An ECDLP-Based Threshold Proxy Signature Scheme Using Self-Certified Public Key System*

Qingshui Xue[1], Fengying Li[1, 2], Yuan Zhou[3], Jiping Zhang[3], Zhenfu Cao[4], and Haifeng Qian[5]

[1] School of Techniques, Shanghai Jiao Tong University, 201101, Shanghai, China
xue-qsh@sjtu.edu.cn
[2] Dept. of Education Information Technology, East China Normal University, 200062, Shanghai, China
fyli@sjtu.edu.cn, jpzhang@deit.ecnu.edu.cn
[3] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing, 100029, China
zhouyuantdt@163.com
[4] Dept. of Computer Science and Engineering, Shanghai Jiao Tong University, 200240, Shanghai, China
zfcao@cs.sjtu.edu.cn
[5] Dept. of Computer Science and Technology, East China Normal University, 200062, Shanghai, China
hfqian@cs.ecnu.edu.cn

Abstract. In a (t, n) threshold proxy signature scheme, one original signer delegates a group of n proxy signers to sign messages on behalf of the original signer. When the proxy signature is created, at least t proxy signers cooperate to generate valid proxy signatures and any less than t proxy signers can't cooperatively generate valid proxy signatures. So far, all of proposed threshold proxy signature schemes are based on public key systems with certificates, which have some disadvantages such as checking the certificate list when needing certificates. Most threshold proxy signature schemes use Shamir's threshold secret share scheme. Identity-based public key system is not pretty mature. Self-certified public key systems have attracted more and more attention because of its advantages. Based on Hsu et al's self-certified public key system and Li et al's proxy signature scheme, one threshold proxy signature scheme based on ECDLP and self-certified public key system is proposed. As far as we know, it is the first scheme based on ECDLP and self-certified public key system. The proposed scheme can provide the security properties of proxy protection, verifiability, strong identifiability, strong unforgeability, strong repudiability, distinguishability, known signers and prevention of misuse of proxy signing power. That is, internal attacks, external attacks, collusion attacks, equation attacks and public key substitution attacks can be resisted. In the proxy signature verification phase, the authentication of the original and the proxy signers' public keys and the verification of the threshold proxy signature are executed together. In addition, the computation overhead and communication cost of the proposed scheme are analyzed as well.

* This paper is supported by the National Natural Science Foundation of China under Grant No. 60673079 and 60873217, and National Basic Research Program of China (973 Program) under Grant No.2007CB311100.

A.U. Schmidt and S. Lian (Eds.): MobiSec 2009, LNICST 17, pp. 58 – 70, 2009.

1 Introduction

The proxy signature scheme [1], a variation of ordinary digital signature schemes, enables a proxy signer to sign messages on behalf of the original signer. Proxy signature schemes are very useful in many applications such as electronics transaction and mobile agent environment.

Mambo et al. [1] provided three levels of delegation in proxy signature: full delegation, partial delegation and delegation by warrant. In full delegation, the original signer gives its private key to the proxy signer. In partial delegation, the original signer produces a proxy signature key from its private key and gives it to the proxy signer. The proxy signer uses the proxy key to sign. As far as delegation by warrant is concerned, warrant is a certificate composed of a message part and a public signature key. The proxy signer gets the warrant from the original signer and uses the corresponding private key to sign. Since the conception of the proxy signature was brought forward, a lot of proxy signature schemes have been proposed [2]-[19].

Recently, many threshold proxy signature schemes were proposed [2] [6]-[14]. In threshold proxy signature schemes, a group of n proxy signers share the secret proxy signature key. To produce a valid proxy signature on the message m, individual proxy signers produce their partial signatures on that message, and then combine them into a full proxy signature on m. In a (t,n) threshold proxy signature scheme, the original signer authorizes a proxy group with n proxy members. Only the cooperation of t or more proxy members is allowed to generate proxy signatures. Threshold signatures are motivated both by the demand which arises in some organizations to have a group of employees agree on a given message or document before signing, and by the need to protect signature keys from attacks of internal and external adversaries.

