JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 63-79 (2017)
DOI: 10.6688/JISE.2017.33.1.5

Traceable Threshold Proxy Signature*

KE GU[1,2], YONG WANG[2] AND SHENG WEN[3]
[1]*School of Computer and Communication Engineering*
Changsha University of Science and Technology
Changsha, 410114 P.R. China
[2]*School of Information Science and Engineering*
Central South University
Changsha, 410083 P.R. China
[3]*School of Information Technology*
Deakin University
Melbourne, 3217 Australia
E-mail: {gk4572@163.com}

Threshold proxy signature (TPS) is a variant of proxy signature, which allows that a delegator may delegate his signing rights to many proxy signers. Compared with proxy signature, threshold proxy signature can limitedly prevent that some of proxy signers abuse their signing rights. In this paper, we show a traceable threshold proxy signature (TTPS) frame. Also, we propose a traceable (t, n) threshold proxy signature scheme, which is based on the Waters' signature scheme and the Xiong *et al*.'s threshold signature scheme in the standard model. The proposed scheme is proved to have a security reduction to the CDH assumption and the threshold proxy signature unforgeability under an adaptive chosen message attack, and to have the threshold proxy signature traceability. Compared with other (t, n) threshold proxy signature schemes, the proposed scheme is secure in the standard model.

Keywords: threshold proxy signature, traceability, provable security, CDH, standard model

1. INTRODUCTION

The concept of proxy signature was firstly proposed by Mambo *et al*. in [20]. In the proxy signature schemes [2, 20, 21, 23, 24, 39, 40], delegator (original signer) may delegate his signing rights to proxy signer. If proxy signer gets proper delegation, then proxy signer can represent delegator to exercise his signing rights. Proxy signature is an extension of ordinary signature. Compared with the ordinary signature schemes [3, 5, 22, 30], the proxy signature schemes have four security properties [1], which are unforgeability, non-repudiation, strong identifiability and prevention of misuse. First provable security model of proxy signature was proposed by Boldyreva *et al*. in 2003. Then many researchers and scholars proposed some improved security models for proxy signature [21, 23, 24], which are based on the Boldyreva *et al*.'s model [2]. In 2012, Boldyreva *et al*. summarized the related work [2, 10, 21, 23] about the provable security models of proxy signature in [1], and questioned how to measure adversary's ability in the different security models. In [1], Boldyreva *et al*. proposed a more complete and accurate security model for proxy signature. In their security model, security of a proxy signature scheme needs to be analyzed in four situations, and generating self-proxy signature is considered

Received November 12, 2015; revised May 28, 2016; accepted June 25, 2016.
Communicated by Hung-Min Sun.
* This work is supported by the National Natural Science Foundation of China (No. 61402055, No. 61462048).

as a weak secure situation [1, 23].

Currently, many variants of proxy signature were also proposed, such as multi-proxy signature [4, 7, 10, 14, 16, 25, 29, 31], proxy multi-signature [6, 18, 26, 27], proxy blind signature [15, 19, 41] and so on. In the multi-proxy signature schemes, delegator may delegate his signing rights to many proxy signers. Compared with the proxy signature schemes, there are two types of collusion attacks in the multi-proxy signature schemes, which are collusion attack of proxy signers and collusion attack of original signer and partial proxy signers. Then many provable security models of multi-proxy signature [4, 7, 10, 14, 16, 25, 29, 31] were proposed. Because all proxy signers must participate in signing, the multi-proxy signature schemes are not flexible. Based on the multi-proxy signature schemes, the threshold proxy signature schemes [9, 11-13, 17, 33] were proposed. Compared with the multi-proxy signature schemes, the concept of (t, n) threshold [8, 28, 35, 42] is introduced into the threshold proxy signature schemes. If delegator delegates his signing rights to n proxy signers and the number of proxy signers participating in signing can meet threshold value t, then t proxy signers may compute out a threshold proxy signature. Thus, the threshold proxy signature schemes are more flexible. Compared with multi-proxy signature, threshold proxy signature can be more easily used for many applications, such as distributed systems, grid computing, mobile agent applications, distributed shared object systems, and so on.

