Practical anonymous proxy signature schemes without trusted alias issuers

Hwang, Shin-Jia and Hsu, Pi-Hung*
Department of Computer Science and Information Engineering,
TamKang University, Tamsui, Taipei Hsien, 251, Taiwan, R.O.C.
E-mail: sjhwang@mail.tku.edu.tw
**E-mail: 795410132@s95.tku.edu.tw*

Abstract*-Anonymous proxy signature schemes are used to protect proxy signers' privacy. Among the proposed anonymous proxy signature schemes, Shum and Wei's scheme is more practical than the other proposed scheme. However, their scheme does not satisfy strong unforgeability, and their scheme needs the help of trusted alias issuers to protect the proxy signer's anonymity. To overcome this disadvantage, Hwang and Hsu's scheme is proposed. But Hwang and Hsu's scheme still needs the help of trusted alias issuers. To remove the trusted alias issuers, a new practical anonymous proxy signature scheme is first one with the help of concurrent signature schemes to deal with the signatures exchange between the original signer and proxy signer. In the new scheme, the unused anonymous names of proxy signers keep secret to protect proxy signers' privacy. Once the anonymous names are used, the original signers have the evidences to identify anonymous proxy singers at the same time. Therefore, the right of original signers is also protected.*

Keywords: Anonymity, proxy protection, proxy signatures, concurrent signatures

1.Introduction

Mambo et al. [10, 11] first proposed the concept of proxy signature schemes in 1996. In a proxy signature scheme, an original signer (denoted by U_O) is able to authorize a proxy signer (denoted by U_P) to generate proxy signatures on behalf of the original signer U_O. Mambo et al. [10, 11] proposed three authorization types: Full delegation, partial delegation, and delegation by warrant. Based on these three types of authorization, many proxy signature schemes are proposed [3- 8].

Some of these proposed proxy signature schemes provide proxy protection. The proxy signer protection is obtained by using proxy signers' public key to help the verification of proxy signatures. Due to proxy signers' public keys are used, the identity of a proxy signer should be known. But this damages the proxy signers' privacy. To protect privacy, it is better that proxy signers are anonymous in proxy signature schemes. To deal with anonymous issues, there are two kinds of proxy schemes are proposed. One is Mehta and Harn's one-time proxy signature scheme [12], and the other are Shum and Wei's strong proxy signature scheme [13] and Hwang and Hsu's scheme [2].

Mehta and Harn's [12] scheme is proposed based on online/offline signature schemes in the delegation by warrants. Their scheme has anonymity property for proxy signers because proxy signatures are validated only by a using U_O's public key. Then only the original signer U_O and proxy signer U_P know the actual signer of proxy signatures. When any dispute caused by proxy signatures happens, it is necessary to find out the actual signer of proxy signatures. In Mehta and Harn's scheme, both U_O and U_P have the evidence to identify the actual proxy signers with the help of a trust authority U_T. The proxy signatures in [12] are perfectly secure. However, for each delegation by warrant, each proxy signer can generate only one proxy signature. So their scheme is impractical in the real word.

On the other hand, Shum and Wei's scheme [13] is practical with the help of trusted alias issuers. The trusted alias issuer issues certificates for the anonymous name and anonymous public key for each proxy signer. Then the certificated anonymous public key is used to generate anonymous proxy private and public key with the original signer's authorization. Then proxy signatures are generated by the anonymous proxy private key, so any third party cannot know the proxy signer's identity. However, Lee et al. [9] showed that Shum and Wei's [13] scheme doesn't satisfy the strong unforgeability property. To overcome this problem, Hwang and Hsu [2] proposed a practical proxy scheme with the help of trusted alias issuers. However, the

maintaining load of trusted alias issuers is heavy. To remove the trusted alias issuers makes anonymous proxy signature schemes being more practical. Therefore, an anonymous proxy signature scheme without trusted alias issuers is proposed.

An anonymous scheme has to satisfy security properties given below. Some of these properties have been previously listed by Mambo et al. [10, 11] and Shum and Wei [13] for their schemes, respectively.