In 1999, Sun proposed a threshold proxy signature scheme with known signers [9]. Then Hwang et al. [7] pointed out that Sun's scheme was insecure against collusion attack. By the collusion, any $t-1$ proxy signers among t proxy signers can cooperatively obtain the secret key of the remainder one. They also proposed an improved scheme which can guard against the collusion attack. After that, [6] showed that Sun's scheme was also insecure against the conspiracy attack. In the conspiracy attack, t malicious proxy signers can impersonate some other proxy signers to generate valid proxy signatures. To resist the attack, they also proposed a scheme. Hwang et al pointed out [8] that the scheme in [7] was also insecure against the attack by the cooperation of one malicious proxy signer and the original signer. In 2002, Li et al. [2] proposed a threshold proxy signature scheme with good properties and performance.

All of the proposed schemes are based on Shamir's secret share protocol and the public key systems using certificates, which have some disadvantages such as checking the certificate list when needing certificates, and high computation overheads and communication cost.

So far, there are three kinds of public key systems involving using certificates, identity-based and self-certified public keys. Currently, identity-based public systems are not mature, as makes it not used in the real life.

The self-certified public key system was first introduced by Girault in 1991 [20]. In self-certified public key systems, each user's public key is produced by the **CA** (Certification Authority), while the corresponding private key is only known to the user. The authenticity of public keys is implicitly verified without the certificate. That is, the verification of public keys can be performed with the subsequent cryptographic applications such as key exchange protocols and signature schemes in a single step. Compared with other two public systems, the system has the following advantages [18]: ①the storage space and the communication overheads can be reduced since the certificate is not needed; ②the computation overhead can be reduced as it doesn't require public key verification.

In public key cryptosystems, there are several kinds of cryptographic assumptions to be used. Currently, only the discrete logarithm problem and factorization problem are widely accepted. In addition, the elliptic curve cryptosystem (ECC) [21] is constructed by integer points over elliptic curves in finite fields. The advantage of ECC is that it can reach the same level of security constituted by DSA [22] or RSA [23] and provides better efficiency than both discrete logarithm and factorization systems.

There are four trust levels for the security of public key systems [12]. Hsu et al [18] pointed out that the self-certified public key systems might be the ideal choice for realizing cryptographic applications according to security and efficiency. Further, Hsu et al [18] proposed a kind of self-certified public key system. In 2004, Hwang et al [19] proposed a generation of proxy signature based on elliptic curves. In the paper, based on Hsu et al's self-certified public key system and Hwang's et al's proxy signature scheme, a threshold proxy signature scheme using self-certified public system is proposed by us. The main advantage of the proposed scheme is that the authenticity of the original and the proxy signers' public keys, and the verification of the proxy signature can be simultaneously executed in a single step. As far as we know, this threshold proxy signature scheme is the first one using self-certified public key system.

In the paper, we will organize the content as follows. In section 2, we will detail the proposed threshold proxy signature scheme, which is based on the self-certified public key system [18] and Hwang et al's proxy signature scheme [17]. The security of the proposed scheme will be analyzed and discussed in section 3. In section 4, we will analyze the computational overheads and communication cost of the proposed scheme. Finally, the conclusion is given.

2 The Proposed Scheme

In the scheme, a system authority (**SA**) whose tasks are to initialize the system, the original signer U_o, certification authority (**CA**) whose tasks are to generate the public key for each user, the proxy group of n proxy signers $G_P = \{U_{P_1}, U_{P_2}, ..., U_{P_n}\}$, one designated clerk **C** whose tasks are to collect and verify the individual proxy signatures generated by the proxy signers, and construct the final threshold proxy signature, and the signature verifier are needed.