Some (t, n) threshold proxy signature schemes [9, 11-13, 17, 33, 34, 36-38] have been proposed. In [11], Huang *et al.* proposed a (t, n) threshold proxy signature scheme based on the Schnorr's scheme. Compared with the existing (t, n) threshold proxy signature schemes, their scheme reduces the amount of computations and communications. However, their scheme has a security weakness, because it can't resist collusion attacks. In [33], Yang *et al.* proposed an efficient threshold proxy signature scheme. However, their scheme is also not secure, because it can't resist frame attack and public-key substitute attack [12]. In [12], Hu *et al.* proposed an improved threshold proxy signature scheme based on the Yang *et al.*'s scheme [33]. In [9, 13, 17, 38], some (t, n) threshold proxy signature schemes were also proposed, but a complete security model of (t, n) threshold proxy signature is still not proposed. Although the papers [9, 13, 17] claim their schemes are secure, their schemes still need to be further studied due to lacking complete security proof. Additionally, in a (t, n) threshold proxy signature scheme, delegator delegates his signing rights to n proxy signers, then t proxy signers of n proxy signers represent n proxy signers to compute a threshold proxy signature, the (t, n) threshold proxy signature scheme has second proxy property. So, we need to consider the following situation. If there is not a trusted third party of computing signatures in a (t, n) threshold proxy signature scheme, and some of t proxy signers participating in signing (include the proxy signer collecting distributed proxy signatures) deny to have participated in signing, then the signature receivers will not know who the t proxy signers participating in signing are. Thus, a secure (t, n) threshold proxy signature scheme must prevent the situation. Obviously, it is very important to trace t proxy signers participating in signing. However, most of the existing (t, n) threshold proxy signature schemes cannot trace t proxy signers participating in signing, the schemes are not adequately secure. Although the schemes of Huang *et al.* and Hu *et al.* [11, 12] also have the limited threshold proxy signature traceability by adding the identities of the actual proxy signers to the final signature, the method still needs to be further studied due to lacking complete security proof. Yang *et al*. [34] proposed a traceable certificateless threshold proxy signature scheme based on bilinear pairings in the random oracle model. However, their scheme traces the *t* proxy signers participating in signing by setting an additional tag and lacks the security proofs. Further, the tag-added method may influence the acceptance of verifiers and breaks the anonymity of *t* proxy signers.

Thus, we propose a traceable (t, n) threshold proxy signature scheme in the standard model, which is based on the Waters' signature scheme [30] and the Xiong *et al*.'s threshold signature scheme [32]. The proposed scheme can be proved to have a security reduction to the CDH assumption and the threshold proxy signature unforgeability under an adaptive chosen message attack, and to have the threshold proxy signature traceability so as to trace *t* proxy signers participating in signing.

2. A FRAME FOR TRACEABLE THRESHOLD PROXY SIGNATURE

Definition 2.1: Traceable Threshold Proxy Signature: Let **TTPS**=(**DS,** ***TDelegate, TProxyKeyGen, TProxySign, TProxyVerify, TProxyTrace***) be a (t, n) threshold proxy signature frame, where **DS=(*Setup, KeyGen, Sign, Verify*)** is an ordinary signature frame. In **TTPS**, all algorithms are described as follows:

1. ***Setup*:** The randomized algorithm inputs a security parameter 1^k, and then outputs all system parameters.
2. ***KeyGen*:** The randomized algorithm generates user's public/private key pair (pk_i, sk_i) with $i\in\{1, 2, \ldots, n, n+1\}$, where i is the indexed number of user, pk_i is the public key of user i, sk_i is the private key of user i.
3. ***Sign*:** The randomized algorithm is a standard signature algorithm. Signer needs to sign a message $\mathcal{M}\in\{0, 1\}^*$. The algorithm inputs $(sk, \mathcal{M})$, and then outputs a standard signature σ, where $\sigma\in\{0, 1\}^*\cup\{\perp\}$, sk is the private key of signer and pk is the corresponding public key.
4. ***Verify*:** The signature receivers verify a standard signature σ. The deterministic algorithm inputs $(\mathcal{M}, pk, \sigma)$, and then outputs the boolean value, ***accept*** or ***reject***.
5. ***TDelegate*:** We assume the user i is a delegator with $i\in\{1, 2, \ldots, n, n+1\}$. The randomized algorithm inputs $(pk_i, PK_{\{1, 2, \ldots, n, n+1\}/i}, sk_i, w_i)$, where $PK_{\{1, 2, \ldots, n, n+1\}/i}$ is a public key list including all users' public keys except for that of the user i, $w_i\subseteq\{0, 1\}^*$ is the warrant of the user i. Then the algorithm outputs a delegation $\delta_{i,j}$ for every proxy signer j with $j\in\{1, 2, \ldots, n, n+1\}/i$.
6. ***TProxyKeyGen*:** The randomized algorithm generates a distributed proxy signing key. *TProxyKeyGen* run by every proxy signer j inputs $(\delta_{i,j}, pk_i, PK_{\{1, 2, \ldots, n, n+1\}/i}, sk_j)$, and then outputs $(w_i, psk_{i,j})$, where $psk_{i,j}$ is a distributed proxy signing key of the proxy signer j in w_i.
7. ***TProxySign*:** We assume t proxy signers of n proxy signers need to sign a message $\mathcal{M}\in\{0, 1\}^*$ *on the signing delegation of the user i*. Without loss of generality, we set the indexed numbers of t proxy signers are $\{1, 2, \ldots, t\}$, then two following steps need to be finished:
   - For every proxy signer j participating in signing, the algorithm run by the proxy signer j inputs $(w_i, psk_{i,j}, \mathcal{M})$, and then outputs a distributed proxy signature $p\sigma_{i,j}$ with

$j \in \{1, 2, \dots, t\}$.

- One proxy signer p collects the distributed proxy signatures of t proxy signers including p own distributed proxy signature with $p \in \{1, 2, \dots, t\}$ and $p \neq j$; then the algorithm run by the proxy signer p inputs ($p\sigma_{i,1}$, $p\sigma_{i,2}$, …, $p\sigma_{i,t}$), and outputs a threshold proxy signature (w_i, $p\sigma_i$), where $p\sigma_i \in \{0, 1\}^* \cup \{\perp\}$; if the number of proxy signers participating in signing are less than t, then the signing procedure will abort.

8. ***TProxyVerify*:** The signature receivers verify a threshold proxy signature(w_i, $p\sigma_i$) on $\mathcal{M} \in \{0, 1\}^*$. The deterministic algorithm inputs ($\mathcal{M}$, pk_i, $PK_{\{1, 2, \dots, n, n+1\}/i}$, w_i, $p\sigma_i$), and then outputs the boolean value, ***accept*** or ***reject***.
9. ***TProxyTrace*:** The trusted authority traces the t proxy signers participating in signing by the traceable threshold proxy signature (w_i, $p\sigma_i$) on $\mathcal{M} \in \{0, 1\}^*$. The deterministic algorithm inputs ($\mathcal{M}$, pk_i, $PK_{\{1, 2, \dots, n, n+1\}/i}$, w_i, $p\sigma_i$), and then outputs the public keys of the t proxy signers participating in signing.

3. PRELIMINARIES

3.1 Bilinear Maps

Let G_1 and G_2 be groups of prime order q and g be a generator of G_1. We say G_2 has an admissible bilinear map, e: $G_1 \times G_1 \rightarrow G_2$ if the following two conditions hold. The map is bilinear; for all a, b, we have $e(g^a, g^b)=e(g, g)^{ab}$. The map is non-degenerate; we must have that $e(g, g) \neq 1$.

3.2 Computational Diffie-Hellman Assumption

Definition 3.1: Computational Diffie-Hellman (CDH) Problem: Let G_1 be a group of prime order q and g be a generator of G_1; for all $(g, g^a, g^b) \in G_1$, with $a, b \in Z_q$, the CDH problem is to compute g^{ab}.

Definition 3.2: The (h, ε)-CDH assumption holds if no h-time algorithm can solve the CDH problem with probability at least ε.