1. **Unforgeability**: Only the proxy signer with proxy authorization can generate valid proxy signatures. Proxy signatures cannot be forged by any unauthorized user, except the original signer.
2. **Verifiability**: Anyone can validate whether or not proxy signatures are generated by the authorized proxy signers are correct.
3. **Proxy signer's deviation**: The proxy signer cannot forge the original signer's signatures or obtain original signers' private keys.
4. **Distinguishability**: Anyone is able to distinguish proxy signatures, original signers' signatures, and proxy signers' signatures in polynomial time.
5. **Identifiability**: Original signer can to identify the generator of proxy signatures.
6. **Proxy protection**: Due to the anonymity, an original signer can authorize himself/herself as the proxy signer to generate valid proxy signatures. But an original signer cannot falsely incriminate anyone as the proxy singer.
7. **Undeniability**: For a valid proxy signature, the proxy signer cannot deny the generation of proxy signatures. Moreover, the original signer cannot deny the proxy certificates generated by him/her.
8. **Anonymity**: Besides the proxy signer or original signer, any third party cannot directly find the proxy signer's identity out, even through proxy signatures or proxy certificates.
9. **Original signer's deviation**: Original signer cannot forge the proxy signer's signatures. Original signer cannot discover the private key from the proxy signer's signatures.

To propose a practical anonymous proxy signature scheme, a new scheme is proposed without the trusted authority that is necessary in [13]. To deal with the signature exchange between original signers and proxy signers, our scheme utilizes the asymmetric concurrent signature schemes [6]. The following section gives the review of Nguyen's asymmetric concurrent signature schemes [6] that is used in our scheme. Our scheme is proposed in Section 3. Then the security analysis of our scheme is given in Section 4. The last section is our conclusions.

2. Review of Asymmetric Concurrent Signature Schemes

The asymmetric concurrent signature scheme proposed by Nguyen [5] is reviewed here. The scheme is consisted of five algorithms and one protocol. The five algorithms are described in the following subsection and then the protocol is given in Section 2.2.

2.1 Asymmetric concurrent signature algorithms

The six algorithms in Nguyen's scheme are **SETUP**, **ISIGN**, **SSIGN**, **IVERIFY**, **SVERIFY**, and **CVERIFY**. They are described one by one below. To describe the six algorithms, the notations y_i and y_j denotes two distinct public keys with $y_i \neq y_j$.

SETUP

The input of this algorithm is a security parameter *l*. On the input *l*, **SETUP** algorithm outputs the following parameters:

(1) Two large public prime numbers p and q such that q|(p-1) and q is exponential in *l*,
(2) A public element $g \in Z_p^*$ of order q,
(3) A public cryptographic hash function $H:\{0, 1\}^* \rightarrow Z_q^*$,
(4) Three public functions **KGEN**$(x)= g^x \bmod p$, **KGEN**$_j(k) = y_j^k \bmod p$ for all j, and $KTRAN(s, x)= s^x \bmod p$,
(5) Three public spaces $M= \{0,1\}^*$, $K= Z_q^*$, and F is the subgroup of Z_q^* generated by g, and
(6) The certified private-public key pair (x_i, y_i) for the user U_i, where $x_i \in Z_q^*$ and $y_i = g^{x_i} \bmod p$.

ISIGN

Algorithm **ISIGN** takes the input (y_i, x_i, m_i), and then outputs the promise of the Schnorr signature $\sigma_i= (s_i, c_i)$ and keystone k, where x_i is the private key matching with y_i (i.e. $y_i = g^{x_i} \bmod p$) and $m_i \in M$ is the message to be signed. **ISIGN** algorithm is consisted of the following steps.

(1). Select a random number $r \in Z_q$.
(2). Compute $c_i = H(g^r \bmod p, m_i)$.

