A Practical (t, n) Threshold Proxy Signature Scheme Based on the RSA Cryptosystem

Min-Shiang Hwang, *Member*, *IEEE*, Eric Jui-Lin Lu, and Iuon-Chang Lin

Abstract—In a (t, n) threshold proxy signature scheme, the original signer delegates the power of signing messages to a designated proxy group of n members. Any t or more proxy signers of the group can cooperatively issue a proxy signature on behalf of the original signer, but $(t-1)$ or less proxy signers cannot. Previously, all of the proposed threshold proxy signature schemes have been based on the discrete logarithm problem and do not satisfy all proxy requirements. In this paper, we propose a practical, efficient, and secure (t, n) threshold proxy signature scheme based on the RSA cryptosystem. Our scheme satisfies all proxy requirements and uses only a simple Lagrange formula to share the proxy signature key. Furthermore, our scheme requires only 5 percent of the computational overhead and 8 percent of the communicational overhead required in Kim's scheme.

Index Terms—Lagrange interpolating polynomial, RSA cryptosystem, threshold proxy signature.

1 Introduction

In traditional digital signature schemes [8], [11], [22], a legal user can use her/his private key to sign a message. Any recipient of the signed message can verify the signature by using the signer's public key. However, when the original signer is traveling on business or on vacation and wishes to delegate the power of signing messages to a designated person during her/his absence, traditional signature schemes would see their limit. Fortunately, the proxy signature technology provides a good solution to this problem. The technology allows proxy signers to sign messages on behalf of the original signer without exposing the original signer's private key.

In the history of proxy signature technological development, the $(1, n)$ threshold proxy signature technique was the first to come. In $(1, n)$ proxy signature schemes [16], [17], [24], a legal proxy signature can be generated by a designated proxy signer by using a proxy signing key. The proxy signing key is computed from the original signer's private key, but the private key should not be computed from the proxy signing key in any way. In the eye of a modern user, such schemes are simple but not flexible.

In order to extend proxy signature schemes to fit various practical situations, many (t, n) threshold proxy signature schemes have been proposed [13], [15], [23], [26]. For example, we have (t, n) threshold proxy signature schemes that allow any t or more proxy signers from a designated group of n members to cooperatively sign messages while $(t-1)$ or less members cannot generate the legal proxy signature. In practice, the original signer can flexibly choose the threshold t. The approach agrees with $(1, n)$, (t, n), and (n, n) threshold delegations. Moreover, a practical and secure threshold proxy signature scheme must satisfy the following requirements [13], [23], which we shall call proxy requirements throughout the rest of the paper:

- *Secrecy.* The original signer's private key is very important. It must be kept secret. If it is discovered, the security of the system is ruined. Therefore, the system must ensure that the private key never gets derived from any information such as the sharing of the proxy signing key or the original signer's public key. Furthermore, no proxy signers should be able to cooperatively derive the original signer's private key.
- *Proxy protected.* Only a delegated proxy signer can generate his partial proxy signature. Even the original signer cannot masquerade as a proxy signer to generate a partial proxy signature. This property protects the authority of the proxy signer.
- *Unforgeability.* A valid proxy signature can only be cooperatively generated by t or more proxy signers. Nondelegated signers have no capability to generate a valid proxy signature. Also, $(t-1)$ or less proxy signers have no capability of forging a valid proxy signature.
- *Nonrepudiation.* Any valid proxy signature must be generated by t or more proxy signers. The verifier can make sure that the signed message is a correct one by using the proxy signing keys. The original signer cannot deny having delegated the power of signing messages to the proxy signers. Furthermore, the proxy signers cannot deny that they haved signed the message.
- *Time constraint.* The proxy signing keys can be used only during a stipulated period. Once expired, proxy signing keys become invalid; as a result, the signing capability of the proxy signers disappears. However,

- *M.-S. Hwang and E.J.-L. Lu are with the Department of Information Management, Chaoyang University of Technology, 168, Gifend E. Rd., Wufeng, Taichung County, Taiwan 413, R.O.C. E-mail: {mshwang, jlu}@cyut.edu.tw.*
- *I.-C. Lin is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chaiyi, Taiwan, 62107 R.O.C. E-mail: iclin@cs.ccu.edu.tw.*

Manuscript received 13 Nov. 2000; revised 10 Oct. 2001; accepted 22 Aug. 2002.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 113136.

the original signer's private key can be repeatedly used. This is more suitable for use in the real world.
- *Known signers.* For internal auditing purposes, the system is able to identify the actual signers in the proxy group.

Among the best-known threshold proxy signature schemes [13], [15], [23], [26], only Sun's scheme [23] satisfies all of the requirements described above. However, Sun's scheme uses a verifiable secret sharing scheme [20] to share a random number among a group. Therefore, it is necessary for the proxy group to perform several modular exponential computations and communications to obtain and verify a shared secret. Furthermore, Hwang et al. [12] have shown that Sun's scheme has a security weakness. An adversary can impersonate a legal proxy signer to generate a proxy signature and the real proxy signer cannot deny having signed the proxy signature.

In this paper, we shall propose a new (t, n) threshold proxy signature scheme based on the RSA cryptosystem. Our new scheme only requires the Lagrange formula [5], [10] to share the proxy signing key. The RSA cryptosystem is a popular security technique. Currently, it is widely used in digital signatures, e-commerce, and secure communication. We can easily apply the regular RSA cryptosystem to the proposed scheme without using extra cryptographic techniques. Furthermore, the pair of RSA keys can be repeatedly used. Each user only holds one pair of RSA keys to do e-commerce or secure communication or to carry out the threshold proxy signature scheme. Moreover, our scheme can fully satisfy the proxy requirements, and the overhead is smaller than those of the existing schemes. Therefore, the proposed scheme is practical and efficient.

