A Time-Stamping Proxy Signature Scheme Using Time-Stamping Service

Eric Jui-Lin Lu[1] and Cheng-Jian Huang[2]
(Corresponding author: Eric Jui-Lin Lu)

Department of Management Information Systems, National Chung Hsing University[1]
250 Kuo Kuang Road, Taichung, Taiwan 402, R.O.C. (Email: jllu@nchu.edu.tw)
Department of Information Management, Chaoyang University of Technology[2]
168 Gifeng E. Rd., Wufeng, Taichung County, Taiwan 413, R.O.C.

(Received May 28, 2005; revised and accepted July 6, 2005)

Abstract

In current proxy signature schemes, an original signer delegates her signing capability to a proxy signer, and then the proxy signer can sign messages on behalf of the original signer. Although these schemes have succeeded in proxy delegations, they share a common problem. That is, a verifier cannot ascertain that a proxy signature was signed by the proxy signer during the valid delegation period. Additionally, these schemes do not support revocations. In this paper, we proposed a new time-stamping proxy signature scheme using time-stamping service to resolve the above problems. The security and performance of the proposed scheme are also analyzed.

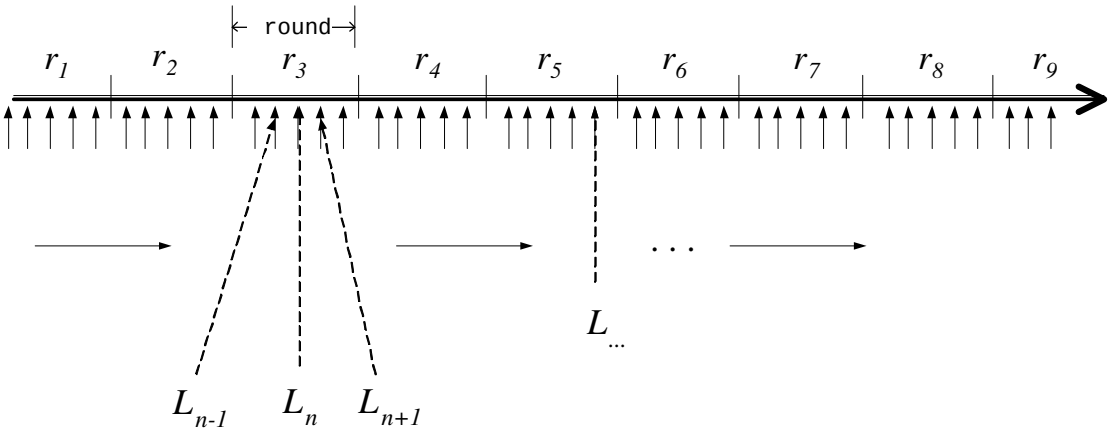
Keywords: Proxy signature, time-stamp, time-stamping service

1 Introduction

A digital signature is like a "hand-written" signature in an electronic form that can be used to authenticate the identity of the signer of a document. In general, the signer uses her secret key to sign messages by using a signature scheme such as ElGamal or Schnorr signature scheme [6, 16, 17]. However, in many situations, the signer cannot sign messages by herself. For instance, the signer is on a business trip or on vacation. Therefore, the signer needs a proxy signer to sign messages on behalf of her.

Currently, there are three types of delegation, namely, full delegation, partial delegation, and delegation by warrant. In the full delegation, the proxy signer is given the same private key that the original signer has. However, this approach does not satisfy the security requirements of proxy signatures. In partial proxy signature schemes [12, 13], the original signer delegates her signing capability to a proxy signer by giving the proxy signer a proxy key. With the proxy key, the proxy signer can sign messages on behalf of the original signer. In cases where the proxy signer abuses her delegated rights, the original signer needs to revoke the proxy signer's signing capability. Currently, the proxy revocation protocols can be classified into two approaches. One approach is to change the public key of the original signer. This approach is impractical because, once the public key of the original signer is changed, all signatures generated earlier by the original signer cannot longer be verified. Even if we can come up with some public key management scheme so that the verification is possible, the tasks to verify all versions of signatures will be extremely complex if the original signer has to revoke delegations from time to time. The other approach is to put r_A (i.e. a part of the proxy key generated by the original signer) on a public revocation list. Any verifier must ensure that r_A is not on the list before verifications. However, these approach has two serious drawbacks. One is, once r_A is posted, all valid proxy signatures generated earlier cannot longer be verified. This also results in difficulties in verifying the validness of proxy signatures generated by the proxy signer even after the proxy key is revoked because the proxy signer can argue that these proxy signatures were generated before the proxy key is revoked. The other drawback is that the size of the revocation list will grow unlimitedly as long as it is required to verify all proxy signatures at all times.

