Design and Implementation of a Secure Instant Messaging Service based on Elliptic-Curve Cryptography

Chung-Huang Yang[1,*] Tzong-Yih Kuo[1] TaeNam Ahn[2] Chia-Pei Lee[1]

[1] Graduate Institute of Information and Computer Education
National Kaohsiung Normal University
Kaohsiung 802, Taiwan, ROC
chyang@nknucc.nknu.edu.tw

[2] Security Engineering Research Center
Hannam University
Daejeon 306-797, Korea
taenamahn@hotmail.com

Received 16 November 2007; Revised 30 November 2007; Accepted 9 December 2007

Abstract. Instant Messaging (IM) is a useful communication and work collaboration tool between individuals, groups, or enterprises. Unfortunately, most IM systems lack the needed security mechanism capable of ensuring the secure communications of IM client-client and IM client-server. In order to find a solution to secure IM communications, we designed and implemented a Secure Instant Messaging and Presence Protocol (SIMPP) based on elliptic-curve cryptography. The proposed IM service is compatible with the IETF XMPP (eXtensible Messaging and Presence Protocol)/Jabber Standard. Open source jabberd software was revised to create a SIMPP server on the Linux platform, wherein we used C++Builder to create a SIMPP client on the Windows platform. Our IM client and IM server use open source MIRACL cryptographic libraries with iksemel XMPP library.

Keywords: instant messaging, XMPP, peer-to-peer, key exchange, open source

1 Introduction

Instant Messaging (IM) [1] is one of the most important Internet applications and people are using IM both for personal and business reasons. However, most IM systems are not secure [2,3]. For instance, in the MSN Messenger, any user that has successfully logged into the system will communicate in plaintext with other users [4,5], communications are not properly protected. In year 2000, IETF released the RFC 2778 standard [6] which defines IM systems to be composed of two types of services, Presence Service and Instant Messaging Service, as shown in Fig. 1. The Presence Service, shown in Fig. 1(a), is responsible for the presence exchanges where the Watcher will receive presence information provided by the Presentity. Presence information includes users' status and willingness to accept or decline a chat session. The Instant Messaging Service, shown in Fig. 1(b), is responsible for the inter-client real-time message exchanges.

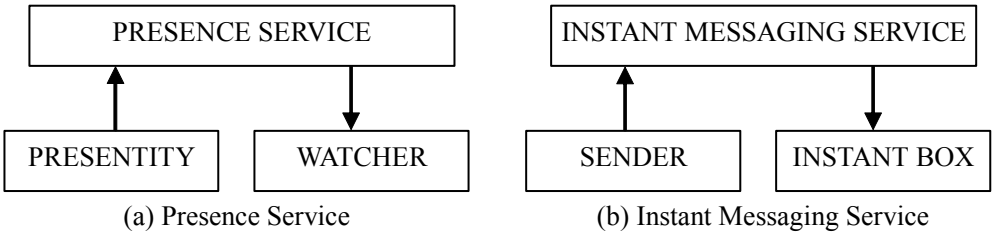


Fig. 1. Two types of IM service models defined in RFC 2778

* Correspondence author

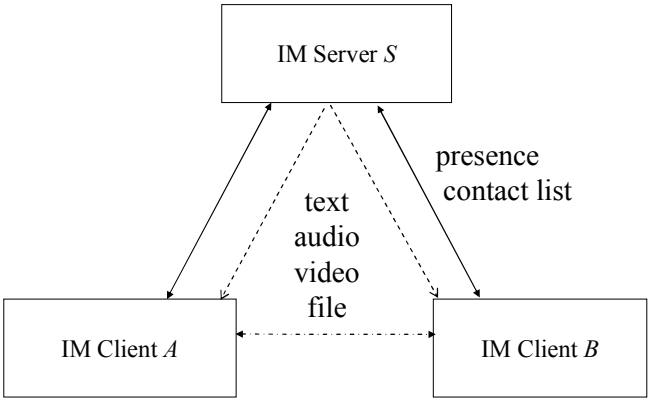


Fig. 2. Three-Way IM communication model

Under the IM service models, data communications between any two 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 contents revealed at the server after message has been transmitted between clients and from clients to the server. In this research, we designed and implemented a secure IM protocol, SIMPP (Secure Instant Messaging and Presence Protocol) for the 3-way IM communication model, shown in Fig. 2, thus we enhanced the communication security of the IM system. The proposed SIMPP was implemented with the XMPP/Jabber [7,8,9] protocol of RFC 3920-3923 and JEPs (Jabber Enhancement Proposals) while ECC (Elliptic Curve Cryptosystem) [10] was selected to speedup public-key cryptographic functions.