The organization of the paper is as follows: In the next section, we will briefly review related work on threshold proxy signature schemes. These reviewed schemes do not yet satisfy all of the proxy requirements. In Section 3, we propose a new (t, n) threshold proxy signature scheme based on the RSA cryptosystem. In Section 4, the security of our proposed scheme is analyzed and we compare our scheme with related work. In Section 5, we provide same possible applications. Finally, we summarize the benefits of our scheme in the last section of the paper.

2 Related Works

Generally speaking, proxy signature schemes are classified into three delegation types: *full delegation*, *partial delegation*, and *delegation by warrant* [13]. In full delegation schemes, the original signer gives her/his signing key to a proxy signer. Therefore, when the proxy signer signs a message, she/he uses a signature identical to the one created and held by the original signer. This full delegation approach is not secure since the original signer has to reveal her/his private key to the proxy signer. If the proxy signer uses the private key to sign a message against the will of the original signer, then the original signer still has to take all the responsibility because she/he cannot prove that the signature is issued by the proxy signer.

In the partial delegation schemes [16], [17], [24], a proxy signer is given a proxy signing key which is computed from the original signer's private key. The proxy signer has the capability to sign on behalf of the original signer, but from the proxy signing key the proxy signer cannot recover the original signer's private key.

The idea of delegation by warrant in proxy signature schemes was first proposed in [18], [25]. The warrant contains important information such as the validity period of the delegation, the identities of the original signer and the proxy signers, and the generated proxy verification key. The original signer signs the warrant with her/his private key and publishes the signature on the warrant. Then, the original signer delivers the proxy signing key to the designated proxy signers in a secure manner. The proxy signer can use the proxy signing key to sign a message on behalf of the original signer. According to the warrant, any verifier can verify whether or not the signature has expired or is signed by a legal proxy signer.

Compared with the full delegation approach, partial delegation and delegation by warrant are more secure approaches. To be more specific, the partial delegation approach usually has a higher processing speed, but delegation by warrant is more practical than partial delegation. For example, the signing capability of the proxy signer should disappear when the proxy period expires. However, the partial delegation approach cannot achieve the goal efficiently.

Recently, Kim et al. have combined partial delegation with delegation by warrant and proposed two new types of delegation systems called *partial delegation with warrant* and (t, n) *threshold delegation* [13]. In the partial delegation by warrant approach, the proxy signing key is computed from both the original signer's private key and the warrant. However, it is very difficult to recover the original signer's private key from the proxy signing key and the warrant. In the (t, n) threshold delegation approach, the partial delegation approach is further extended. The technology allows an original signer not to only delegate the power of proxy signing to a proxy signer but also to delegate the power to a group of n proxy signers. Furthermore, the original signer can choose the threshold value t freely within the range $1 <= t <= n$. Only t or more proxy signers of the group can cooperatively create a valid proxy signature. Therefore, the threshold proxy signature approach is more practical and secure than other types of proxy signature schemes.

The concept of threshold cryptosystems was brought up by Desmedt and Frankel [6]. They adapted the ElGamal [8] public key cryptosystem and used Lagrange interpolation or geometry to produce the shadows. Kim et al. also built their proxy signature scheme upon the concept of threshold cryptosystems. So far, to the best of our knowledge, all existing (t, n) threshold proxy signature schemes [13], [15], [23], [26] are based on the discrete logarithm and require a secret random number to be shared among the proxy group members. Among these threshold signature schemes, Kim et al.'s scheme is augmented with the nonrepudiation property so that any verifier can identify the proxy group, but Zhang's scheme [26] is not. Furthermore, Kim proposed two types of threshold proxy signature schemes, which were the proxy-protected scheme and the proxy-unprotected scheme. In the proxy protected scheme, the original

signer cannot impersonate a proxy signer to issue a valid proxy signature. The proxy signing key combines the original signer's secret sharing key and a secret value among the t proxy signers. Therefore, the original signer cannot obtain the proxy signing keys. This property is called proxy-protected. One major drawback in Kim et al.'s scheme is that the actual signers cannot be identified. This can be very inconvenient for internal auditing.

In order to remedy the problem of unknown signers, Sun [23] revised Kim et al.'s proxy-protected type threshold proxy signature scheme and made the actual signers able to be identified. Though Sun's scheme satisfies all proxy requirements listed in Section 1, it has been proven that the scheme is insecure since any $n-1$ proxy signers in the group can conspire to obtain the secret key needed by the remainder of the group [12]. Also, the computational and communicational overhead of Sun's scheme is high. With t or more proxy signers needed to cooperatively issue a proxy signature, they have to generate and share a random number among them, and that requires several expansion modular exponential computations and communications [20], [21].

In this paper, we propose a new (t, n) threshold proxy signature scheme based on the RSA cryptosystem. In the typical RSA digital signature scheme [22], a RSA modulus $N = p \times q$ is publicly known, where p and q are two secret large primes. The security of RSA makes it difficult to factor N into p and q. A signer randomly chooses a private signing key d and computes its corresponding public verification key e such that $d \times e = 1 \; mod \; \phi(N)$, where $\phi(N) = (p-1) \times (q-1)$. For a simple (n, n) proxy signature approach [3], [9] based on the RSA cryptosystem, the original signer randomly selects the proxy signing keys d_i, such that $\sum_{i=1}^{n}(d_i) \; mod \; \phi(N) = d$. Then, the original signer delivers d_i to each proxy signer i. To sign a message m, the partial proxy signature s_i of m is created by the proxy signer i, where $s_i = m^{d_i} \; mod \; N$. Then, a combiner, who may be the secretary of the original signer, collects s_i, and obtains the proxy signature S from $S = \prod_{i=1}^{n}(s_i) \; mod \; N$. The proxy signers sign messages without revealing the proxy signing key d_i. However, this method is not flexible since it always requires n out of n proxy signers to cooperatively issue a proxy signature on behalf of the original signer. Also, the method does not fully satisfy the proxy requirements.

