hash
			E	*D*	*S*	*M*	*K*	*E*	*D*	*C*	
Registration	Kikuchi	1				1					
	SIMPP	3	1					1	1		1
PAKE	IMKE	3	1	1				1	1		3
	SIMPP	3	1		1			1	1		
	SIMPP [a]	3	2		1			2	2		1
Public Key Distribution	Kikuchi	1									
	IMKE	1						1		1	
	SIMPP	1						1		1	
C-C Key Exhcnage	Kikuchi	1				1		1	1		
	SIMPP	3					2	1	1		
	IMKE	3	2	2				2	2		4

Note: *a*. SIMPP includes Registrastion phase.

Table 3: A Comparative Study of the Kikuchi, IMKE and SIMPP Security Functions

Implementation Mechanisms / Protocols	SIMPP	IMKE	Kikuchi
Registration:	√	×	√
Wards off offline dictionary and guessing attack (G2)	√	×	×
PAKE and Client-Server Communication:	√	√	×
Supports Client using smart card authentication	√	×	
Protocol model	Asymmetrical	Symmetrical	
Server protects authentication data (G1)	√	×	
Preparation of many different hash functions	Not required	Required	
Security (G4)	*C*, *I*, *A*	*C*, *I*, *A*	
PAKE security requirement (G6)	*S*, *K*, *M*	*S*, *F*, *K*, *M*	
Wards off offline dictionary and guessing attack (G2)	√	√	
Wards off replay attack (G3)	√	√	
Wards off man-in-the-middle attack	√	√	
Wards off DNS spoof attack	√	√	
Public Key Distribution:	√	√	√
Security	*C*, *I*, *A*	*C*, *I*, *A*	×
Wards off man-in-the-middle attack	√	√	√
Client-Client key Exchange and Communication:	√	√	√
Security (G5)	*C*, *I*, *A*	*C*, *I*, *A*	*C*
PAKE security requirement (G6)	*S*, *F*, *K*, *M*	*S*, *F*, *K*, *M*	×
Wards off man-in-the-middle attack	√	√	×

the cost comparison side, we compared the computation cost of these four phases. Table 2 presents a comparative study of the Server and Client computation costs. The computation cost sets *E* as the usage encryption count. Moreover, *D* is set as the decryption count; *V* is set as the digital signal verification count; *M* is set as the modulus computation count; *S* is set as the digital signal singing count; *K* is set as the key generation count; *C* is set as the MAC singing (or verification) count. Table 3 presents a comparative study of the Kikuchi, IMKE, and SIMPP security functions. In the security (G4 and G5) table of comparison, *C* stands for Confidentiality, *I* stands for Integrity, *A* stands for Authentication. In the PAKE security requirement (G6) item, *S* stands for Session-Key Security, *F* stands for Forward Secrecy, *K* stands for Known-Key Security, and *M* stands for Mutual Authentication.

6. System Implementation

The SIMPP Server is used to institute amendments from jabberd 2.0s10 to allow operation in Linux (Fedora Core 6). Windows XP/Borland C++ Builder 6 is used to develop the SIMPP Client. Table 4 shows the implementation specifications and computation cost of the cryptosystem. Table 5 shows the average efficiency of the Server and Client after 1,000 rounds of tests conducted in the three different SIMPP and IMKE phases, (1) Registration, (2) PAKE and (3) Client-Client Key Exchange, under the environments shown in Table 4. Table 4 don't include key-pair generation and network transmission time.