Throughout the paper, q is a large prime and E is an elliptic curve over a finite $GF(q)$. G is a base point on E with order n. h is a secure one-way hash function. The parameters (q,G,n) and the function h are made public. Let ID_i be the identifier of the user U_i. Assume that x_{CA} and y_{CA} are the private and public keys of the **CA**, respectively, where $x_{CA} \in Z_q^*$ and

$$y_{CA} = x_{CA}G \tag{1}$$

m_w is a warrant which records the identities of the original signer and the proxy signers of the proxy group, parameters t and n, message type to sign, the valid delegation time, etc. *ASID* (Actual Signers' *ID*) denotes the identities of the actual proxy signers.

The proposed scheme consists of four phases: registration, proxy share generation, proxy signature issuing without revealing proxy shares and proxy signature verification. We will detail them as follows.

2.1 Registration

Step 1. Each user U_i selects an integer $1 \le t_i \le n-1$ at random, computes

$$v_i = h(t_i, ID_i)G \tag{2}$$

and sends (v_i, ID_i) to the **CA**.

Step 2. Upon receiving (v_i, ID_i) from U_i, the **CA** selects $1 \le z_i \le n-1$, calculates

$$y_i = v_i + z_iG \tag{3}$$

$$e_i = z_i + h((y_i)_x, ID_i)x_{CA} \bmod n \tag{4}$$

and returns (y_i, e_i) to U_i. Here $(\cdot)_x$ denotes the x-coordinate of point $(\cdot)$ on E.

Step 3. U_i computes

$$x_i = e_i + h(t_i, ID_i) \bmod n \tag{5}$$

and confirms its validity by checking that

$$y_{CA}h((y_i)_x, ID_i) + y_i = x_iG \tag{6}$$

If it holds, U_i accepts (x_i, y_i) as his private and public keys. Moreover, the **CA** publishes $U_i's$ public key y_i when the registration is complete. Note that the **CA** needn't issue extra certificate associated with y_i.

2.2 Proxy Share Generation

Step 1. The original signer chooses randomly an integer $1 \le k_o \le n$, computes

$$K_o = k_oG \tag{7}$$

$$\sigma_o = k_o(K_o)_x + x_o h(m_w,(K_o)_x) \bmod n \tag{8}$$

and sends (m_w, K_o, σ_o) to each of proxy signers.

Step 2. After receiving (m_w, K_o, σ_o), each of proxy signers confirms the validity of (m_w, K_o, σ_o) by

$$\sigma_o G = (K_o)_x K_o + h(m_w,(K_o)_x)[y_{CA}h((y_o)_x, ID_o) + y_o] \tag{9}$$

If it holds, each of proxy signers regards σ_o as its proxy share.

2.3 Proxy Signature Issuing without Revealing Proxy Shares

Without loss of generality, the proposed scheme allows any *t* or more proxy signers to represent the proxy group to sign a message *m* cooperatively on behalf of the original signer U_o.

Let $G_{P'} = \{U_{P_1}, U_{P_2}, ..., U_{Pt'}\}$ be the actual proxy signers for $t \le t' \le n$. $G_{P'}$ as a group performs the following steps to generate a threshold proxy signature.

Step 1. Each proxy signer $U_{P_i} \in G_{P'}$ chooses an integer $k_i \in Z_q^*$ at random, computes

$$K_i = k_i G \tag{10}$$

and sends it to the other $t'-1$ proxy signers in $G_{P'}$ and the designated clerk **C**.

Step 2. Upon receiving K_j $(j = 1,2,...,t'; j \ne i)$, each $U_{P_i} \in G_{P'}$ computes *K* and s_i as follows:

$$K = \sum_{j=1}^{t'} K_j \tag{11}$$

$$s_i = k_i(K)_x + (\sigma_o t'^{-1} + x_{P_i})h(m, ASID) (\bmod n) \tag{12}$$

Here, s_i is an individual proxy signature which is sent to **C**.