In 1991, Denmedt and Frankel proposed a threshold RSA signature scheme [7]. This technique allows t out of n individuals to generate a signature for a message. The signature is on behalf of the group of n members, hence, we also call it group signature. In this paper, we extend the concepts and principles from Denmedt and Frankel's threshold RSA signature scheme to develop a threshold RSA proxy signature scheme. Our scheme allows t out of n proxy signers to generate a signature on behalf of the original signer. Furthermore, our scheme satisfies all of the proxy requirements and is more efficient.

3 A New (t, n) Threshold Proxy Signature Scheme Based on the RSA Cryptosystem

In this section, we describe a new (t, n) threshold proxy signature scheme based on the RSA cryptosystem. There are three types of participants in our scheme: the original signer, the n proxy signers, and a combiner. The original signer allows a group of n proxy signers to sign a message. The combiner can be the secretary of the original signer.

The proposed threshold proxy signature scheme can be divided into three phases:

1. *the proxy sharing phase*,
2. *the proxy signature issuing phase*, and
3. *the verification phase*.

In the proxy key generation phase, the original signer computes the partial proxy signing keys from his private key and sends them to each designated proxy signer. In the proxy signature issuing phase, the proxy signers cooperatively create a valid signature on a message M. In the verification phase, the verifier can identify not only the original signer, but also the actual signers. For the rest of this paper, P_0 stands for the original signer and $P_1, P_2, \cdots, P_n$ stand for the n proxy signers. N_i is a public RSA modulus for P_i such that $N_i = p_i \times q_i$, where p_i and q_i are two secret large primes. Let d_i be a private key for P_i and its corresponding public key be e_i, such that $d_i \times e_i = 1 \mod \phi(N_i)$, where $\phi(N_i) = (p_i - 1) \times (q_i - 1)$. The parameters e_i and N_i can be published. The parameters d_i and $\phi(N_i)$ are kept secret by the holder. $[M]^{d_i} \; mod \; N_i$ represents M signed with P_i's private key d_i, and $[M]^{e_i} \; mod \; N_i$ represents M encrypted with P_i's public key e_i using the ordinary RSA cryptosystem. The message m_w stands for a warrant that is minted by the original signer and it contains important information such as the validity period of the proxy key, the identities of the proxy signers, and the original signer, etc. The details of the new (t, n) threshold proxy signature scheme are described as follows.

3.1 The Proxy Sharing Phase

Assume that an original signer P_0 delegates the power to sign messages to n members during a stipulated period. The steps to generate the proxy key are as follows:

1. (Proxy generation). P_0 generates the group proxy signature key D and its corresponding proxy verification key E, where

$$D = d_0^{m_w} \mod \phi(N_0) \; and \tag{1}$$

$$E = e_0^{m_w} \mod \phi(N_0). \tag{2}$$

   Then, P_0 publishes $\{m_w, \; E, \; [m_w \parallel E]^{d_0} \; mod \; N_0\}$.
2. (Proxy sharing). P_0 randomly generates a secret polynomial f of degree $t-1$, of the form

$$f(x) = D + a_1 X + \cdots + a_{t-1} X^{t-1} \mod \phi(N_0), \tag{3}$$

   where $a_1, a_2, \ldots, a_{t-1}$ are random numbers. The original signer P_0 computes P_i's partial proxy signing key, $k_i = f(i)$ and sends $[[k_i]^{d_0} \; mod \; N_0 \| k_i]^{e_i} \; mod \; N_i$ to

the proxy signer P_i, where i is the user's identity and for all $L_i \in Z$.

3. (Proxy share generation). After receiving

$$[[k_i]^{d_0} \ mod \ N_0 \| k_i]^{e_i} \ mod \ N_i,$$

each proxy signer can decrypt the ciphertext to obtain $\{[k_i]^{d_0} \ mod \ N_0, \ k_i\}$. Then, each proxy signer P_i can confirm the validity of k_i and keep it secret.

3.2 The Proxy Signature Issuing Phase

Assume that any t or more proxy signers out of the designated group of n members want to cooperatively sign a message M on behalf of P_0. Let T denote these group members. The steps of the proxy signature issuing phase are listed as follows:

1. Each member of T signs the message M with his partial proxy signing key k_i, where $i \in T$. The partial proxy signature s_i for each actual proxy signer P_i is computed as

$$s_i \ = \ M^{(L_i \times k_i)} \ \bmod \ N_0, \tag{4}$$

where

$$L_i \ = \ \prod_{i,j \in T, j \neq i} \frac{-j}{i-j}. \tag{5}$$

Then, each actual proxy signer sends

$$\{[s_i]^{d_i} \ mod \ N_i, s_i\}$$

to the combiner.

2. The combiner verifies the s_i using the public key of the proxy signer P_i, $i \in T$, and collects $[s_i]^{d_i} \ mod \ N_i$. And, the proxy signature S on message M can be generated using the following equation from the Lagrange formula:

$$\begin{aligned} S &= \prod_{i \in T} s_i \ \bmod \ N_0 \\ &= \prod_{i \in T} (M^{L_i \times k_i}) \ \bmod \ N_0 \\ &= M^{\sum_{i \in T}(L_i \times f(i))} \ \bmod \ N_0 \\ &= M^{f(0)} \ \bmod \ N_0 \\ &= M^D \ \bmod \ N_0. \end{aligned} \tag{6}$$

In this phase, the proxy signers sign the message without revealing their secret partial proxy signing keys.