4. TRACEABLE THRESHOLD PROXY SIGNATURE SCHEME

Let **TTPS**=(***Setup, KeyGen, Sign, Verify, TDelegate, TProxyKeyGen, TProxySign, TProxyVerify, TProxyTrace***) be a traceable (t, n) threshold proxy signature scheme. In **TTPS**, all algorithms are described as follows:

TTPS.*Setup*: The algorithm run by the authority system inputs a security parameter 1^k. Additionally, let G_1 and G_2 be groups of prime order q and g be a generator of G_1, and let e: $G_1 \times G_1 \rightarrow G_2$ denote the bilinear map. The size of the group is determined by the security parameter. A hash function, $H:\{0, 1\}^* \rightarrow Z_{1^k \cdot q}$ can be defined and used to generate any integer value in $Z_{1^k \cdot q}$ (where 1^k represents the corresponding decimal number). And all messages will be represented as bit strings of length n_m, then a collision-resistant hash function, $H_1:\{0, 1\}^* \rightarrow \{0, 1\}^{n_m}$ can be defined and used to create messages of the desired length. Then the system parameters are generated as follows. One group element $\theta \in G_1$ and a $(n+1)$-length vector $\Upsilon=(g_i)$ are randomly chosen, with $g_i \in G_1$ and $i \in \{1, 2, \ldots, n, n+1\}$. Additionally, the system chooses a random $\tau' \in G_1$ and a random n_m-length vector $\Phi=(\tau_i)$, whose elements are randomly chosen from G_1. Finally, the system outputs the public parameters, $(G_1, G_2, e, g, \theta, \Upsilon, \tau', \Phi, t, n)$.

TTPS.*KeyGen*: Private Key Generator (PKG) generates all users' public/private key pairs. PKG randomly chooses $sk_i \in Z_q$ as the private key of user i, and computes and outputs the corresponding public key, $pk_i=g^{sk_i}$ with $i \in \{1, 2, \ldots, n, n+1\}$, where i is the indexed number of user. Then PKG respectively sends the private key to the corresponding user by a secure channel.

TTPS.*Sign*: Let $\mathcal{M}$ be the bit string of length n_m representing a message, and let $\mathcal{M} \subseteq \{1, 2, \ldots, n_m\}$ be the set of index d such that $\mathcal{M}[d]=1$, where $\mathcal{M}[d]$ is the d-th bit of $\mathcal{M}$. The algorithm run by signer chooses a random $v \in Z_q$, and computes $Q_1=g_1^{sk} \cdot (\tau' \cdot \prod_{d \in M} \tau_d)^v$ and $Q_2=g^v$, and then outputs a signature $\sigma=(Q_1, Q_2)$.

TTPS.*Verify*: The algorithm run by verifier verifies the signature $\sigma=(Q_1, Q_2)$ by the equation, $e(Q_1, g)=e(g_1, pk) \cdot e(\tau' \cdot \prod_{d \in M} \tau_d, Q_2)$..

TTPS.*TDelegate*: We assume the user i is a delegator with $i \in \{1, 2, \ldots, n, n+1\}$. The algorithm run by the user i inputs $(pk_i, PK_{\{1, 2, \ldots, n, n+1\}/i}, sk_i, w_i)$, where $PK_{\{1, 2, \ldots, n, n+1\}/i}$ is a public key list including all users' public keys except for that of the user i, and w_i is the warrant of the user i. Then the algorithm chooses a random $r_i \in Z_q$, and computes $\lambda_i=g^{\frac{sk_i}{n}} \cdot \theta^{\frac{r_i \cdot H(pk_i, PK_{\{1, 2, \ldots, n, n+1\}/i}, w_i)}{n}}$ and $R_i=g^{r_i}$. Finally, the algorithm outputs a delegation $\delta_{i,j}=(R_i, \lambda_i, w_i)$ for every proxy signer j with $j \in \{1, 2, \ldots, n, n+1\}/i$, and sends $\delta_{i,j}$ to the proxy signer j by a secure channel.