Step 3. For each received s_i $(i = 1,2,...,t')$, **C** checks whether the following congruence holds:

$$\begin{aligned} s_i G = K_i(K)_x + \{t'^{-1}[(K_o)_x K_o + h(m_w,(K_o)_x)\cdot(y_{CA}h((y_o)_x, ID_o) + y_o)] + \\ [y_{CA}h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\}h(m, ASID) \end{aligned} \tag{13}$$

If it holds, (K_i, s_i) is a valid individual proxy signature on *m*. If all the individual proxy signatures of *m* are valid, the clerk **C** computes

$$S = \sum_{i=1}^{t'} s_i \bmod n \tag{14}$$

Then, $(m_w, K_o, m, K, S, ASID)$ is the proxy signature on *m*.

2.4 Proxy Signature Verification

After receiving the proxy signature $(m_w, K_o, m, K, S, ASID)$ for *m*, any verifier can verify the validity of the threshold proxy signature by the following steps.

Step 1. According to m_w and $ASID$, the verifier can obtain the value of t and n, the public keys of the original signer and proxy signers from **CA** and knows the number t' of the actual proxy signers. Then the verifier checks whether $t' \geq t$, if it holds, he/she continues the following steps, or else, he/she will regard the threshold proxy signature $(m_w, K_o, m, K, S, ASID)$ invalid.

Step 2. The verifier confirms the validity of the proxy signature on m by checking

$$SG = K(K)_x + h(m, ASID)\{[(K_o)_x K_o + h(m_w, (K_o)_x) \cdot (y_{CA} h((y_o)_x, ID_o) + y_o)] + \sum_{i=1}^{t'} [y_{CA} h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\} \quad (15)$$

If it holds, the proxy signature $(m_w, K_o, m, K, S, ASID)$ is valid.

3 Correctness of the Proposed Scheme

In the section, we shall prove that the proposed scheme can work correctly by the following theorems.

Theorem 1. *In the registration phase, the user U_i can verify the validity of its private and public key pair (x_i, y_i) by Equation (6).*

Proof. From Equations (4) and (5), we have $x_i = z_i + h((y_i)_x, ID_i) x_{CA} + h(t_i, ID_i) \bmod n$. By raising both sides of the above equation by multiplying them by the base point G, we have

$$x_i G = z_i G + h((y_i)_x, ID_i) x_{CA} G + h(t_i, ID_i) G .$$

From Equations (2) and (3), we have $z_i G = y_i - h(t_i, ID_i) G$. Then, we have

$$x_i G = y_i - h(t_i, ID_i) G + h((y_i)_x, ID_i) x_{CA} G + h(t_i, ID_i) G = y_i + h((y_i)_x, ID_i) x_{CA} G$$

From Equation (1), the above equation can be rewritten as

$$y_{CA} h((y_i)_x, ID_i) + y_i = x_i G$$

Theorem 2. *If the proxy share is constructed correctly, it will pass the verification of Equation (9).*

Proof. By raising both sides of Equation (8) by multiplying them by the base point G, we have

$$\sigma_o G = k_o (K_o)_x G + x_o h(m_w, (K_o)_x) G$$

According to Equations (6) and (7), the above equation can be rewritten as

$$\sigma_o G = (K_o)_x K_o + h(m_w, (K_o)_x)[y_{CA} h((y_o)_x, ID_o) + y_o]$$

Theorem 3. *In the proxy signature generation phase, the clerk **C** can verify any individual proxy signature s_i sent from U_{P_i} by Equation (13).*

Proof. By raising both sides of Equation (12) by multiplying them by the base point G, we have

$$s_i G = k_i (K)_x G + (\sigma_o t'^{-1} G + x_{P_i} G) h(m, ASID)$$

According to Equations (6), (9) and (10), the above equation can be rewritten as

$$s_i G = K_i (K)_x + \{t'^{-1} [(K_o)_x K_o + h(m_w, (K_o)_x) \cdot (y_{CA} h((y_o)_x, ID_o) + y_o)] + [y_{CA} h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\} h(m, ASID)$$

Theorem 4. *If the threshold proxy signature is constructed correctly, it will pass the verification of Equation (15).*