(3). Compute $k = r + c_i x_i \bmod q$.
(4). Compute $s_i = \mathbf{KGEN}(k) = g^k \bmod p = g^{r+c_i x_i} \bmod p$.
(5). Output a promise of Schnorr signature $\sigma_i = (s_i, c_i)$ and the keystone k.

SSIGN

Algorithm **SSIGN** takes the input (y_j, x_j, m_j, s_i), and then outputs the promise of Schnorr-like signature $\omega_j = (s_j, k_1, c_j)$ and keystone k_1, where x_j is the private key matching with y_j (i.e. $y_j = g^{x_j} \bmod p$), s_i is the promise of Schnorr signature's parameter and $m_j \in M$ is the message to be signed. **SSIGN** algorithm is consisted of the following steps.

(1). Select a random number $r' \in Z^*_q$.
(2). Compute $s_j = KTRAN(s_i, x_j) = s_i^{x_j} \bmod p$.
(3). Compute $c_j = H(g^{r'} s_j, m_j)$.
(4). Compute $k_1 = (r' - c_j) x_j^{-1} \bmod q$.
(5). Output the promise of Schnorr-like signature $\omega_j = (s_j, k_1, c_j)$, where k_1 is the keystone.

IVERIFY

IVERFY algorithm takes the input (σ_i, m_i, y_i), and returns **accept** if $c_i \equiv H(s_i y_i^{-c_i} \bmod p, m_i)$; otherwise IVERFY returns **reject**.

SVERIFY

SVERFY algorithm takes the input (ω_j, m_j, y_j), and returns **accept** if $c_j \equiv H(g^{c_j} y_j^{k_1} s_j \bmod p, m_j)$; otherwise **SVERFY** returns **reject**.

CVERIFY

This algorithm is used to verify the promise of signature. The algorithm can describe in two cases:

(1) On input the promise of Schnorr signature $\sigma_i = (s_i, c_i)$, m_i, and keystone k, the algorithm output **accept** if $\mathbf{KGEN}(k) = s_i$ and $\mathbf{IVERIFY}(\sigma_i, m_i, y_i)$=accept.
(2) On input the promise of Schnorr-like signature $\omega_j = (s_j, k_1, c_j)$, m_j, and keystone k and the algorithm output **accept** if $\mathbf{KGEN}_j(k) = s_j$ and $\mathbf{SVERIFY}(\omega_j, m_j, y_j)$= accept.

2.2 Concurrent signature protocol

The protocol of Nguyen's scheme is stated below. Assume that Alice is the initial signer and Bob is the matching signer. Alice and Bob perform the following protocol to generate and exchange their concurrent signatures.

Step 1: Alice generates the promise of Schnorr signature σ_A on the message $m_A \in M$ by the following steps.
(1) Perform $\mathbf{ISIGN}(y_A, x_A, m_A)$ to obtain the promise of Schnorr signature $\sigma_A = (s_A, c_A) = \mathbf{ISIGN}(y_A, x_A, m_A)$ and the keystone k.
(2) Send (σ_A, m_A) to Bob.

Step 2: Bob performs the following steps to first validate Alice's promise of Schnorr signature σ_A and then generates Bob's promise of Schnorr-like signature ω_B if σ_A is valid.
(1) Validate Alice's promise of Schnorr signature σ_A and the message m_A by performing $\mathbf{IVERIFY}(\sigma_A, m_A, y_A)$. If $\mathbf{IVERIFY}(\sigma_A, m_A, y_A) \neq$ **accept**, then abort.
(2) Perform $\mathbf{SSIGN}(y_B, x_B, m_B, s_A)$ to generate Bob's promise of Schnorr-like signature $\omega_B = (s_B, k_1, c_B) = \mathbf{SSIGN}(y_B, x_B, m_B, s_A)$, where s_A is the second component in σ_A.
(3) Send (ω_B, m_B) to Alice.

Step 3: After receiving (ω_B, m_B), Alice runs $\mathbf{SVERIFY}(\omega_B, m_B, y_B)$ to verify the validity of ω_B. If $\mathbf{SVERIFY}(\omega_B, m_B, y_B)$= **accept**, Alice uses the keystone k to verify the keystone fix s_B. If this keystone fix is valid, Alice forwards the keystone k to Bob.