2 Secure IM Protocols

In [11], Berson reported that SKYPE 1.3 was implemented with cryptographic functions correctly and efficiently, but few details were given about SKYPE protocol and system security. No much work has been done on IM security in academia. In order to avoid administrator eavesdropping into client-client communication, Kikuchi, Tada, and Nakanishi [12] designed a secure IM protocol based on the Diffie-Hellman key-agreement algorithm. Their design, as shown in Fig. 3, which will be called KTN protocol in this paper, is consisted of two phases: (1) registration, (2) key establishment in peer-to-peer. On registration, the server will initiate the requesting user to create a client private key (x_i) and public key (y_i). The final step on registration is basically a public key distribution, stored other user's public key. Main drawback of this KTN protocol is that IM server is required to perform modular exponentiation operations both during registration and key establishment Phase. Therefore, heavy computational overhead on the server.

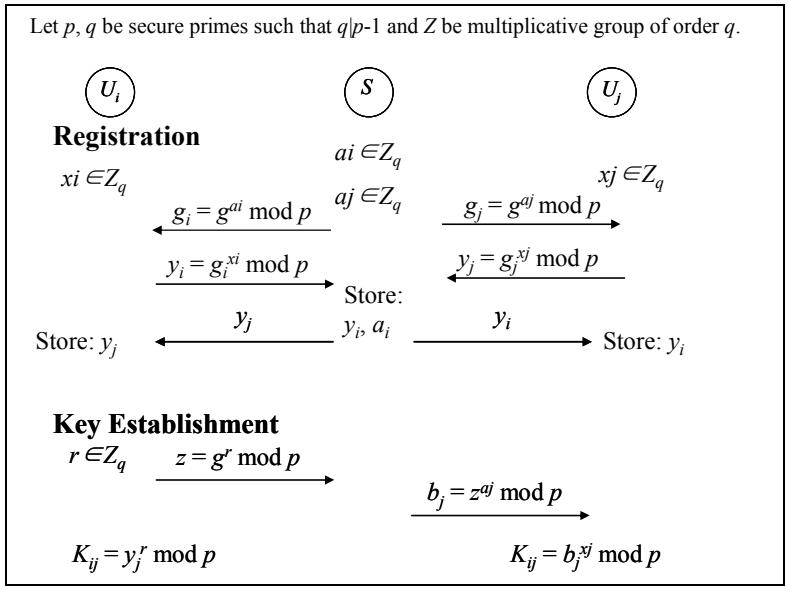
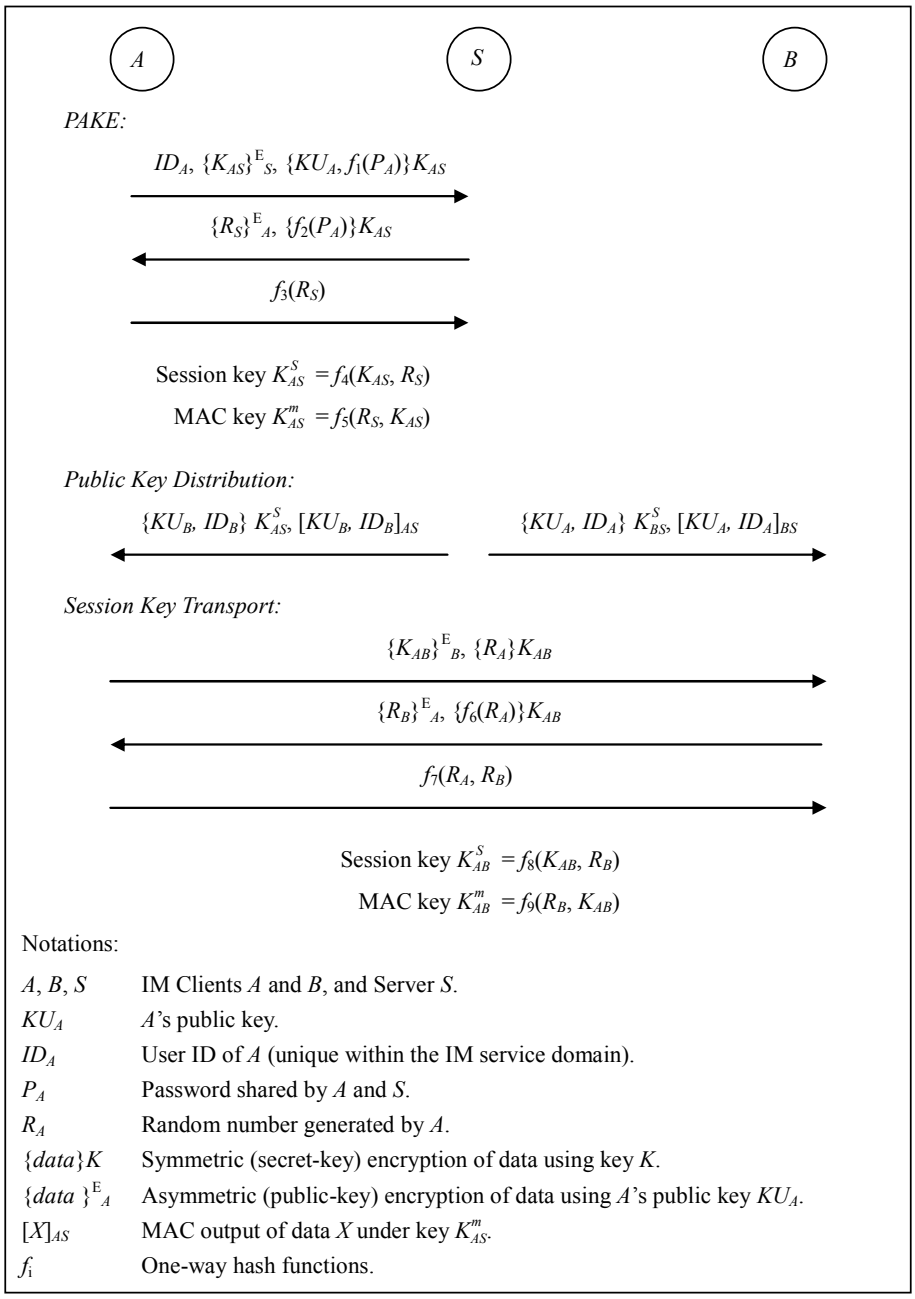