Delegation by warrant is another type of proxy signature schemes [8, 14, 18, 19]. In the delegation by warrant schemes, a proxy warrant is given to the proxy signer to generate proxy signatures. The proxy warrant usually contains the identity of the proxy signer, the valid period of delegation, and some other restrictions on the signing capability delegated to the proxy signer. Therefore, the proxy signer can sign messages on behalf of the original signer only in the valid delegation period. However, delegation by warrant has two major limitations. One limita-



$L_n = H(n, X_n, L_{n-1}, L_{f(n)})$

$r_m = m\text{-}round$

Figure 1: The time-stamp linking scheme

tion is that the declaration of a valid delegation period in the warrant is of no use because no verifier can ensure the exact time when a proxy signature was created. Although Sun [18] claimed that this problem can be resolved in a time-stamped proxy signature scheme he proposed, it had been shown that Sun's scheme is insecure [10]. The other limitation is that, with the warrant, the delegation will be terminated after the delegation period has expired. However, sometimes the original signer must put an end to her delegation earlier than what was planned. Unfortunately, there is no known protocol which allows early termination of delegations.

To resolve all problems stated above, Lu et al. [11] proposed a time-stamping proxy signature scheme with revocation. In their scheme, a trusted third party, called *authentication server*, is utilized. Their scheme ensure that any verifier can check the validness of any proxy signature and, when the verification is successful, it is certain that the proxy signature was issued during the valid delegation period. Also, delegations can be revoked whenever the original signer needs. An important feature is that the size of the revocation list will not grow unlimitedly. Although Lu's scheme resolved the claimed problems, the scheme requires a trusted third party at all times. This security requirement is too high when considering more than 70% of enterprises' servers in US had been attacked [7]. If the authentication server were hacked, verifiers can not longer ascertain the validness of any issued and valid proxy signature. To alleviate the trust level, a time-stamping proxy signature scheme using both time-stamping service (TSS) and Pedersen's threshold cryptosystem [15] is proposed in the paper. By using TSS, it is only required to unconditionally trust a third party (ie. the TSS) during the round when a time-stamp was requested. The validness of all issued and valid proxy signatures will not be affected. The security and performance of the proposed scheme are also analyzed in the paper.

The rest of the paper is structured as follows. In Section 2, the time-stamping service (TSS) proposed by Buldas et al. [3, 4] is first illustrated. In addition, the threshold cryptosystem without a trusted third party proposed by Pedersen [15] are presented. The details of the proposed scheme is discussed in Section 3. To alleviate the unconditional trust on the authentication server, both TSS and the threshold cryptosystem proposed by Pedersen are employed in the new proxy signature scheme. Also, the security and performance of the proposed scheme are analyzed. Finally, we conclude the paper and present possible future research in the Section 4.

2 Related Works

2.1 Time-Stamping Service (TSS)

It is believed that absolute time does not exist. Therefore, it is hard to prove that a proxy signature is created during the valid delegation period. However, as shown in [1, 2, 3, 4, 5], time-stamping in the relative temporal authentication (RTA) can be performed based on the complexity-theoretic assumption on the existence of collision-resistant one-way hash function. In other words, given any two time-stamped documents, RTA enables any verifier to verify the creation order of the two documents.

The concept of the TSS is quite simple. As shown in Figure 1, L_{n-1}, L_n, and L_{n+1} are all time-stamps issued by the TSS. L_n is generated by applying a one-way

hash function H to the concatenation comprising L_{n-1} and a suitably chosen $L_{f(n)}$. As a result, any verifier can validate that L_{n-1} was created earlier than L_n and L_n was created earlier than L_{n+1}. It is generally denoted as $L_{n-1} < L_n < L_{n+1}$.

Traditionally, the verification of any two time-stamps requires $O(N)$ where N is the number of time-stamps issued between these two time-stamps. To improve the performance, Buldas et al. [3, 4] proposed a binary linking scheme such that the time complexity of the verification is reduced to $O(\log N)$.

To be able to verify signed documents in a long period of time, the TSS providers must be trustable. However, as we know, TSS may be damaged or hacked. This results in that all issued time-stamps become invalid. To resolve the problem, after issuing an amount of time-stamps which is called a *round*, all issued time-stamps in one round are publicized. This special feature makes it possible that TSS does not have to be unconditionally trusted all the times because all publicized time-stamps cannot be forged. Therefore, we only have to trust TSS during the round that a time-stamp was requested.

2.2 Review of Pedersen's Threshold Cryptosystem without a Trusted Party Scheme

In 1991, Pedersen [15] proposed a (t, n) threshold cryptosystem scheme which does not require a trusted third party who selects and distributes the secret key. The Pedersen's threshold cryptosystem is illustrated as follows.