3. Our new scheme

The underlying algorithms used to design our scheme are first described. Then our new scheme is described.

3.1 Underlying algorithms in our scheme

In this subsection, some underlying algorithms are defined. Our new scheme adopts concurrent signature schemes to give original signer's authorization to proxy signer, the algorithms, **ISIGN**, **SSIGN**, **IVERIFY**, **SVERIFY,** and **CVERIFY**, in asymmetric concurrent signature scheme are used. Moreover, in order to generate signatures, our scheme needs discrete-logarithm-based signature schemes. The underlying discrete logarithm based signature schemes have two basic algorithms. One is the **SIGN** algorithm to generate signatures using someone's private key, and another is the **VERIFY** algorithm to validate signatures by using someone's public key. These two basic algorithms are defined below.

SIGN

The **SIGN** algorithm takes the input (x_i, m) and outputs signatures C_i, where x_i is a private key and m is a message.

VERIFY

The **VERIFY** algorithm takes the input (y_i, m, C_i) to validate the signature C_i, where y_i is the public key and m is the message. The **VERIFY** outputs **accept** if C_i= **SIGN**(x_i, m); otherwise it outputs **reject**, where x_i is the matching private key of the public key y_i.

3.2 Our scheme

Our scheme consists of three phases: Setup phase, proxy authorization phase, and proxy signature generation and verification phase. Those phases are described in the following.

Setup phase

In this phase, the **SETUP** algorithm is used to initialize the following parameters. The **SETUP** algorithm takes a security parameter *l*, and outputs two large primes p and q such that q|(p-1) and q is exponential in *l*, a public element $g \in Z_p^*$ of order q, a public cryptographic hash function $H:\{0,1\}^* \rightarrow Z_q^*$, three public functions $KGEN(k)=g^k \bmod p$, $KGEN_j(k) = y_j^k \bmod p$, and $KTRAN(s, x_j)= s^{x_j} \bmod p$, three public spaces $M= \{0, 1\}^*$, $K= \{0, 1\}^*$, and $F= Z_q^*$, and the certified private-public key pair (x_i, y_i) for the user U_i, where $x_i \in Z_q^*$ and $y_i = g^{x_i} \bmod p$. Let the user U_i's identity be denoted by ID_i.

There are three entities involved in our schemes: An original signer U_O, a proxy signer U_P, and a verifier U_V.

Proxy authorization phase

Suppose that an original signer U_O wants some user U_p to be its anonymous proxy agent. The proxy signer U_p first randomly chooses an anonymous name ID_A and a randomly constructed public key y_A. U_O authorizes U_p to generate proxy signature by exchanging their asymmetric concurrent signatures. The detail of proxy authorization is described in the following steps.

Step 1: U_P first generates the anonymous name ID_A and ID_A's private-public key pair (x_A, y_A), and then generates the promise of Schnorr signature σ_P= (s_A, c_A) and a keystone k.
  (1) Choose an anonymous name $ID_A \in \{0,1\}^*$.
  (2) Choose a random number $x_A \in Z_q^*$ as a private key, and compute $y_A = g^{x_A} \bmod p$ as a public key.
  (3) Generate the promise of Schnorr signature σ_P= (s_A, c_A) and keystone k by running **ISIGN**(y_P, x_P, $m_P||ID_A||y_A$), where m_p denotes the proxy agreement containing some specification for proxy detail, the U_P, and U_O.
  (4) Send the promise of Schnorr signature σ_P= (s_A, c_A) and $m_P||ID_A||y_A$ to U_O securely.