Fig. 3. The KTN protocol [5]



Notations:

A, B, S	IM Clients A and B, and Server S.
KU_A	A's public key.
ID_A	User ID of A (unique within the IM service domain).
P_A	Password shared by A and S.
R_A	Random number generated by A.
$\{data\}K$	Symmetric (secret-key) encryption of data using key K.
$\{data\}^{E}_{A}$	Asymmetric (public-key) encryption of data using A's public key KU_A.
$[X]_{AS}$	MAC output of data X under key K^m_{AS}.
f_i	One-way hash functions.

Fig. 4. The IMKE protocol [6]

Mannan and van Oorschot then designed an IMKE (Instant Messaging Key Exchange) protocol [13] to ensure the confidentiality, integrity and authentication of client-server and client-client communications. The IMKE is consisted of three phases: (1) password-authenticated key exchange (PAKE) [14], (2) public key distribution, and (3) session key transport. Fig. 4 illustrates the IMKE protocol. The session key used for message encryption in IMKE is derived from short-lived secrets and provides the so-called "forward secrecy" to users. Also, IMKE allows authentication of exchanged messages between tow parties, and the send is able to repudiate a message. The authors also implemented IMKE based on RSA cryptographic algorithm. In comparison, KTN protocol is more efficient than IMKE protocol but IMKE protocol provides better security than the KTN protocol.

3 Secure Instant Messaging and Presence Protocol (SIMPP)

The main objective of the proposed secure IM design is to reduce computational overhead imposed on an IM systems due to security enhancement. Since security does not come from free and additional computational time is required to perform security functions, therefore, we revised the IMKE protocol to get a more efficient IM

system, while still maintaining its security. The proposed Secure Instant Messaging and Presence Protocol (SIMPP) contains three phases: (1) registration, (2) client-server communications, and (3) client-client communications. Our design used notations and definitions shown in Table 1.