2.2.1 Notations

p: a large prime with $2^{511} < p < 2^{512}$.

q: a large prime with $q|p-1$.

g: an order q generator in Z_p^*.

$h()$: a collision resistant hash function.

U_0: the original signer.

U_i: n members of a group $(U_1, ..., U_n)$.

x_i, y_i: the secret/public key pair for user i where $x_i \in Z_q$ and $y_i = g^{x_i} \ mod \ p$.

m: a message.

2.2.2 Selecting and Distributing the Keys

1) U_i $(i = 1, 2, \cdots, n)$ chooses at random a polynomial $f_i(x)$ of degree at most $t-1$ such that $f_i(0) = x_i$. Let

$$f_i(x) \quad = \quad a_{i,0} + a_{i,1}x^1 + \cdots + a_{i,t-1}x^{t-1} \ mod \ q.$$

2) U_i sends $f_i(j)$ secretly to U_j.

3) U_i computes and broadcasts a check vector $CV_i = [g^{a_{i,0}} \ mod \ p, g^{a_{i,1}} \ mod \ p, \cdots, g^{a_{i,t-1}} \ mod \ p]$.

4) U_i verifies that the share $f_j(i)$ received from U_j:

$$g^{f_j(i)} \quad = \quad g^{a_{j,0}} \cdot [g^{a_{j,1}}]^i \cdot [g^{a_{j,2}}]^{i^2} \cdot \cdots \cdot [g^{a_{j,t-1}}]^{i^{t-1}} \ mod \ p.$$

If this fails, U_i broadcasts that an error has been found, and then stops.

5) If all receiving $f_j(i)$ $(1 \le j \le n$ and $j \ne i)$ are correct, then U_i computes his share as the sum of all shares received in step 2.

$$s_i \quad = \quad \sum_{j=1}^{n} f_j(i) \ mod \ q.$$

6) U_i checks the validity of s_i by verifying whether or not the following equation holds.

$$g^{s_i} \quad = \quad [\prod_{j=1}^{n} g^{a_{j,0}}] \cdot [\prod_{j=1}^{n} g^{a_{j,1}}]^i \cdot \cdots \cdot [\prod_{j=1}^{n} g^{a_{j,t-1}}]^{i^{t-1}} \ mod \ p.$$

After finishing all steps stated above, all of the members can get her own share s_i. The group can then obtain a polynomial $f(x)$ of degree at most $t-1$ in a form such that:

$$\begin{aligned} & f(x) \\ = \ & \sum_{j=1}^{n} a_{j,0} + (\sum_{j=1}^{n} a_{j,1})x + \cdots + (\sum_{j=1}^{n} a_{j,t-1})x^{t-1} \ mod \ q, \\ = \ & a_0 + a_1 x + \cdots + a_{t-1}x^{t-1} \ mod \ q, \end{aligned}$$

and $s_i = f(i)$.

2.2.3 Rebuild the Secret

Any t members can obtain the secret $\sum_{j=1}^{n} a_{j,0}$ by Lagrange interpolating polynomial and check the validity by the following equation.

$$g^{\sum_{j=1}^{n} a_{j,0}} \quad = \quad \prod_{j=1}^{n} g^{a_{j,0}} \ mod \ p.$$

3 A Time-Stamping Proxy Signature Using TSS

The proposed proxy signature scheme is based on the discrete logarithm problem. There are five major steps in the proposed scheme which are pictured in Figure 2.