Proof. From Equations (12) and (14), we have

$$S = \sum_{i=1}^{t'} [k_i (K)_x + (\sigma_o t'^{-1} + x_{P_i}) h(m, ASID)] (\mathrm{mod}\, n)$$
$$= \sum_{i=1}^{t'} k_i (K)_x + \sigma_o h(m, ASID) + \sum_{i=1}^{t'} x_{P_i} h(m, ASID) (\mathrm{mod}\, n)$$

By raising both sides of the above equation by multiplying them by the base point G, we have

$$SG = \sum_{i=1}^{t'} k_i G(K)_x + \sigma_o G h(m, ASID) + \sum_{i=1}^{t'} x_{P_i} G h(m, ASID)$$

According to Equations (6), (9), (10) and (11), the above equation can be rewritten as

$$SG = K(K)_x + h(m, ASID)\{[(K_o)_x K_o + h(m_w, (K_o)_x) \cdot (y_{CA} h((y_o)_x, ID_o) + y_o)] + \sum_{i=1}^{t'} [y_{CA} h((y_{Pi})_x, ID_{Pi}) + y_{Pi}]\}$$

4 Security Analysis

In the section, we will propose several theorems about the security below and prove that they are right.

Theorem 5. *The user can't forge his/her private key without interaction with the* ***CA*** *and the* ***CA*** *can forge the user's public key without the interaction with the user neither.*

Proof. From Equation (4), we know that although the user can select a random integer $z_i \in Z_q^*$ and compute $y_i = v_i + z_i G$, because having no the knowledge of the **CA**'s private key x_{CA}, he/she can't get a valid value of e_i to construct his self-certified private key. Obviously, the user can forge a valid private key with the probability of $1/n$. That's, the user's private key has to be set up by the interaction with the **CA**.

Similarly, if the **CA** wants to forge the user's new public key which satisfies Equation (6), he/she has to solve the difficult discrete logarithm problem and the secure hash function, as we know it is impossible. Thus the **CA** can't forge a new public key of the user. To generate the user's public key, the **CA** has to interact with the user.

Theorem 6. *The user can't forge his public key by its private key without the interaction with the* ***CA*** *and the* ***CA*** *can't get the user's private key from the interaction with the user either.*

Proof. If the user wants to forge his/her new public key which satisfies Equation (6), he/she has to solve the difficult discrete logarithm problem and secure hash functions, as we know it is impossible. Thus the user can't forge a new public key of the user without the interaction with the **CA**. From Equation (5), we know that because the **CA** has no the knowledge of $t_i \in Z_q^*$ selected by the user, the **CA** can't obtain the user's private key x_i. In addition, from the verification equation (6), the **CA** is unable to get the user's private key x_i since he/she is faced with the difficulty of solving discrete logarithms and secure hash functions. Therefore, we can draw the above conclusion.

Theorem 7. *Any* $t'' < t$ *proxy signers can't generate a valid threshold proxy signature on a new message* m'.

Proof. From Equations (12) and (14), we have

$$S = \sum_{i=1}^{t'} k_i (K)_x + \sigma_o h(m, ASID) + \sum_{i=1}^{t'} x_{P_i} h(m, ASID) (\mathrm{mod}\, n) \tag{16}$$

Because any $t'' < t$ proxy signers have no the knowledge of k_i or $\sum k_i$, and x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers, any $t'' < t$ proxy signers are unable to cooperate to generate the valid proxy signature on a new message m'. Although any $t'' < t$ proxy signers can generate $(m_w, K_o, m', K, S, ASID)$ and it can pass the verification Equation (15), the number of actual proxy signers is less than *t*, as makes it can't pass the verification step 1. Thus the forged proxy signature $(m_w, K_o, m', K, S, ASID)$ by $t'' < t$ proxy signers is invalid.