Elliptic curve cryptosystems (ECCs) have been accepted in the standardization bodies and recently adopted as ANSI X9.62, ISO 11770, IEEE P1363, and FIPS 186-2 standards. We use the ECC over prime field [10] to implement SIMPP. Significantly smaller cryptographic parameters can be used in ECC than in other competitive public-key cryptographic systems such as the popular RSA cryptosystem but with equivalent levels of security. For ECC in our design, first an elliptic curve $\boldsymbol{E}$: $y^2=x^3+ax+b$ over prime field $\boldsymbol{F}(p)$ is chosen. Then system parameters of a base point $\boldsymbol{G} = (x_G, y_G)$ on $\boldsymbol{E}$ is selected which must have large order n (ANSI X9.62 mandates that $n > 2^{160}$). Each entity then find a statistically unique and unpredictable integer d in the interval [1, n-1] and compute the point $\boldsymbol{Q} = (x_Q, y_Q) = d \bullet \boldsymbol{G}$, where d is the private key while $\boldsymbol{Q}$ is the public key of entity.

We will assume that IM server and all IM clients will use the same base point $\boldsymbol{G}$, IM server has randomly selected a big integer KR_S as its private key and derives corresponding public key point $KU_S = KR_S \bullet G$. This public key KU_S is known to all IM clients.

Table 1. Notations used in the proposed SIMPP

A, B, S	User A, B and server S
ID_A	A's ID
pw_A	A's password, randomly generated string
R_S	Random string generated by S
KU_A, KR_A	A's public key KU_A and private key KR_A
K_A, K_B	Master key sharing between A (B) and server S
$\{data\}K_{AS}$	Encrypt *data* with symmetric key K_{AS}
$[data]K_{AS}$	HMAC of *data* with key K_{AS}
$[data]^{sign}_A$	Sign *data* with A's private key KR_A
h	One-way hash function

3.1 Registration

Every new IM user has to register for once. SIMPP registration phase is shown in Fig. 5.

1. User A generates a random secret number KR_A in the interval [1, n-1] or computes KR_A from $h(pw_A)$ with randomly generated secret password pw_A that might be stored on a smart card. Public key of A will be $KU_A = KR_A \bullet G$. A then calculates the master key K_A to be shared with server
$$K_A = KR_A \bullet KU_S$$
where KU_S is the known public key of server S. This K_A is actually a point of elliptic curve but its coordinate could be used for cryptographic key. A sends message ID_A and KU_A (R1 in Fig. 5) to S.
2. Once S receives (R1), S calculates the master key sharing with A by the following equation
$$K_A = KR_S \bullet KU_A$$
where KR_S is the secret key of server S. S then generates a random string R_S and sends encrypted message (R2) back to A.

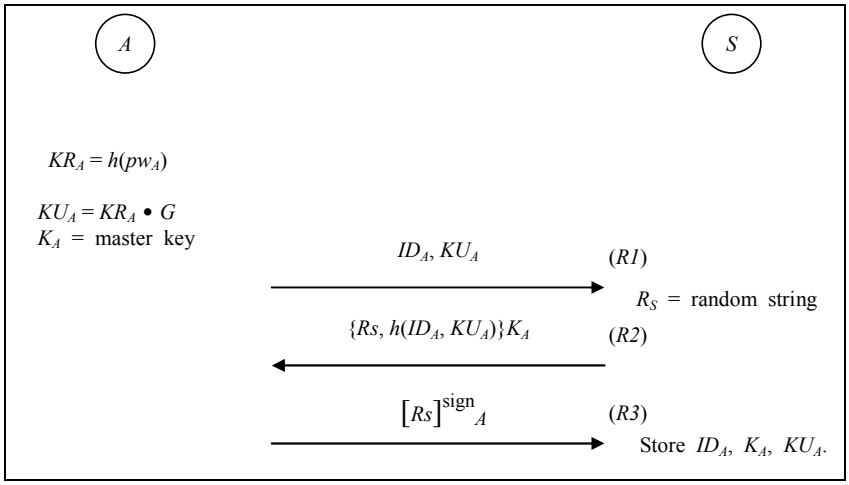


Fig. 5. SIMPP registration

3. *A* decrypts message (R2), compares it with the hash value of message (R1) created before, and disconnects if the two quantities are unequal. Otherwise, *A* secretly stores K_A and digitally signs R_S with private key KR_A and sends message (R3) to *S*.
4. *S* verifies signature (R3) with public key KU_A of *A* and disconnects if verification failed. Otherwise, *S* stores ID_A, KU_A, and K_A.