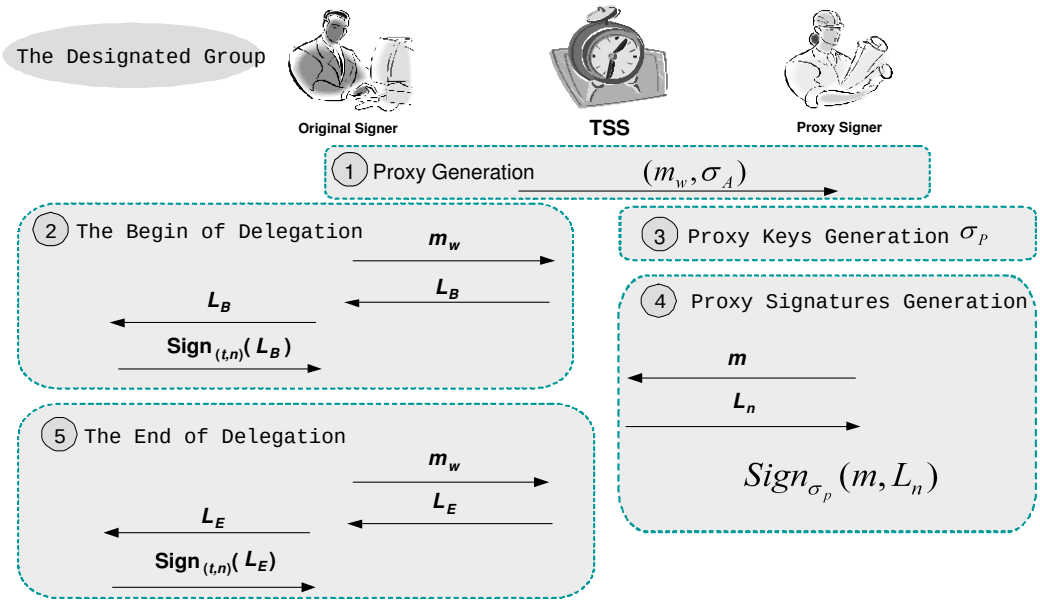


Figure 2: Five steps in the proposed scheme

Proxy Generation: The original signer creates a warrant m_w which contains related information such as the identification of the proxy signer, the delegation period, etc. Also, the original signer generates a signature for m_w and transmits both the signature and m_w to the proxy signer.

The Begin of Delegation: As the delegation becomes effective, the original signer will use m_w to get a time-stamp denoted as L_B from a TSS. Additionally, L_B needs to be signed by any t members of a designated group of size n using Pedersen's threshold cryptosystem. The members of the group can be the supervisors of the original signers or other managers in an organization. Both L_B and its signature will be published. The design of using the TSS and Pedersen's threshold cryptosystem has several advantages:

1) By using the TSS, it is not required to unconditionally trust a third party at all times. One only has to unconditionally trust TSS during the rounds that she requests for time-stamps. Therefore, the trust level can be reduced significantly.
2) Since Pedersen's threshold cryptosystem is used, there is no need for an extra trusted third party. It is noted that, although Pedersen's cryptosystem is used in the proposed scheme, it can be replaced by any (t, n) threshold cryptosystem without a trusted third party.
3) The original signer cannot maliciously create another $L_{B'}$ and later claim that $L_{B'}$ represents the begin of delegation. This is because L_B has to be verified and signed by the designated group.

Proxy Keys Generation: After the proxy signer received the warrant and its signature from the original signer, she will first verify the signature. If the verification is successful, the proxy signer will use them to generate the proxy keys.

Proxy Signatures Generation: After L_B and its signature are published, the proxy signer can start generating proxy signatures for documents. Before generating a proxy signature for a document m, the proxy signer sends m to the TSS and the TSS issues a time-stamp L_n for m. After receiving L_n, the proxy signer will sign on m and L_n.

The End of Delegation: After the delegation has expired or been revoked, the original signer uses m_w to get a time-stamp L_E from the TSS and requests any t members of the designated group to sign on L_E. This is to prevent the original signer from maliciously creating a time-stamp $L_{E'}$ which can be either $L_{E'} < L_E$ or $L_E < L_{E'}$. In either cases, any valid document signed between $L_{E'}$ and L_E may become invalid.

3.1 Notations

In the proposed proxy signature scheme, we shall use the following notations.

x_A: the private key of the original signer.

y_A: the public key of the original signer.

x_B: the private key of the proxy signer.

y_B: the public key of the proxy signer.

p: a large prime with $2^{511} < p < 2^{512}$.

q: a large prime with $q|p-1$.

g: an order q generator in Z_p^*.

$H()$: a collision resistant hash function.

3.2 Basic Protocol

The details of the proposed proxy signature scheme are described as follows:

1) (Proxy generation) The original signer generates a random number $k_A \in_R Z_{p-1}$ and computes the followings:

$$\begin{aligned} r_A &= g^{k_A} \bmod p, \text{ and} \\ \sigma_A &= k_A + x_A H(m_w, r_a) \bmod q. \end{aligned}$$

2) (Proxy delivery) The original signer sends (m_w, r_A, σ_A) to the proxy signer through a secure channel.

3) (Proxy Verification) The proxy signer checks the validity of (m_w, r_A, σ_A) by verifying whether or not the following equation holds.

$$g^{\sigma_A} \equiv r_A y_A^{H(m_w, r_A)} \bmod p.$$

4) (Alteration of the proxy) If the verification is successful, the proxy signer then computes an alternative proxy private key σ_p and public key y'_p such that

$$\begin{aligned} \sigma_p &= \sigma_A + x_B H(m_w, r_A) \bmod q, \text{ and} \\ y'_p &= g^{\sigma_p} = r_A (y_A y_B)^{H(m_w, r_A)} \bmod p. \end{aligned}$$

5) (The begin of delegation) The original signer sends m_w to a TSS and requests for a time-stamp. The TSS generates the time-stamp L_B such that

$$L_B = H(n, m_w, L_{B-1}, L_{f(B)}).$$

And the TSS sends L_B to the original signer. Any t members of a group of size n can construct digital signatures on behalf of the group. The group signature on the message L_B is $Sign_{(t,n)}(L_B)$. The original signer makes the (m_w, L_B) to the public.