3.3 The Verification Phase

The parameters N_i, e_i, m_w, and E are publicly known.

1. First, any receiver computes m_w and E with the original signer's public key. Then, the receiver checks the validity of the stipulated period. If the period has expired, the proxy verification key is invalid.
2. The receiver can then verify that the message M was signed on behalf of the original signer with the proxy verification key E as shown in the following equation:

$$\begin{aligned} S^E \ \bmod \ N_0 &= (M^D)^E \ \bmod \ N_0 \\ &= M^{d_0^{m_w} e_0^{m_w}} \ \bmod \ N_0 \\ &= M^{(d_0 \times e_0)^{m_w}} \ \bmod \ N_0 \\ &= M, \end{aligned} \tag{7}$$

where $d_0 \times e_0 = 1 \ \bmod \ \phi(N_0)$.

3. For internal auditing purposes, the original signer can differentiate the actual signers from the signatures $[s_i]^{d_i} \ mod \ N_i$ on message s_i, where $i \in t$.

4 Discussions

In this section, we examine the security of this scheme and compare our scheme with Kim et al.'s scheme.

4.1 Property Analysis

4.1.1 Secrecy

In our scheme, both signing and encrypting are based on the RSA cryptosystem. In the RSA scheme, any party's private key d_i and $\phi(N_i)$ must be kept secret. From the public key, no one can derive the corresponding private key. The security of RSA is based on the difficulty of factoring the RSA modulus N. Furthermore, we also cannot compute the original signer's private key from the group proxy signature key $D = d_0^{m_w} \ mod \ \phi(N_0)$ since the parameters d_0 and $\phi(N_0)$ are unknown. Even if t out of n proxy signers collaborate to deliver the group proxy signature key, without $\phi(N_0)$ they cannot calculate the original signer's private key d_0. Therefore, in our scheme, the original signer's private key can always be kept secret and used repeatedly.

In the Lagrange interpolating polynomial, the polynomial of degree $t-1$ requires t or more proxy signers to reconstruct the secret polynomial f. This enforces that we need any t or more proxy signers to reconstruct the proxy signature key $D = f(0)$, but $t-1$ or less proxy signers cannot. Furthermore, when the stipulated period has expired, the proxy signature key becomes invalid. Therefore, our scheme meets the property of secrecy.

4.1.2 Proxy Protected

In our scheme, the partial proxy signing key k_i is calculated by the original signer. Hence, the original signer can compute the partial proxy signature s_i from (4). However, the original signer cannot generate a valid signature for s_i on behalf of P_i since the original signer does not know P_i's private key d_i. The combiner does not accept the partial proxy signature s_i when the received signature of s_i is incorrect. Therefore, the original signer has no ability to substitute for proxy signers. The property of proxy protection is fulfilled in our scheme.

4.1.3 Unforgability

Each member only knows a secret share $f(i)$ of the polynomial f. Therefore, only the designated group of n proxy signers can sign messages. Without the proxy signature key k_i, no one can forge the proxy signer P_i to create s_i and pass the verification.

Furthermore, as in Lagrange interpolation, the polynomial of degree $t-1$ such that $t-1$ or less proxy signers cannot reconstruct the polynomial to get $f(0) = D$. Without

D, no one can forge a valid proxy signature on behalf of the original signer. We also cannot deliver the partial proxy signing key k_i from s_i in (4). The security is the result of the difficulty in calculating discrete logarithms. Therefore, our scheme ensures that a valid proxy signature can be generated only when t or more proxy signers cooperatively sign the message.

4.1.4 Nonrepudiation

In the property of nonrepudiation, both the original signer and the actual proxy signers cannot deny having signed the proxy signature. From the signature $[s_i]^{d_i}\ mod\ N_i$, P_i cannot deny having signed the partial proxy signature because only P_i has the private key d_i. Therefore, any verifier can identify the actual signers. Furthermore, the proxy verification key E is signed with the original signer's private key. The original signer cannot deny having delegated the power of signing to the proxy signers. The proxy signature is equivalent to the signature of the original signer.

4.1.5 Time Constraint

In this scheme, the warrant records the stipulated period of this proxy. In the verification phase, the verifier checks whether or not the warrant has expired. The information in the warrant m_w can be trusted because it is signed by the original signer. If the period has expired, the proxy verification key E becomes invalid. If a proxy signature was generated using the overdue proxy signature keys, the verification of the proxy signature would fail. This is very practical because, when the original signer returns from his business travel, the proxy signing key becomes invalid and the capability of signing messages on behalf of the original signer disappears.

4.1.6 Known Signer

For auditing purposes, the combiner needs to know who actually signs a message. This mechanism improves the responsibility of each proxy signer. In our scheme, each proxy signer sends the partial proxy signature to the combiner, who must append the proxy signer's signature. The combiner can verify the signature using the signer's public key e_i. Therefore, we can identify the actual signer.

4.2 Comparisons

In research relating to threshold proxy signatures, this concept was first proposed by Kim et al. [13]. They defined the requirements of the threshold proxy signature and presented a secure method which satisfied the requirements. However, in Kim et al.'s scheme, the actual signers are unknown. This is very inconvenient for internal auditing. Sun improved upon Kim et al.'s scheme so that actual signers can be identified [23]; however, there is a security weakness [12]. For this reason, we only compare our scheme with Kim et al.'s scheme [13]. Kim et al.'s scheme consists of the random number generation phase, the proxy sharing phase, and the proxy signature issuing and verification phase. To clarify the comparison, the processes are briefly described as follows.

4.2.1 Kim's Scheme

Step 1. The random number generation phase.