3.2 Client-server Communications

In the client-server communication phase, user *A* and server *S* authenticate each other using the pre-established master key K_A during the registration phase and create a dynamic secret session key K_{AS}. This client-server communications phase is shown in Fig. 6.

1. *A* randomly generates symmetric session key K_{AS} which is then encrypted with the master key K_A sharing with *S*. *A* sends (S1) to *S*.
2. *S* looks up master key K_A using ID_A, symmetrically decrypts (S1) with K_A, and gets K_{AS}. *S* generates a random challenge R_S, calculates $h(ID_A, K_{AS})$, and responds with encrypted message (S2) to *A*.
3. Once *A* receives (S2), decrypts with the master key K_A, calculates $h(ID_A, K_{AS})$ independently and compares it with the corresponding value received with (S2), and disconnects if the two quantities are unequal. Otherwise, *A* computes hashed value of R_S, encrypts with session key K_{AS}, and responds with (S3).
4. *S* independently computes hashed value of R_S and compares it with the quantity received in message (S3). If they mismatch, *S* disconnects; otherwise, *S* indicates *A* successful IM client login.

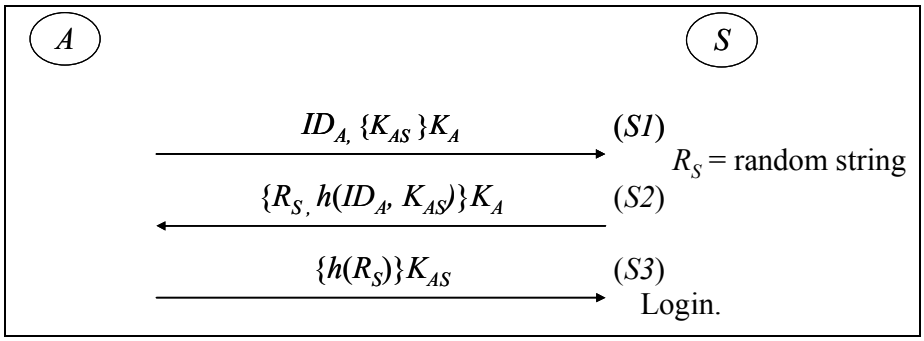


Fig. 6. SIMPP client-server communications

3.3 Client-client Communications

When two users, *A* and *B*, both successfully login into system and accept each other for communication, they will at first obtain public key of each other. Then they apply the Elliptic Curve Diffie-Hellman (ECDH) key exchange mechanism to establish a symmetric key. To avoid the common man-in-the-middle problem of Diffie-Hellman schemes, they will mutually authenticate data with the Elliptic Curve Signature Algorithm (ECDSA). This client-client communications phase is shown in Fig. 7.

1. *A* and *B* securely receive the other party's public key from *S* through messages (C1) and (C2).
2. *A* randomly generates an integer x in the interval [1, n-1] and computes $\boldsymbol{G}_x = x \bullet \boldsymbol{G}$, sends message (C3) of $\boldsymbol{G}_x$ to *B* with its signature.
3. *B* randomly generates an integer y in the interval [1, n-1] and computes $\boldsymbol{G}_y = y \bullet \boldsymbol{G}$, sends message (C4) of $\boldsymbol{G}_y$ to *A* with its signature.
4. *A* and *B* verify the signature received from the other party, disconnects if verification failed. Otherwise, they use ECDH algorithm (C5) to calculate the common curve point K_{AB}.
$$K_{AB} = x \bullet \boldsymbol{G}_y = y \bullet \boldsymbol{G}_x$$
5. With common key K_{AB}, *A* and *B* could then securely communicate with each other by encryption and message authentication (C6 and C7).