6) (Proxy signature generation) The proxy signer uses σ_p to perform an ordinary signing operation.

7) (Time-stamp request) The proxy signer sends a message m to the TSS and requests a time-stamp.

8) (Time-stamp generation and delivery) The TSS generates a time-stamp L_n such that

$$L_n = H(n, m, L_{n-1}, L_{f(n)}).$$

And the TSS sends L_n to the proxy signer.
Finally, the valid proxy signature for the message m is
$(m, m_w, r_A, Sign_{\sigma_p}(m, L_n), L_n)$.

9) (The end of delegation) When the delegation is expired or revoked by the original signer, the original signer sends the warrant m_w to the TSS and requests a time-stamp. The TSS generates a time-stamp L_E such that

$$L_E = H(n, m_w, L_{E-1}, L_{f(E)}).$$

And the TSS sends L_E to the original signer.
Any t members of the group can again construct a digital signature for L_E. The group signature on L_E is $Sign_{(t,n)}(L_E)$. The original signer makes the (m_w, L_B, L_E) to the public.

10) (Proxy signature verification) Any verifier can use the same verification procedures of the original signing scheme to check the validness of $Sign_{\sigma_p}(m, L_n)$. Furthermore, the verifier has to check whether or not the following equations hold.

$$y'_p = g^{\sigma_p} = r_A (y_A y_B)^{H(m_w, r_A)} \bmod p.$$

11) (Time-stamp Verification) The time-stamp of the proxy signature also needs to check to make sure $L_B < L_n < L_E$. If the verification is successful, it is ensured than the proxy signature is created during the valid delegation period as shown in Figure 3.

3.3 Security Analysis

In this section, we shall analyze that the proposed proxy signature scheme not only satisfies all the security requirements of proxy signatures stated in [9, 12], but also resolves the time-stamping issues without having to unconditionally trust a third party at all times.

- Verifiability: Any verifier can be convinced of the agreement of the original signer on the signed message from its corresponding proxy signature. In the proposed scheme, the proxy signature consists of (m, m_w, r_A, $Sign_{\sigma_p}(m, L_n)$). From the warrant m_w and r_A generated by the original signer, any verifier can be convinced of the original signer's agreement on the signed message. Additionally, any verifier can ascertain that the proxy signature was created during the valid delegation period by checking $L_B < L_n < L_E$.

- Strong Unforgeability: Strong unforgeability requires that only the proxy signer can create a valid proxy signature. Even the original signer cannot create a valid proxy signature. Because the generation of a proxy signature requires the private key x_B of the proxy signer in the proposed scheme, no one can forge a valid proxy signature.

- Strong Identifiability: Anyone can determine the identity of the proxy signer from the proxy signatures created by her. In the proposed scheme, anyone can identify the identity of the proxy signer from the warrant m_w.