From the Equation (16), any $t'' < t$ proxy signers are unable to get the values of k_i or $\sum k_i$, and x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers, if the message m is replaced with m', any $t'' < t$ proxy signers can't get the new value of S'. Also, from the verification Equation (15), when m is replaced with m', given fixed some variables of the set $\{m_w, K_o, K, S, ASID\}$, the values of the other variables in the set $\{m_w, K_o, K, S, ASID\}$ will be unable to be gotten because of the difficult discrete logarithm and secure hash function. That is, from a known proxy signature $(m_w, K_o, m, K, S, ASID)$, any $t'' < t$ proxy signers can't generate valid threshold proxy signature on a new message m'. So the theorem is proved to be true.

Theorem 8. *Any $t'' < t$ proxy signers can't forge another valid threshold proxy signature on the original message m from the proxy signature $(m_w, K_o, m, K, S, ASID)$.*

Proof. On one hand, from Equation (16), any $t'' < t$ proxy signers can't get the knowledge of k_i or $\sum k_i$, and x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers. Thus, the values of some variables in set $\{K, S, ASID\}$ can't be changed by changing the values of the other variables in set $\{K, S, ASID\}$. Here note that as far as any $t'' < t$ proxy signers are concerned, the values of m_w and K_O can't be changed, as can be guaranteed by Equation (9). On the other hand, from the verification Equation (15), by fixing some variables of the set $\{m_w, K_o, K, S, ASID\}$, the values of the other variables in the set $\{m_w, K_o, K, S, ASID\}$ will not be able to be gotten because of the difficult discrete logarithm and secure hash functions. Therefore, the theorem is proved true.

Theorem 9. *The original signer and any $t'' < t$ proxy signers can't cooperatively generate a valid threshold proxy signature on a new message m'.*

Proof. The case is similar with Theorem 7. The difference is that the original signer is one of the forgers. First, if the original signer does not change the values of m_w and K_O, the difficulty of forging a valid proxy signature on m' is equivalent to that of Theorem 7 since the original signer also has no the knowledge of x_{P_j} or $\sum x_{P_j}$ of other $t - t''$ proxy signers. Second, if the original signer changes the values of m_w and K_o, correspondingly, the proxy share σ_o is also changed, however, the values of t and n should not be changed, as is obvious in the case. Thus the condition is similar with that of the first case. Therefore, the original signer and any $t'' < t$ proxy signers can't cooperatively generate a valid threshold proxy signature on a new message m'.

Theorem 10. *The original signer and any $t'' < t$ proxy signers can't cooperatively forge another valid threshold proxy signature on the original message m from the proxy signature $(m_w, K_o, m, K, S, ASID)$.*

Proof. The theorem is similar with Theorem 8. The first condition is similar with that of the proof of Theorem 8. Here the kind of proof is omitted. Let us see the second condition. From the verification Equation (15), by fixing some variables of the set $\{m_w, K_o, K, S, ASID\}$, the values of the other variables in the set $\{m_w, K_o, K, S, ASID\}$ will not be able to be obtained because of the difficult discrete logarithm and secure hash function, although the original signer is easy to change the values of m_w and K_o. Thus the theorem holds.

Theorem 11. *Any third party, the original signer and any $t'' < t$ proxy signers can't cooperatively generate a valid threshold proxy signature on a new message m'.*

Proof. The theorem is the same as Theorem 9 since the third party knows less information than the original signer. The proof is the same as that of the Theorem 9.

Theorem 12. *Any third party, the original signer and any* $t'' < t$ *proxy signers can't cooperatively forge another valid threshold proxy signature on the original message m from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. The theorem is the same as Theorem 10 since the third party knows less information than the original signer. The proof is the same as that of the Theorem 10.

Theorem 13. *Any can be convinced of the original signer's agreement on the signed message from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. From the proxy signature verification equation (15), the warrant m_w, the identities and public keys of the original and actual proxy signers are used. In this case, any can be convinced of the original signer's agreement on the signed message from the proxy signature $(m_w, K_o, m, K, S, ASID)$.