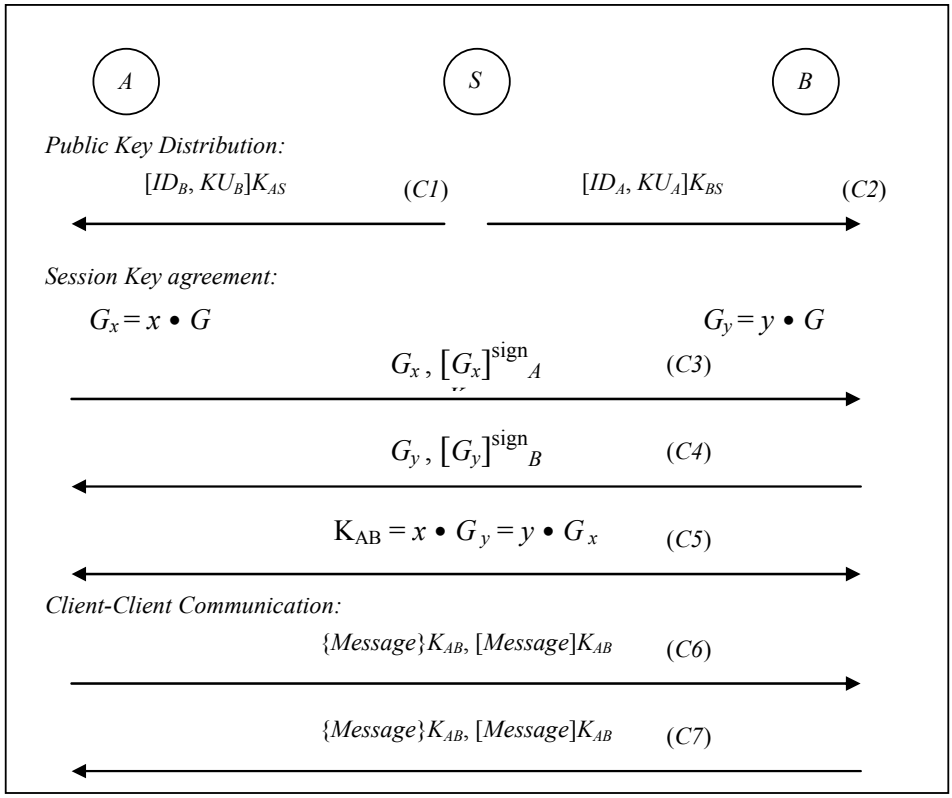


Fig. 7. SIMPP client-client communications

4 System Implementation

We implemented a SIMPP IM prototype system through the open source XMPP/Jabber (http://www.jabber.org/) and used iksemel (http://iksemel.jabberstudio.org/) to conduct XMPP encoding and decoding. Cryptographic functions of our system is established with MIRACL (http://www.shamus.ie/). The SIMPP server is revised from open source jabberd (http://jabberd.jabberstudio.org/) and operated in Linux platforms. On the other hand, Borland C++ Builder 6 is used to develop the SIMPP client on Windows platforms. Fig. 8 illustrates main menu for our client for registration and file transfer. Table 2 shows the implementation specifications of our secure IM systems.