Kim et al.'s (t, n) threshold proxy signature scheme requires a protocol [20], [21] to generate a random number among the group without the dealer. Let P_0 be the original signer and $P_1, P_2, \cdots, P_n$ be the n proxy signers of the proxy group.

1. Each proxy signer P_i selects a secret polynomial of degree $t-1$ such that
$$f_i(x) = r_i + a_{i,1}x + a_{i,2}x^2 + \cdots + a_{i,t-1}x^{t-1} \mod q, \tag{8}$$
where $r_i, a_{i,1}, a_{i,2}, \cdots, a_{i,n}$ are random numbers. If we ignore the overhead for creating a random number, it requires a total of n polynomial generations for n proxy signers.
2. Then, each P_i computes $f_i(j) \mod q$ and sends it to P_j for all $1 \le j \le n$ and $j \ne n$. Furthermore, P_i computes
$$g^{r_i},\ g^{a_{i,1}},\ g^{a_{i,2}} \cdots, g^{a_{i,t-1}} \pmod p \tag{9}$$
and broadcasts them. Since each proxy signer has to calculate and transmit $f_i(j)$ to other $(n-1)$ proxy signers, it requires $n \times (n-1)$ polynomial computations and transmissions. Also, it requires $n \times t$ modular exponential computations and $n \times t$ broadcasts for (9).
3. After receiving $f_j(i)\ (for\ j = 1, \cdots, n,\ and\ j \ne i)$, P_i confirms the validity of $f_j(i)$ by checking whether or not $g^{f_j(i)}$ satisfies the following equation:
$$g^{f_j(i)} = g^{r_j} \times \left(g^{a_{j,1}}\right)^{i^1} \times \cdots \times \left(g^{a_{j,t-1}}\right)^{i^{t-1}} \mod p. \tag{10}$$
Each proxy signer performs the verification $(n-1)$ times and each verification requires t modular exponential computations and $(t-1)$ modular multiplications. Thus, this step requires $n \times (n-1) \times t$ modular exponential computations and $n \times (n-1) \times (t-1)$ modular multiplications.
4. If the verifications in Step 3 hold, each P_i computes the secret share
$$s_i = \sum_{j=1}^{j=n} f_j(i) \tag{11}$$
and computes the public outputs
$$\begin{aligned} r &= \prod_{j=1}^{j=n} r_j \mod p, \\ g^{a_1} &= \prod_{j=1}^{j=n} g^{a_{j,1}} \mod p, \\ g^{a_2} &= \prod_{j=1}^{j=n} g^{a_{j,2}} \mod p, \\ &\vdots \\ g^{a_{t-1}} &= \prod_{j=1}^{j=n} g^{a_{j,t-1}} \mod p. \end{aligned} \tag{12}$$

It is assumed that additional computational overhead can be ignored. To calculate both equations, $n \times t \times (t-1)$ modular multiplications are required.

In total, the following computations are required:

- n polynomial generations,
- $n \times (n-1)$ polynomial computations,
- $n^2 \times t$ modular exponential computations, and
- $n^2 \times (t-1) + n \times (t-1)^2$ modular multiplications.

Furthermore, the following communications are required:

- $n \times (n-1)$ transmissions and
- $n \times t$ broadcasts.

Step 2. The proxy sharing phase.

1. (Group key generation). First, the proxy group must execute the above protocol to obtain the share s_i and the public outputs $y_G = g^{a_0} \mod p$, $A_j = g^{a_j} (\mod p)$, where $j = 1, 2, \cdots, t-1$.
2. (Proxy generation). The original signer computes $K = g^k \mod p$ and $e = h(m_w, K)$, where k is a random number, m_w is a warrant, and $h()$ is a one-way hash function. After this, P_0 computes $\sigma = e \times x_0 + k \mod q$, where x_0 is a private key of the original signer. Therefore, the original signer requires one modular exponential computation, one modular multiplication, and one hash function computation.
3. (Proxy sharing). P_0 randomly chooses a polynomial such that

$$f'(x) = \sigma + b_1 x + b_2 x^2 + \cdots + b_{t-1} x^{t-1},$$

where $b_1, b_2, \cdots, b_{t-1}$ are random numbers. Then, P_0 computes $f'(i)$ and sends it to each P_i in a secret manner. P_0 also computes

$$B_1 = g^{b_1}, B_2 = g^{b_2}, \cdots, B_{t-1} = g^{b_{t-1}} \ (\mod p)$$

and publishes $m_w, K, B_1, B_2, \cdots, B_{t-1} \ (\mod p)$. Therefore, this step requires one polynomial generation, n polynomial computations, and $t-1$ modular exponential computations. Also, P_0 needs to broadcast $t+1$ parameters and deliver n parameters to each proxy signer.
4. (Proxy-Share generation). After receiving $f'(i)$, each P_i has to validate $f'(i)$ using the following equation:

$$g^{f'(i)} = y_0^{h(m_w, K)} K \prod_{j=1}^{t-1} B_j^{ij} \ (\mod p), \tag{13}$$

where y_0 is the original signer's public key. If it holds, each proxy signer P_i computes the proxy sharing $\sigma_i' = f'(i) + s_i \times e \mod q$. For all proxy signers, $n \times (t+1)$ modular exponential computations, $n \times (t+1)$ modular multiplications, and n hash function computations are required.

In total, the following computations are required:

- $n+1$ polynomial generations,
- n^2 polynomial computations,
- $n^2 \times t + n \times (t+1) + t$ modular exponential computations,
- $n^2 \times (t-1) + n \times (t^2 - t + 2) + 1$ modular multiplications, and
- $n+1$ hash function computations.

Also, the following communications are required:

- n^2 transmissions and
- $n \times t + t + 1$ broadcasts.

Step 3. The proxy signature issuing and verification phases.