Step 2: After receiving U_P's promise of Schonorr signature σ_P, U_O verifies the promise σ_P. If the promise is correct, U_O generates the promise of Schonorr-like signature on the proxy warrant m_W to authorize U_P.
  (1) Perform **IVERIFY**(σ_P, $m_P||ID_A||y_A$, y_P) to check whether or not the promise of Schonorr signature is generated by U_P. If **IVERIFY**(σ_P, $m_P||ID_A||y_A$, y_P) return **reject**, then abort.
  (2) Generate the promise of Schnorr-like signature ω_O= (s_O, k_1, c_O) by running **SSIGN**(y_O, x_O, $m_W||ID_A||y_A$, s_P), where m_W is the proxy warrant specifying important proxy information.
  (3) Send the ω_O and m_W to U_P.

Step 3: U_P validates the promise of Schnorr-like signature ω_O= (s_O, k_1, c_O), and defines the authorized proxy private and public keys.
  (1) Check whether or not **SVERIFY(**ω_O, $m_W||ID_A||y_A$, y_O)= **accept**.
  (2) If **SVERIFY**(ω_O, $m_W||ID_A||y_A$, y_O)= **accept**, U_P sets the proxy private key as x_A and the proxy public key as y_A.

Finally, the proxy signer obtains the original signer's signature (k, ω_O) on the warrant $m_W||ID_A||y_A$.

Proxy signature generation and verification phase

The proxy signer U_P generates the anonymous proxy signature S_A on a message m by using the proxy private key x_A. Then a verifier uses the proxy public key y_A to validate the proxy signature S_A. The generation and verification of proxy signatures are described in the following steps.

Step 1: U_P computes the proxy signature S_A= **SIGN**(x_A, m).

Step 2: U_P sends {(S_A , m), (k, ω_O, $m_W||ID_A||y_A$)} to a verifier U_V.

Step 3: U_V verifies the original signer U_O's authorization and the proxy signature S_A as below:
  (1) Check whether or not **CVERIFY**(k, ω_O , $m_W||ID_A||y_A$) = **accept**.
  (2) Check whether or not **VERIFY**(y_A, m, S_A) = **accept**.

If both **VERIFY**(y_O, $m_W||ID_A||y_A$, C_O) and **VERIFY**(y_A, m, S_A) return **accept**, the proxy signature S_A is

valid.

Anonymity disclosure phase

When disputes happen, the verifier asks the original signer to disclose the identity of the anonymous proxy signer. Suppose that the given $\{(S_A , m), (k, \omega_O, m_W||ID_A||y_A)\}$ has been validated to be correct. Then the original signer obtains the proxy signer's concurrent signature $(k, \sigma_P= (s_A, c_A))$ on $m_P||ID_A||y_A$. With the help of $(k, \sigma_P= (s_A, c_A))$ on $m_P||ID_A||y_A$, the original signer can show the verifier the proxy signer's information $m_P||ID_P||y_A$ and the proxy signer's concurrent signature $(k, \sigma_P= (s_A, c_A))$. The verifier uses the **CVERIFY** algorithm to check the proxy signer's concurrent signature $(k, \sigma_P= (s_A, c_A))$ on $m_P||ID_P||y_A$. If the proxy signer's concurrent signature $(k, \sigma_P= (s_A, c_A))$ is valid, the proxy signer's anonymity is revoked.

4. Security analysis

The security analysis on our scheme is given below. The basic security assumption of our scheme is that the underlying discrete logarithm based signature scheme and the underlying concurrent signature scheme are secure.

Unforgeability

The proxy signature $S_A= SIGN(x_A, m)$ in our scheme are generated by the underlying discrete logarithm based signature scheme. Therefore, the unforgeability of proxy signatures is guaranteed by the unforgeability of underlying discrete logarithm based signature scheme. In order to forge proxy signatures, the only chance for attacks is to find the proxy private key x_A. However, to find the proxy private key x_A directly from y_A is protected by the discrete logarithm hard problem (DLP for short). So attacks cannot forge proxy signatures.

Verifiability

The proxy signatures in our scheme are verifiable by any verifier. In our scheme, any verifier can validate the promise of Schnorr-like signature ω_O on $m_W||ID_A||y_A$ and keystone k first. Then the verifier can use the certificated proxy public key y_A to verify any proxy signature S_A on the message m by **VERIFY**(y_A, m, S_A).