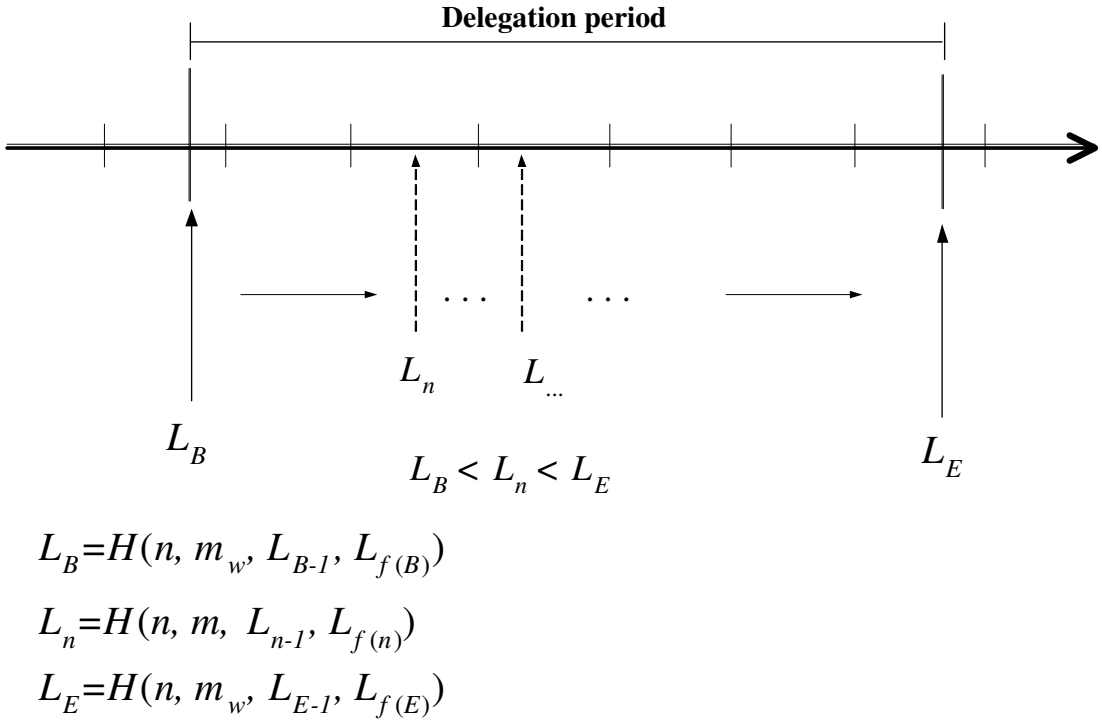


Figure 3: The verification of time-stamps

- Strong Undeniability: This requirement states that the proxy signer cannot repudiate the valid proxy signatures created by her. This is also called "non-repudiation" in some literatures. Since a proxy signature is created by using the proxy signer's private key x_B, the proxy signer cannot disavow the proxy signature she created.

- The Validation of Time-Stamped Proxy Signatures: In traditional proxy signature schemes, it is not possible to verify whether or not a proxy signature was issued during the valid delegation period. In the proposed scheme, for a specific delegation defined in a warrant m_w, since both L_B and L_E are verified and signed by a designated group, and since any verifier can validate $L_B < L_n < L_E$ where L_n is the time-stamp for a signed document m, it is ensured that the proxy signature for m was issued during the valid delegation period.

- Prevention of Misuse: It should be confident that the proxy key pair cannot be used for other purposes. In the case of misuse, the responsibility of the proxy signer should be determined explicitly. Because the delegation rights are clearly stated in the warrant and the warrant is signed by the original signer, the proxy signer is only allowed to do whatever were delegated. Also, since the time-stamp L_n for a proxy signature must be generated between L_B and L_E, there is no way for the proxy signer to sign on a document when the delegation has expired.

  It is also likely that the original signer is malicious. She may wish to create invalid time-stamps L_{B_2}, L_{E_1}, and L_{E_2} such that $L_B < L_{B_2}$ or $L_{E_1} < L_E < L_{E_2}$; and publish either one of them instead of the valid time-stamps L_B and L_E. In the case that L_{B_2} were published, all valid proxy signatures issued between L_B and L_{B_2} become invalid. Similarly, in the cases that either L_{E_1} or L_{E_2} were published, all valid proxy signatures issued either between L_{E_1} and L_E or between L_E and L_{E_2} become invalid; respectively. In the proposed scheme, since both L_B and L_E have to be verified and signed by a designated group, the original signer cannot arbitrarily publish time-stamps such as L_{B_2}, L_{E_1}, and L_{E_2}.

3.4 Performance Analysis

In this section, we show the communication overhead and computational complexity of the proposed scheme. To analyze the computation complexity and communication overhead, we fist define the following notations. The results of the performance analysis are summarized in Table 1 and Table 2.

T_{exp}: The time for computing modular exponentiation.

T_{mpy}: The time for computing modular multiplication.

T_h: The time for computing one way hash function.

$|x|$: The length of a message x. For example, x can be a warrant or a time-stamp.

$T_{Sign}(m)$: The time for generating a proxy signature for the message m.

Table 1: The summary of computational complexities

Steps	Original Signer	Proxy Signer	TSS	Group	Verifier
Proxy generation	T_{exp}+T_{mpy}+T_h				
Proxy delivery					
Proxy verification		$T_{mpy} + 2T_{exp}$			
Alternation of the proxy		$T_h + 4T_{mpy} + 2T_{exp}$			
The begin of delegation			T_h	T_{PTN}	
Proxy signature generation		$T_{Sign}(m, L_n)$			
Time-stamp generation			T_h		
The end of delegation			T_h	T_{PTN}	
Proxy signature verification					$2T_{exp} + 3T_{mpy} + T_h + T_{Verify}(m, L_n)$
Time-stamp verification					T_{TSV}
Total	T_{exp}+T_{mpy}+T_h	$5T_{mpy} + 4T_{exp} + T_h + T_{Sign}(m, L_n)$	$3T_h$	$2T_{PTN}$	$2T_{exp} + T_h + 3T_{mpy} + T_{TSV} + T_{Verify}(m, L_n)$

$T_{Verify}(m)$: The time for verifying a proxy signature of the message m.

T_{PTN}: The time for using Pedersen's (t, n) threshold cryptosystem to sign a message.