1. The t or more actual signers have to execute the random number generation phase to obtain the secret output s_i' and public outputs $y = g^{c_0}$,

$$C_1 = g^{c_1}, C_2 = g^{c_2}, \cdots, C_{t-1} = g^{c_{t-1}} \ (\mod p),$$

where

$$s_i' = f''(i) = c_0 + c_1 i + c_2 i^2 + \cdots + c_{t-1} i^{t-1}.$$

The procedure requires t polynomial generations, $(t^2 - t)$ polynomial computations, t^3 modular exponential computations, and $2t^3 - 3t^2 + t$ modular multiplications. It also requires $t^2 - t$ parameter transmissions and t^2 parameter broadcasts.
2. Then, each actual signer uses his proxy signature key to issue a partial proxy signature such that $e' = h(y, m)$ and $\gamma_i = s_i' + \sigma_i' \times e' (\mod q)$, where m is the message. Then, each actual signer reveals γ_i. This step requires t modular multiplications, t hash function computations, and t transmissions.
3. Everyone can verify the validity of γ_j by the following equation:

$$g^{\gamma_j} = \left(y \prod_{i=1}^{t-1} C_i^{j^i} \times \left[(y_0^{h(m_w, K)} K \prod_{i=1}^{t-1} B_i^{j^i} \right) \times \left(y_G \prod_{i=1}^{t-1} A_i^{j^i} \right)^{h(m_w, K)} \right]^{h(y,m)} \mod p. \tag{14}$$

The verification requires $3t^2 + t$ modular exponential computations, $3t^2$ modular multiplications, and $2t$ hash function computations.
4. If the previous verification holds, the signature on m is (m, T, e', K, m_w), where $T = c_0 + \sigma \times e' = f''(0) + f(0) \times e'$ can be computed by applying the Lagrange formula.
5. To verify the validity of the signature, anyone can examine the following equation:

$$y' = g^T \times (y_0^{h(m_w, K)} K)^{-e'} \mod p, \ and \tag{15}$$

$$e' = h(y', m). \tag{16}$$

The verification requires three modular exponential computations, two modular multiplications, and two hash function computations.

The total computational and communicational overheads of Kim et al.'s scheme are summarized in Table 1 and Table 2, respectively.

TABLE 1
A Comparison of Computational Overheads of the Proposed Scheme and Kim et al.'s Scheme

Schemes		# of polynomial generations	# of polynomial computations	# of modular exponential computations	# of modular multiplications	# of hash calculations	Total
Kim's scheme	Proxy sharing	n+1	n^2	$n^2t+n(t+1)+t$	$n^2(t-1)+n(t^2-t+2)+1$	n+1	$\mathbf{n^2(2t+2)+n(t^2+5)+3}$ $\mathbf{t^3+4t^2+7t+10}$
	Proxy signature issuing and verification	t	t^2-t	t^3+3t^2+t+3	$2t^3+2t+2$	3t+2	
Proposed scheme	Proxy sharing	1	n	4n+3	0	0	$\mathbf{5n+t^2+4t+5}$
	Proxy signature issuing and verification	0	0	3t+2	t^2+t-1	0	

4.2.2 Our Proposed Scheme

In our proposed scheme, a mechanism to share a number among the group is not required. The computational and communicational overheads of the proxy sharing phase are described as follows:

1. In proxy generation, the original signer generates the group signature key D and its corresponding proxy verification key E by using (1) and (2). Then, P_0 publishes $\{m_w, E, [m_w \parallel E]^{d_0}\}$. These tasks require three modular exponential computations and three broadcasts.
2. In proxy sharing, the original signer computes the shares of all proxy signers. This requires one polynomial generation, n polynomial computations, $2n$ modular exponential computations, and n transmissions of keys.
3. In proxy share generation, each proxy signer can obtain and validate the proxy share. It requires $2n$ modular exponential computations.

In total, the following computations are required:

- one polynomial generation,
- n polynomial computations, and
- $4n + 3$ modular exponential computations.

TABLE 2
A Comparison of Communicational Overheads between the Proposed Scheme and Kim et al.'s Scheme

Schemes		# of transmissions	# of broadcasts	Total
Kim's scheme	Proxy sharing	n^2	nt+t+1	$\mathbf{n^2+nt+3t^2-t+1}$
	Proxy signature issuing and verification	$2(t^2-t)$	t^2	
Proposed scheme	Proxy sharing	n	3	**n+t+3**
	Proxy signature issuing and verification	t	0	

Furthermore, the following communications are required:

- n transmissions and
- three broadcasts.

In the proxy signature issuing and verification phases, the required computations and communications are described as follows:

1. Each actual signer executes (4) and (5) to issue the partial proxy signature and sends $[s_i]^{d_i}$ to the combiner. This step requires $2t$ modular exponential computations, t^2 modular multiplications, and t transmissions.
2. Then, the combiner verifies the digital signature of all s_i which requires t modular exponential computations. The generation of S on M requires $t-1$ modular multiplications.
3. To verify the proxy signature, the verifier first checks the validity of the warrant which requires one modular exponential computation, and then verifies whether or not the message is signed on behalf of the original signer as shown in (7). This requires one modular exponential computation.

The total computations in the proxy signature issuing and verification phases are listed below:

- $3t + 2$ modular exponential computations and
- $t^2 + t - 1$ modular multiplications.

And, the communicational is

- t transmissions.

The overall computational and communicational overheads of both schemes are summarized in Table 1 and Table 2, respectively. For comparison purposes, we consider a $(5, 10)$ threshold proxy signature scheme and assume that the modulus of the two schemes are all 512 bits, the overhead of each computation and communication is one unit, and the problem of overflow is ignored. For $n = 10$ and $t = 5$, the computations are 100 and $2,020$ for our scheme and Kim et al.'s scheme, respectively. The communicational overheads are 221 and 18 for our scheme and Kim et al.'s scheme, respectively. Note that, although Kim et al.'s scheme does not identify the signer's identity and ours does, our scheme is still far superior. The overheads of our

scheme constitute only 5 percent of the computations and 8 percent of the communications of Kim et al.'s scheme.