Table 4: Cryptosystem Implementation Specifications and computation cost

Asymmetric Cryptosystem		ECC *GF*(*p*) Server 224 bits / Client 160 bits RSA Server 2048 bits / Client 1024 bits					
Digital signature		ECDSA (Elliptic Curve Digital Signature Algorithm)					
Symmetric cryptosystem		128 bits AES/CBC mode					
One-way hash function		SHA-256 / HMAC (using SHA-256)					
Libraries		MIRACL 5.21 & OpenSSL 0.9.8a					
Operations / Machines	Hardware	RSA		ECC		ECDSA	
		Encryption	Decryption	Encryption	Decryption	Singing	Verifcation
Server	Pentium Dothan 3.2GHz	1.21	23.29	5.34	2.97	2.50	3.31
Client	Pentium M 740MHz	0.39	4.28	3.95	2.23	1.86	2.42

Table 5: Average Efficiency of the SIMPP and IMKE Server and Client (unit: millisecond)

Steps / Protocol	Registration		PAKE (without Registration [a])			PAKE (with Registration [b])			Client-Client Key Exchange [c]
	S	C	S	C	Total	S	C	Total	
SIMPP	3.16	8.05	6.54	11.31	17.85	9.70	19.36	29.06	3.85
IMKE	╳		24.05	5.66	29.71	24.05	5.66	29.71	9.64

Note: *a*. Efficiency under conditions where SIMPP does not include a Registration phase.
b. Efficiency under conditions where SIMPP includes a Registration phase.
c. Efficiency during the Client testing of the Client-Client Key Exchange (excluding the network transmission time of transmissions routed through the Server).

7. Conclusion

In view of the IM system framework (Figure 1) and security requirements, this paper designs a three-party PAKE based SIMPP to resolve the security problems in the Registration, Client-Server authentication, and Client-Client communication processes. When compared to the Kikuchi, the SIMPP is found to offer more security functions. When compared to the IMKE, the asymmetric model protocol design of the SIMPP enables it to protect the authentication data stored in the Server, as well as provide suggestions that could ward off attackers from acquiring user authentication data and thereafter, launch offline dictionary attack and guessing attack on user (§4.1). If an attacker manages to obtain the authentication data stored in the Server, attacker would not be able to launch an impersonation attack. The SIMPP ensures the IM secure design, and thus is quite valuable for the security applications of the communications and coordination matters of businesses, groups, and individuals.

Acknowledgement

This project is sponsored by National Science Council of Taiwan (NSC 95-2221-E-017-007). This work was supported in part by TWISC@NCKU, National Science Council under the Grants NSC 94-3114-P-006-001-Y.

References

[1] C. E. Dalton and W. Kannengeisser, “Instant Headache,” in *Hong Kong CERT/CC Security Bulletin*, Apr. 2003, pp.2-7.

[2] H. Kikuchi, M. Tada, and S. Nakanishi, “Secure Instant Messaging protocol preserving confidentiality against administrator,” in *18th Int’l Conf. Advanced Information Networking and Applications (AINA 2004), Fukuoka, Japan*, Mar. 2004, Vol. 2, pp. 27–30.

[3] Jeong Ok Kwon, Ik Rae Jeong, Kouchi Sakurai, and Dong Hoon Lee, “Directions in Password-Authenticated Key Exchange,” in *Communications of the CCISA, Special Issue-Implementation Aspects of Cryptosystems*, Vol. 12 No. 2, Apr. 2006, pp. 43-60.

[4] Kiyomichi Araki, Takakazu Satoh, and Shinji Miura, “Overview of Elliptic Curve Cryptography,” in *Lecture Notes in Computer Science*, Vol. 1431, Feb. 1998, pp. 29-48.

[5] M. Day, J. Rosenberg, and H. Suugano, “A Model for Presence and Instant Messaging,” in *IETF RFC 2778*, Feb. 2000.

[6] M. Mannan, P.C. van Oorschot, “A Protocol for Secure Public Instant Messaging,” in *Financial Cryptography and Data Security 2006 (FC'06)*, Feb. 27-Mar. 2 2006.

[7] M. Mannan and P.C. van Oorschot, “Secure Public Instant Messaging: A Survey,” in *Second Annual Conference on Privacy, Security and Trust (PST), Fredericton, NB*, Oct. 2004, pp 69-77.

[8] N. Leavitt, “Instant Messaging: A New Target for Hackers,” in *IEEE Computer Society*, Vol. 38, No. 7, Jul. 2005, pp.20-23.