Proxy signer's deviation

In our scheme, the original signer's private key x_O is never disclosed in the communication. In our scheme, the original signer's private key x_O is only used to generate the promise of Schnorr-like signature ω_O on $m_W||ID_A||y_A$ and keystone k. Due to the underlying Schnorr-like signature scheme is secure, the original signer's private key x_O is never released from the promise of Schnorr-like signature ω_O. Because both the original signer's private key x_O and the underlying discrete-logarithm-based signature scheme are secure, there is no chance for the proxy signer to forge the original signer's signatures. In other words, there is no proxy signer's deviation in our scheme.

Distinguishability

Proxy signatures, proxy signers' signatures, and the original signers' signatures are distinguishable in our scheme. The reason is that those signatures are validated by using different public keys. The proxy signature $S_A =$ **SIGN** (x_A, m) is verified by using the ID_A's public key y_A. On the other hand, the original signers' signature $S_O= SIGN(x_O, m)$ is validated by the original signer's public key y_O while the proxy signers' signature $S_P= SIGN(x_P, m)$ is validated by the proxy signer's public key y_P. These three public keys y_A, y_O, and y_P are different.

Identifiability

The original signer can determine and prove the identity of the proxy signer according to the proxy signatures. Because the promise of Schnorr-like signature ω_O is generated on $m_W||ID_A||y_A$ and keystone k, the proxy signature S_A being successfully validated by **VERIFY**(y_A, m, S_A) is generated by the owner of the proxy private key x_A. To find the owner of x_A, the original signer needs the anonymous name ID_A's promise of signature $\sigma_P= (s_A, c_A)$, keystone k, and the message $m_p||ID_A||y_A$ to bind y_A and the proxy signer U_P together. Since **KGEN**$(k) = s_P$ and **IVERIFY**$(\sigma_P, m_p||ID_A||y_A, y_P)$=accept, the concurrent signature (k, σ_P) is generated by U_P. Therefore the original signer can prove that the owner of x_A is the user U_P by using the concurrent signature (k, σ_P).

Proxy protection

The concurrent signature (k, σ_P) on the message $m_P||ID_A||y_A$ is not only used to prove the proxy signer's identity but also to protect proxy signers. Due to unforgeability of (k, σ_P), only the proxy signer U_P can generate (k, σ_P) on his/her anonymous name ID_A and a proxy public key y_A. So the original signer cannot forge the proxy signer's concurrent signature (k, σ_P) even though the original signer can generate the concurrent signature ω_O on $m_W||ID'_A||y'_A$ alone. The proxy signatures validated by y'_A cannot be falsely incremented to the proxy signer U_P. Therefore, our scheme has proxy protection property.

Undeniability

In our scheme, the proxy signer U_P cannot deny the generation of proxy signatures that were really generated by U_P. The original signer cannot deny the proxy authorization. When any dispute happens, the original signer can use (k, σ_P) to prove that only the proxy signer U_P know the proxy private key x_A. Due to the unforgeability of the underling signature scheme, the proxy signer can't deny the generation of proxy signatures validated by y_A. Due to the unforgeability of the concurrent signature ω_O on the warrant $m_W||ID_A||y_A$, the original cannot deny the proxy authorization.

Anonymity

In our proposed protocol, the proxy signer U_P has an anonymous proxy authorization on the anonymous name ID_A and a randomly-chosen private and public keys x_A and y_A. Due to the randomness of x_A and y_A, no one can find out who is the proxy signer. Therefore, our scheme provides anonymity for proxy signers.

Original Signer's deviation

Only the concurrent signatures $\omega_O = (s_O, k_1, c_O)$ is generated by using proxy signer's private key. Due to the secure underlying concurrent signature schemes, the proxy signer's private key is secure in our scheme. In our scheme, there is no original signer's deviation to forge proxy signer's signature or to disclosure proxy signer's private key.