5 Applications

In this section, we outline some applications of (t, n) threshold proxy signature in the areas of mobile agent and electronic contract.

5.1 Mobile Agent

The mobile agent offers a new computing paradigm and becomes increasingly important for the applications of electronic commerce, virtual enterprises, and distributed database systems. The technology allows an agent in the form of software programs, data, or execution state that acts on behalf of its owner to perform one or more tasks autonomously and return the results to the agent owner. Furthermore, the mobile agent is developed for the use in open, distributed, and heterogeneous environments, thus it can migrate across the Internet and resume execution on a remote computer. With the features of mobility and autonomy, the mobile agent has been widely used in many applications such as electronic purses, electronic payment, information retrieval, and negotiation systems to achieve higher efficiency at lower costs [4], [19]. In a word, the mobile agent has such advantages as conserving network bandwidth, reducing network latency, asynchrony, and dynamic adaptation. However, the security threats are still the deployment bottleneck. For instances, the private keys and the confidential information carried by the agent are very difficult to protect in a hostile execution environment. If a mobile agent can sign a message on behalf of the agent owner (customer) in a remote server (Vendor) without revealing customer's private key, the mobile agent can be used not only to search for special products or services, but also to sign a contract with the vendor [14]. Fortunately, the (t, n) threshold proxy signature technology provides a good solution to this problem. It can be used to construct a secure mobile agent. The customer can delegate the power of signing messages to a designated agent. Consequently, the agent can issue a proxy signature on behalf of the customer without revealing the customer's private key. For example, a flight booking agent can work on behalf of a customer to search for a lower price ticket and book it by digital signature. The functions of the agent can include browsing products, gathering shopping information, and analyzing collected information from the vendors. Besides, our technology can also be applied in multiagent systems [14]. A customer can delegate signing power to n designated agents. The agents can have different goals and try to make a good decision without concerning the global decisions. In this case, when t or more agents come to the same decision, then the transaction is valid. Multiagent technology can futhermore reduce the burden of human's effort and can be great help for decision makers.

5.2 Electronic Contract

The concept of (t, n) threshold proxy signature also can be used when there are electronic contracts to sign [1]. In electronic contract systems, the handwritten signatures have to be replaced with digital signatures. However, in traditional digital signature schemes, a message can be signed only by a legal signer with her/his private key. If a customer directly delivers her/his private key to a broker to sign a contract with a vendor, then the private key is no longer secure. As for our approach, it fits the usage scenarios of electronic contract. Our approach allows a customer to delegate her/his power to a broker to work on behalf of the customer and negotiate with vendors and, finally, make a contract with the victorious vendor. In the contract-making process, the customer's private key does not leak out. For making a contract discreetly, the customer can also choose to delegate the signing capability to n designated brokers so that only when t or more brokers sign the contract with the same vendor can the contract be valid.

6 Conclusions

In this article, we have proposed a practical, efficient, and secure (t, n) threshold proxy signature scheme based on the RSA cryptosystem. According to previous discussions, our scheme significantly reduces the cost of computation and communication. Our scheme does not require a protocol to generate a random number. It only takes the ordinary RSA digit signature scheme [22] and the Lagrange formula [2] for a signature receiver to obtain and verify the proxy signature. Therefore, our scheme not only meets all of the requirements of proxy signature, but it is also more efficient than existing threshold proxy signature schemes. In summary, our scheme possesses the following abilities:

- the ability to flexibly choose the threshold number t,
- the ability to repeatedly use the participant's RSA key pairs which can also be used in other work,
- the ability to put time constraints on the threshold delegation, and
- the ability to identify the actual signers.

Acknowledgments

The authors wish to thank many anonymous referees for their suggestions to improve this paper. This research was partially supported by the National Science Council, Taiwan, R.O.C., under contract no.: NSC90-2213-E-324-005.

References

[1] M. Ben-Or, O. Goldreich, S. Micali, and R.L. Rivest, "A Fair Protocol for Signing Contracts," *IEEE Trans. Information Theory*, vol. 36, no. 1, pp. 40-46, 1990.

[2] G.R. Blakley, "Safeguarding Cryptographic Keys," *Proc. Am. Federation of Information Processing Soc.*, pp. 313-317, 1979.

[3] D. Bonch and M. Franklin, "Efficient Generation of Shared RSA Keys," *Proc. Advance in Cryptology (Crypto '97)*, vol. 1233, pp. 425-439, 1997.

[4] P. Dasgupta, N. Narasimhan, L.E. Moser, and P.M. Melliar-Smith, "MagNET: Mobile Agents for Networked Electronic Trading," *IEEE Trans. Knowledge and Data Eng.*, vol. 11, no. 4, pp. 509-525, May/June 1999.

[5] D.E.R. Denning, *Cryptography and Data Security.* Addison-Wesley, 1983.

[6] Y. Desmedt and Y. Frankel, "Threshold Cryptosystems," *Proc. Advance in Cryptology (Crypto '89)*, pp. 307-315, 1989.

[7] Y. Desmedt and Y. Frankel, "Shared Generation of Authenticators and Signatures," *Proc. Advance in Cryptology (Crypto '91)*, pp. 457-469, 1991.