$|PTN|$: The communication overhead of using Pedersen's (t, n) threshold cryptosystem to sign a message.

T_{TSV}: The time for verifying a time-stamp.

$|TSV|$: The communication overhead verifying a time-stamp.

The communication overhead and computational complexity of the proposed scheme are analyzed in detail as follows:

1) (Proxy generation): The original signer needs to compute the followings:

$$r_A = g^{k_A} \bmod p, \text{ and}$$
$$\sigma_A = k_A + x_A H(m_w, r_a) \bmod q.$$

The computational complexity is $T_{exp} + T_{mpy} + T_h$.

2) (Proxy delivery): The original signer sends (m_w, r_A, σ_A) to the proxy signer. The communication overhead is $|m_w| + |r_A| + |\sigma_A|$.

3) (Proxy verification): The proxy signer needs to check the validity of (m_w, r_A, σ_A) by verifying whether or not the following equation holds.

$$g^{\sigma_A} \equiv r_A y_A^{H(m_w, r_A)} \bmod p.$$

The computational complexity is $T_{mpy} + 2T_{exp}$.

4) (Alteration of the proxy): The proxy signer needs to compute an alternative proxy private/public key pair such as

$$\sigma_p = \sigma_A + x_B H(m_w, r_A) \bmod q, \text{ and}$$
$$y'_p = g^{\sigma_p} = r_A (y_A y_B)^{H(m_w, r_A)} \bmod p.$$

As a result, the computational complexity is $T_h + 4T_{mpy} + 2T_{exp}$.

5) (The begin of delegation): The original signer sends the m_w to a TSS and requests a time-stamp. The TSS generates a time-stamp L_B such that

$$L_B = H(n, m_w, L_{B-1}, L_{f(B)}).$$

Then the TSS sends L_B to the original signer. Since L_B also has to be verified and signed by any t members of the designated group, the computational complexity is $T_h + T_{PTN}$ and the communication is $|m_w| + |L_B| + |PTN|$.

6) (Proxy signature generation): The proxy signer uses σ_p to execute an ordinary signing operation. The computational complexity is $T_{Sign}(m, L_n)$.

7) (Time-stamp request): The proxy signer sends a message m to the TSS and requests a time-stamp. The communication overhead is $|m|$.

8) (Time-stamp generation and delivery): The TSS generates a time-stamp L_n for the message m such that

$$L_n = H(n, m, L_{n-1}, L_{f(n)}).$$

And the TSS sends the L_n to proxy signer. Therefore, the computational complexity is T_h and the communication overhead is $|L_n|$.

9) (The end of delegation): When the delegation is expired and revoked by the original signer, the original signer sends the warrant m_w to the TSS and requests a time-stamp. The TSS generates a time-stamp L_E such that

$$L_E = H(n, m_w, L_{E-1}, L_{f(E)}).$$

Then the TSS sends L_E to the original signer. Because L_E also has to be verified and signed by any

Table 2: The summary of communication overhead

Steps	Communication Overhead
Proxy generation	
Proxy delivery	$\|m_w\| + \|r_A\| + \|\sigma_A\|$
Proxy verification	
Alternation of the proxy	
The begin of delegation	$\|m_w\| + \|L_B\| + \|PTN\|$
Proxy signature generation	
Time-stamp request	$\|m\|$
Time-stamp generation and delivery	$\|L_n\|$
The end of delegation	$\|m_w\| + \|L_E\| + \|PTN\|$
Proxy signature verification	
Time-stamp verification	$\|TSV\|$
Total	$3\|m_w\| + \|m\| + \|r_A\| + \|\sigma_A\| + \|L_B\| + \|L_n\| + \|L_E\| + 2\|PTN\| + \|TSV\|$

t members of the designated group, the computational complexity is $T_h + T_{PTN}$ and the communication overhead is $|m_w| + |L_E| + |PTN|$.

10) (Proxy signature verification): Any verifier can verify the proxy signature with the following equation:

$$y'_p \quad = \quad g^{\sigma_p} = r_A(y_A y_B)^{H(m_w, r_A)} \bmod p.$$

Thus, the computational complexity is $2T_{exp} + 3T_{mpy} + T_h + T_{Verify}(m, L_n)$.

11) (Time-stamp verification): Any verifier can verify a time-stamp L_n to ensure that $L_B < L_n < L_E$ holds. The communication overhead and computational complexity are $|TSV|$ and T_{TSV}, respectively.