Theorem 14. *Any can identify the actual proxy signers from the proxy signature* $(m_w, K_o, m, K, S, ASID)$.

Proof. From the proxy signature verification Equation (15), it can be seen that all actual proxy signers' identities and public keys are used. Therefore, any can identify the actual proxy signers from the proxy signature $(m_w, K_o, m, K, S, ASID)$.

Theorem 15. *Any can distinguish proxy signatures from normal signatures.*

Proof. In the proxy signature verification Equation (15), not only the original signer's public key, but also the actual proxy signers' public keys are used. In normal signature verification equation, only signers' public keys are used. So, any can distinguish proxy signatures from normal signatures.

Theorem 16. *The proxy signers can't repudiate having produced the proxy signature which has ever been signed to any one.*

Proof. As seen in proxy signature verification Equation (15), the actual proxy signers' identities and public keys are used. Thus he can't deny having produced the proxy signature which has ever been signed to any one.

From the above several theorems, we know that the proposed scheme can fulfill the securities of verifiability, strong identifiability, distinguishability, strong unforgeability, strong nonrepudiation, proxy protection and prevention of misuse of proxy signing power. In other words, the proposed scheme can resist equation attacks, collaboration attacks, public key substitution attacks, internal attacks and external attacks. In addition, the certificates of users are not needed in the proposed scheme. If users want to change his private key or the **CA** wants to change users' public key, both have to interact to finish it, or else neither of the two parties can succeed. The verifier only needs to obtain the public keys of the original signer and the proxy signers and needn't verify their validity as the verification of their public keys and the proxy

signature is executed together. Thus, the self-certification of public keys can be realized.

5 Performance Evaluation

To facilitate the performance evaluation, we denote the following notations: T_h : The time for performing a one-way hash function h ; T_{mul} : The time for performing a modular multiplication computation; T_{add} : The time for performing a modular addition computation; T_{inv} : The time for performing a modular inverse computation; T_{pa} : The time for performing a point addition computation; T_{sm} : The time for performing a scalar multiplication computation; $|x|$: The bit-length of an integer x or a point x. The computational overhead and communication cost of the proposed scheme are stated in Table 1 and 2, respectively.

Table 1. Computational overhead of the proposed scheme

Phases	Computational overheads
Registration	User: $T_{pa}+3T_{sm}+T_{add}+2T_h$ The **CA**: $T_{pa}+T_{sm}+T_{mul}+T_{add}+T_h$
Proxy share generation	The original signer: $T_{sm}+2T_{mul}+T_{add}+T_h$ Each proxy signer: $T_{pa}+2T_{sm}+T_{mul}+T_{add}+2T_h$
Proxy signature issuing	Each proxy signer: $(t'-1)T_{pa}+T_{sm}+3T_{mul}+2T_{add}+T_{inv}+T_h$ The clerk: $5t'T_{pa}+7t'T_{sm}+(t'-1)T_{add}+t'T_{inv}+4t'T_h$
Proxy signature verification	$(t'+3)T_{pa}+(t'+3)T_{sm}+(t'+2)T_h$

Table 2. Communication cost of the proposed scheme

Phases	Communication cost
Registration	$2\|G\|+\|n\|+\|ID_i\|$
Proxy share generation	$\|G\|+\|n\|+\|m_w\|$
Proxy signature issuing	$t'\|G\|+t'\|n\|$
Proxy signature verification	$2\|G\|+\|n\|+\|m_w\|+\|m\|+\|ASID\|$
Total [a]	$(t'+3)\|G\|+(t'+2)\|n\|+2\|m_w\|+\|m\|+\|ASID\|$

[a] The total communication cost excludes the registration phase.

6 Conclusions

In the paper, based on Hsu et al's self-certified public key system and Hwang et al's proxy signature scheme, one ECDLP-based threshold proxy signature scheme with

self-certified public key system and non Shamir's secret share protocol has been proposed. As far as we know, it is the first scheme based on ECDLP using self-certified public key system. The proposed scheme can provide needed security properties. In the proxy signature verification phase, the authentication of the original and the proxy signers' public keys and the verification of the threshold proxy signature are executed together. In addition, the computation overhead and communication cost of the proposed scheme are analyzed as well.