[8] T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. Information Theory*, vol. 31, no. 4, pp. 469-472, 1985.
[9] Y. Frankel, "A Practical Protocol for Large Group Oriented Networks," *Proc. Advance in Cryptology (Eurocrypt '89)*, pp. 56-61, 1989.
[10] M.S. Hwang, C.C. Chang, and K.F. Hwang, "An Efficient Threshold Decryption Scheme without Session Keys," *Computers & Electrical Eng.*, vol. 27, no. 1, pp. 29-35, 2000.
[11] M.S. Hwang, C.C. Chang, and K.F. Hwang, "An ElGamal-Like Cryptosystem for Enciphering Large Messages," *IEEE Trans. Knowledge and Data Eng.*, vol. 14, no. 2, pp. 445-446, 2002.
[12] M.S. Hwang, I.C. Lin, and E.J.L. Lu, "A Secure Nonrepudiable Threshold Proxy Signature Scheme with Known Signers," *Int'l J. Informatica*, vol. 11, no. 2, pp. 1-8, 2000.
[13] S. Kim, S. Park, and D. Won, "Proxy Signatures, Revisited," *Proc. Int'l Conf. Information Security and Comm. Security (ICICS '97)*, pp. 223-232, 1997.
[14] B. Lee, H. Kim, and K. Kim, "Secure Mobile Agent Using Strong Non-Designated Proxy Signature," *Lecture Notes in Computer Science-Information Security and Privacy*, vol. 2119, Springer Verlag, pp. 474-486, 2001.
[15] N.Y. Lee, T. Hwang, and C.H. Wang, "On Zhang's Nonrepudiable Proxy Signature Schemes," *Proc. Australasian Conf. Information Security and Privacy (ACISP '98)*, pp. 415-422, 1998.
[16] M. Mambo, K. Usuda, and E. Okamoto, "Proxy Signatures: Delegation of the Power to Sign Message," *IEICE Trans. Fundamentals*, vol. E79-A, no. 9, pp. 1338-1354, 1996.
[17] M. Mambo, K. Usuda, and E. Okamoto, "Proxy Signatures for Delegating Signing Operation," *Proc. Third ACM Conf. Computer and Comm. Security*, pp. 48-57, 1996.
[18] B.C. Neuman, "Proxy-Based Authorization and Accounting for Distributed Systems," *Proc. 13th Int'l Conf. Disctributed Systems*, pp. 283-291, 1993.
[19] S. Papastavrou, G. Samaras, and E. Pitoura, "Mobile Agent for World Wide Web Distributed Database Access," *IEEE Trans. Knowledge and Data Eng.*, vol. 12, no. 5, pp. 802-820, 2000.
[20] T. Pedersen, "Distributed Provers with Applications to Undeniable Signatures," *Proc. Advance in Cryptology (Eurocrypt '91)*, pp. 221-238, 1991.
[21] T.P. Pedersen, "A Threshold Cryptosystem without a Trusted Party," *Proc. Eurocrypt '91*, pp. 522-526, 1991.
[22] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, vol. 21, pp. 120-126, Feb. 1978.
[23] H.-M. Sun, "An Efficient Nonrepudiable Threshold Proxy Signature Scheme with Known Signers," *Computer Comm.*, vol. 22, no. 8, pp. 717-722, 1999.
[24] K. Usuda, M. Mambo, T. Uyematsu, and E. Okamoto, "Proposal of an Automatic Signature Scheme Using a Compiler," *IEICE Trans. Fundamentals*, vol. E79-A, no. 1, pp. 94-101, 1996.
[25] V. Varabharajan, P. Allen, and S. Black, "An Analysis of the Proxy Problem in Distributed Systems," *Proc. 1991 IEEE CS Symp. Research in Security and Privacy*, pp. 225-275, 1991.
[26] K. Zhang, "Threshold Proxy Signature Schemes," *Proc. 1997 Information Security Workshop*, pp. 191-197, 1997.

Min-Shiang Hwang received the BS degree in electronic engineering from National Taipei Institute of Technology, Taipei, Taiwan, Republic of China (R.O.C), in 1980, the MS degree in industrial engineering from National Tsing Hua University, Taiwan, in 1988, and the PhD degree in computer and information science from National Chiao Tung University, Taiwan, in 1995. He also studied applied mathematics at National Cheng Kung University, Taiwan, from 1984-1986. Dr. Hwang passed the National Higher Examination in field "Electronic Engineer" in 1988. He also passed the National Telecommunication Special Examination in the field of "Information Engineering," and qualified as an advanced technician, first class in 1990. From 1988 to 1991, he was the leader of the Computer Center at Telecommunication Laboratories (TL), Ministry of Transportation and Communications, R.O.C. He was also a project leader for research in computer security at TL in July 1990. He received the 1997, 1998, and 1999 Distinguished Research Awards from the National Science Council of the Republic of China. He is currently a professor and chairman of the Department of Information Management, Chaoyang University of Technology, Taiwan, ROC. He is a member of IEEE, ACM, and Chinese Information Security Association. His current research interests include database and data security, cryptography, image compression, and mobile communications.

Eric Jui-Lin Lu received the BS degree in transportation engineering and management from National Chiao Tung University, Taiwan, R.O.C, in 1982, the MS degree in computer information systems from San Francisco State University, California, in 1990, and PhD degree in computer science from the University of Missouri-Rolla, Missouri, in 1996. He is currently an associate professor in the Department of Information Management and director of Computer Center, Chaoyang University of Technology, Taiwan, R.O.C. His current research interests include electronic commerce, distributed processing, and security.

Iuon-Chang Lin received the BS degree in computer and information sciences from Tung Hai University, Taichung, Taiwan, Republic of China, in 1998, and the MS degree in information management from Chaoyang University of Technology, Taiwan, in 2000. He is currently pursuing his PhD degree in computer science and information engineering from National Chung Cheng University. His current research interests include electronic commerce, information security, cryptography, and mobile communications.