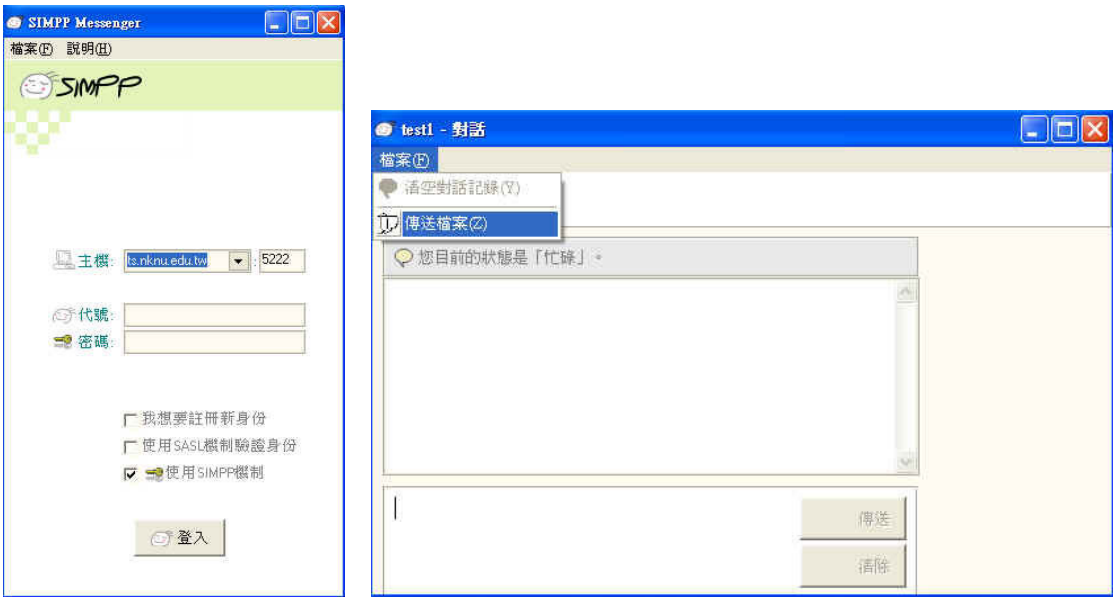


Fig. 8. Secure IM client software with SIMPP

Table 2. SIMPP implementation specifications

Server	Revised from jabberd run on Linux
Client	Software implemented using Borland C++ Builder 6 on Windows XP
Database	MySQL
Public-Key Cryptosystem	ECC *GF*(*p*) Server 224 bits / Client 160 bits
Symmetric Cryptosystem	128 bits AES with CBC mode
One-way Hash Function	SHA-256
MAC Function	HMAC using SHA-256
Crypto Library	MIRACL
XMPP/Jabber Library	iksemel

5 Conclusion

The growth of IM use is inevitable and IM is now everywhere: desktops, laptops, cell phones, PDAs, etc. However, IM also provides a significant security risk and public IM products, such as Microsoft's MSN Messenger, generally contain no provision for message confidentiality. In this research, we designed an efficient Secure Instant Messaging and Presence Protocol (SIMPP) to provide a secure and efficient IM system. SIMPP is complied with the IM service model defined by IETF RFC 2778 standard and is based on elliptic-curve cryptography to give better performance. We also implemented a secure SIMPP IM system where secure IM server is revised from jabberd open source software while secure IM client is developed with ourselves, both use MIRACL and iksemel for cryptographic and XMPP functions.

6 Acknowledgement

This research was partially supported by a grant (NSC 95-2221-E-017-007) from the National Science Council of Taiwan.

References

[1] S. Chatterjee, T. Abhichandani, L. Haiqing, B. TuIu, and B. Jongbok, "Instant messaging and presence technologies for college campuses," *IEEE Network*, Vol. 19, No. 3, pp. 4-13, May-June 2005.

[2] N. Leavitt, "Instant Messaging: A New Target for Hackers," *IEEE Computer*, Vol. 38, No. 7, pp. 20-23, July 2005.

[3] M. Mannan and P. C. van Oorschot, "Secure Public Instant Messaging: A Survey," *Proceedings of the Second Annual Conference on Privacy, Security and Trust (PST)*, pp. 69-77, 2004.

[4] MSN Sniffer, http://www.effetech.com/msn-sniffer/

[5] J. Rittinghouse and J. Ransome, *Instant Messaging Security*, Elsevier Digital Press, 2005.

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

[7] P. Saint-Andre, "Streaming XML with Jabber/XMPP," *IEEE Internet Computing*, Vol. 9, No. 5, pp. 82-89, Sep./Oct. 2005.

[8] P. Saint-Andre, ed., *Extensible Messaging and Presence Protocol (XMPP)*: Core, IETF RFC 3920, Oct. 2004.

[9] I. Shiegoka, *Instant Messaging in Java: The Jabber Protocols*, Manning Publications, 2002.

[10] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[11] T. Berson, "SKYPE Security Evaluation," http://www.skype.com/security/files/2005-031%20security%20evaluation.pdf, Oct. 2005,

[12] H. Kikuchi, M. Tada, and S. Nakanishi, "Secure Instant Messaging Protocol Preserving Confidentiality against Administrator," *Proceedings of the 18th Int'l Conf. Advanced Information Networking and Applications (AINA)*, Vol. 2, pp. 27–30, 2004.

[13] M. Mannan and P.C. van Oorschot, "A Protocol for Secure Public Instant Messaging," *Proceedings of the Financial Cryptography and Data Security 2006 (FC'06)*, pp. 20-35, 2006.

[14] J. O. Kwon, I.R. Jeong, K. Sakurai, and D.H. Lee, "Directions in Password-Authenticated Key Exchange," *Communications of the CCISA*, Vol. 12 No. 2, pp. 43-60, Apr. 2006.