4 Conclusions and Future Works

It is known that it is not possible to verify whether or not a proxy signature was generated during the valid delegation period for all traditional proxy signature schemes. Although Sun attempted to resolve this problem, it had been shown that the proxy signature scheme proposed by Sun is not secure. In this paper, we proposed a time-stamping proxy signature scheme using TSS. By utilizing both TSS and Pedersen's threshold cryptosystem, the proposed scheme not only support revocation, but also resolve the timing issue existed in almost all traditional proxy signatures schemes. Additionally, it is not required to unconditionally trust a third party at all times. However, there is at least one issue that may worth further investigation. In the proposed scheme, a proxy signer has to request a time stamp for every message she signs. This is costly. It will improve the performance significantly if the number of times to request time-stamps can be reduced.

Acknowledgments

This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no.: NSC93-2213-E-005-035.

References

[1] D. Bayer, S. Haber, and W. Scott Stornetta, "Improving the efficiency and reliability of digital time-stamping," in *Sequences'91: Methods in Communication, Security, and Computer Science*, pp. 329–334, 1992.

[2] J. Benaloh and M. de Mare, *Efficient Broadcast Time-stamping*, Technical Report 1, Clarkson University Department of Mathematics and Computer Science, 1991.

[3] A. Buldas and P. Laud, "New linking schemes for digital time-stamping," in *Proceedings of the 1st International Conference on Information Security and Cryptology*, pp. 3–14, 1998.

[4] A. Buldas, P. Laud, H. Lipmaa, and J. Villemson, "Time-stamping with binary linking schemes," in *CRYPTO'98*, LNCS 1462, pp. 486–501, 1998.

[5] S. Haber and W. S. Stornetta, "Secure names for bit-strings," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pp. 28–35, 1997.

[6] M. S. Hwang, I. C. Lin, and Eric J. L. Lu, "A secure nonrepudiable threshold proxy signature scheme with known signers," *International Journal of Informatica*, vol. 11, no. 2, pp. 1–8, 2000.

[7] Computer Security Institue, *2002 CSI/FBI Computer Crime and Security Survey*, Tech. Rep., Computer Security Institue, 2002.

[8] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," in *Proceedings of International Conference on Information and Communications Security*, LNCS 1334, pp. 223–232, 1997.

[9] B. Lee, H. Kim, and K. Kim, "Strong proxy signature and its applications," in *Proceedings of the 2001 Symposium on Cryptography and Information Security (SCIS 2001)*, pp. 603–608, 2001.

[10] Eric J. L. Lu and C. J. Huang, "Cryptanalysis of a time-stamped proxy signature scheme," *International Journal of Computational and Numerical Analysis and Applications*, vol. 5, no. 2, pp. 106–115, 2004.

[11] Eric J. L. Lu, M. S. Hwang, and C. J. Huang, "A new proxy signature scheme with revocation," *Applied Mathematics and Computation*, vol. 161, no. 3, pp. 799–806, 2005.

[12] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: Delegation of the power to sign messages," *IEICE Transactions on Fundamentals*, vol. E79-A, pp. 1338–1354, Sep. 1996.

[13] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operation," in *Proceedings of 3rd ACM Conference on Computer and Communications Security*, pp. 48–57, 1996.

[14] B. C. Neuman, "Proxy-based authorization and accounting for distributed systems," in *Proceedings of the 13th International Conference on Distributed Computing Systems*, pp. 283–291, 1993.

[15] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *Advances in Cryptology, CRYPTO'91*, pp. 522–526, 1991.

[16] B. Schneier, *Applied Cryptography*, New York: Wiley, 1996.

[17] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, pp. 161–174, 1991.

[18] H. M. Sun, "Design of time-stamped proxy signatures with traceable receivers," *IEE Proceedings of Computers and Digital Techniques*, vol. 147, no. 6, pp. 462–466, 2000.

[19] V. Varadharajan, P. Allen, and S. Black, "An analysis of the proxy problem in distributed systems," in *Proceedings of 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 255–275, 1991.

Eric Jui-Lin Lu received his B.S. degree in Transportation Engineering and Management from National Chiao Tung University, Taiwan, ROC, in 1982; M.S. degree in Computer Information Systems from San Francisco State University, CA, USA, in 1990; and Ph.D. degree in Computer Science from University of Missouri-Rolla, MO, USA, in 1996. During 1997-2004, he was a professor of the Department of Information Management and had served as Director of Computer Center and Head of Graduate Institute of Networking and Communication Engineering at Chaoyang University of Technology, Taiwan, ROC. He is currently a professor of the Department of Management Information Systems at National Chung Hsing University, Taiwan, ROC. His current research interests include XML technology, distributed processing, and security.

Cheng-Jian Huang is currently a software engineer at Motech Taiwan Automatic Corp. He received his B.S.B. degree in Information Management from National Taichung Institute Of Technology, Taiwan in 1994, and M.S. degree in Information Management from Chaoyang University of Technology, Taiwan in 2005. His research interests include Cryptograph and Information Security.