5. Conclusions

A new practical anonymous proxy signature scheme is proposed. Except satisfying security properties of an anonymous proxy signature scheme, our scheme has some advantages. First of all, our scheme is the first one to utilize concurrent signature schemes to deal with the anonymous certificate and evidence between original signers and proxy signers. Moreover, the anonymous name is known only by the proxy signer until the anonymous name is actually used in proxy authorization. Our scheme does not need trusted alias issuers to provide anonymity for proxy signers while Shum and Wei's scheme needs. Our scheme provides a practical solution of anonymous protection for proxy signers since each proxy delegation can be used to generate proxy signatures many times. On the other hand, in Mehta and Harn's scheme, each anonymous proxy public key is used to generate only one proxy signature.

References

[1] Chen, L., Kudla, C., and Paterson, K. G., "Concurrent signatures," *Eurocrypt '04*, LNCS 3027, New York: Spriger-Verlag, pp. 287-305, 2004.

[2] Hwang, Shin-Jia and Hsu, Pi-Hung, "A practical anonymous proxy signature scheme with trusted alias issuing authority," to appear in *The 3rd Joint Workshop on Information Security*, July 10-11, 2008, Hanyang University, Seoul, Korea.

[3] Hwang, Shin-Jia and Shi, Chi-Hwai, "A proxy signature scheme without using one-way hash functions," *2000 International computer symposium*, Chiayi, Taiwan, R.O.C., Dec. 6-8, pp. 60-64, 2000.

[4] Hwang, Shin-Jia and Shi, Chi-Hwai, "The specifiable proxy signature," *National computer symposium 1999*, Vol. 1334, Taiwan, pp. 190-197, December 1999.

[5] Kim, S., Park, S., and Won, D., "Proxy signatures, revisited," *ICICS '97,* LNCS 1334, New York: Springer, Berlin, pp. 223-232, 1997.

[6] Khanh Nguyen, "Asymmetric concurrent signatures," *Information and Communications Security Conference, ICICS 2005*, Lecture Notes in Computer Science 3783, pp.181-193, Springer-Verlag, 2005.

[7] Li, Li-Hua, Tzeng, Shiang-Feng, and Hwang, Min-Shiang, "Generalization of proxy signature-based on discrete logarithms," *Computers & Security*, Vol. 22, No. 3, pp. 245-255, 2003.

[8] Lee, Narn-Yih, Hwang, Tzonelih, and Wang, Chin Hung, "On Zhang's nonrepudiable proxy signature schemes," *Third Australasian Conference,* ACISP '98, pp. 415-422, 1998.

[9] Lee, Narn-Yih and Lee, Ming-Feng, "The security of a strong proxy signature scheme with proxy signer privacy protection," *Applied Mathematics and Computation*, Vol. 161, pp. 807-812, 2005.

[10] MAMBO, Masahiro, USUDA Keisuke, and OKAMOTO, Eiji, "Proxy signatures: Delegation of the power to sign message," *IEICE. Trans. Fundamentals*, E79-A, 9, pp. 1338-1354, 1996.

[11] MAMBO, Masahiro, USUDA, Keisuke, and OKAMOTO, Eiji, "Proxy signatures for delegation signing operation," *Proc. 3nd ACM Conference on Computer and Communication Security*, pp. 48-57, 1996.

[12] Mehta, M. and Harn L. "Efficient one-time proxy signatures," *IEE Proc.-Commun.*, Vol. 152, No. 2, April 2005

[13] Shum, K. and Wei, Victor K., "A strong proxy signature scheme with proxy signer privacy protection," *Proceedings of the IEEE International Workshops on Enabling Technologies*: *Infrastructure for*

Collaborative Enterprises (WETICE'02), pp. 55-56, 2002.

[14] Wang, Guilin, Bao, Feng, and Zhou, Jianying, "The Fairness of Perfect Concurrent Signatures," *ICICS 06*, LNCS 4307, New York: Spriger- Verlag, pp. 435-451, 2006.