References

1. Mambo, M., Usuda, K., Okamoto, E.: Proxy Signature for Delegating Signing Operation. In: Proceedings of the 3th ACM Conference on Computer and Communications Security, pp. 48–57. ACM Press, New York (1996)
2. Li, J.G., Cao, Z.F.: Improvement of a Threshold Proxy Signature Scheme. J. of Computer Research and Development 39(11), 515–518 (2002)
3. Li, J.G., Cao, Z.F., Zhang, Y.C.: Improvement of M-U-O and K-P-W Proxy Signature Schemes. J. of Harbin Institute of Technology (New Series) 9(2), 145–148 (2002)
4. Li, J.G., Cao, Z.F., Zhang, Y.C.: Nonrepudiable Proxy Multi-signature Scheme. J. of Computer Science and Technology 18(3), 399–402 (2003)
5. Li, J.G., Cao, Z.F., Zhang, Y.C., Li, J.Z.: Cryptographic Analysis and Modification of Proxy Multi-signature Scheme. High Technology Letters 13(4), 1–5 (2003)
6. Hsu, C.L., Wu, T.S., Wu, T.C.: New Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. The J. of Systems and Software 58, 119–124 (2001)
7. Hwang, M.S., Lin, I.C., Lu Eric, J.L.: A Secure Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. International J. of Informatica 11(2), 1–8 (2000)
8. Hwang, S.J., Chen, C.C.: Cryptanalysis of Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. Informatica 14(2), 205–212 (2003)
9. Sun, H.M.: An Efficient Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. Computer Communications 22(8), 717–722 (1999)
10. Sun, H.M., Lee, N.Y., Hwang, T.: Threshold Proxy Signature. In: IEEE Proceedings on Ccomputers & Digital Techniques, pp. 259–263. IEEE Press, New York (1999)
11. Zhang, K.: Threshold Proxy Signature Schemes. In: Information Security Workshop, pp. 191–197 (1997)
12. Hsu, C.L., Wu, T.S., Wu, T.C.: Improvement of Threshold Proxy Signature Scheme. Applied Mathematics and Computation 136, 315–321 (2003)
13. Tsai, C.S., Tzeng, S.F., Hwang, M.S.: Improved Nonrepudiable Threshold Proxy Signature Scheme with Known Signers. Informatica 14(3), 393–402 (2003)
14. Hwang, S.J., Shi, C.H.: A Simple Multi-Proxy Signature Scheme. In: Proceeding of the Tenth National Conference on Information Security, Taiwan, pp. 134–138 (2000)
15. Denning, D.E.R.: Cryptography and Data Security. Addison-Wesley, Reading (1983)
16. Pedersen, T.: Distributed Provers with Applications to Undeniable Signatures, p. 547. Springer, New York (1991)
17. Li, L.H., Tzeng, S.F., Hwang, M.S.: Generalization of proxy signature-based on discrete logarithms. Computers & Security 22(3), 245–255 (2003)
18. Hsu, C.L., Wu, T.S.: Efficient proxy signature schemes using self-certified public keys. Applied Mathematics and Computation. In: Press, Corrected Proof, Available online July 9 (2003)

19. Hwang, M.S., Tzeng, S.F., Tsai, C.S.: Generalization of proxy signature based on elliptic curves. Computer Standards & Interfaces 26(2), 73–84 (2004)
20. Girault, M.: Self-certified public keys. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 490–497. Springer, Heidelberg (1991)
21. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
22. National Institute of Standards and Technology (NIST), The digital signature standard proposed by NIST. Communication of the ACM 35(7), 36–40 (1992)
23. Chang, C.C., Hwang, M.S.: Parallel computation of the generating keys for RSA cryptosystems. IEE Electronics Letters 32(15), 1365–1366 (1996)