TTPS.*TProxyKeyGen*: The algorithm run by every proxy signer j generates a distributed proxy signing key with $j \in \{1, 2, \ldots, n, n+1\}/i$. In the algorithm, the following steps need to be finished:

1. The algorithm run by every proxy signer j verifies the delegation, $\delta_{i,j}=(R_i, \lambda_i, w_i)$ by the equation $e(\lambda_i^n, g)=e(g_i, pk_i) \cdot (\theta, R_i)^{H(pk_i, PK_{(1, 2, \ldots, n, n+1)/i}, w_i)}$; if the above equation is not correct, then the proxy signer j may require the user i resends the delegation $\delta_{i,j}$.
2. The algorithm run by every proxy signer j computes

$$osk_{i,j} = g_j^{sk_j} \cdot \lambda_i = g_j^{sk_j} \cdot g_i^{\frac{sk_i}{n}} \cdot \theta^{\frac{r_i \cdot H(pk_i, PK_{\{1, 2, \ldots, n, n+1\}/i}, w_i)}{n}}.$$

3. The algorithm run by every proxy signer j randomly chooses one polynomial:

$$f_{i,j}(x) = a_{i,j,1} \cdot x + a_{i,j,2} \cdot x^2 \ldots + a_{i,j,t-1} \cdot x^{t-1},\ a_{i,j,d} \in Z_q,\ d \in \{1, \ldots, t-1\}.$$

Then the algorithm sets $F_{i,j}(x)=osk_{i,j} \cdot \theta^{f_{i,j}(x)}$, and computes $A_{i,j,d}=e(\theta, g)^{a_{i,j,d}}$. Finally, the algorithm broadcasts $A_{i,j,d}$.

4. The algorithm run by every proxy signer j computes $s_{i,j,p}=F_{i,j}(p)$ for every proxy signer p with $p\in\{1,\ldots, n, n+1\}/i$, and if $j\neq p$, then the algorithm sends $s_{i,j,p}$ to the corresponding proxy signer p by a secure channel.
5. After every proxy signer p receives $s_{i,j,p}$ from other proxy signer j, the algorithm run by every proxy signer p verifies it by the equation $e(s_{i,j,p}, g)=e(g_j, pk_j)\cdot e(\lambda_i, g)\cdot\prod_{d=1}^{t-1}(A_{i,j,d})^{p^d}$; if the equation is correct, then the whole procedure continues, otherwise the proxy signer p may require that the corresponding proxy signer j resends $s_{i,j,p}$; then thealgorithm run by every proxy signer p respectively computes averifying data, $B_{i,j}=\prod_{p=1,p\neq i}^{n+1}(e(g_l, pk_l)\cdot e(\lambda_i, \text{g})\prod_{d=1}^{t-1}\cdot(A_{i,l,d})^{j^d})$ for each distributed proxy signature computed by other proxy signer j.
6. Finally, the algorithm run by every proxy signer j computes and outputs a distributed proxy signing key $psk_{i,j}=\prod_{p=1,p\neq i}^{n+1} s_{i,p,j}$ according to $s_{i,j,p}$.

TTPS.*TProxySign*: We assume t proxy signers of n proxy signers need to sign a message $\mathcal{M}$ on the signing delegation of the user i. Let $\mathcal{M}$ be the bit string of length n_m representing a message, and let $\mathcal{M}\subseteq\{1, 2, \ldots, n_m\}$ be the set of index d such that $\mathcal{M}[d]=1$, where $\mathcal{M}[d]$ is the d-th bit of $\mathcal{M}$. Additionally, without loss of generality, we set the indexed numbers of t proxy signers are $\{1, 2, \ldots, t\}$, then the following steps need to be finished:
1. For every proxy signer j participating in signing with $j\in\{1, 2, \ldots, t\}$, the algorithm run by the proxy signer j computes $Q_{i,j,1}=psk_{i,j}\cdot\left(\tau'\cdot\prod_{d\in M}\tau_d\right)^{r_j}$ and $Q_{i,j,2}=g^{r_j}$, and then outputs a distributed proxy signature $p\sigma_{i,j}=(Q_{i,j,1}, Q_{i,j,2})$.
2. One proxy signer p collects the distributed proxy signatures of t proxy signers including p own distributed proxy signature with $p\in\{1, 2, \ldots, t\}$ and $p\neq j$. Then the following two steps are finished. Firstly, the algorithm run by the proxy signer p verifies each distributed proxy signature $p\sigma_{i,j}$ by the equation, $\underline{B}_{i,j}=e(Q_{i,j,2}, g)\cdot e(Q_{i,j,2}, \tau'\cdot\prod_{d\in M}\tau_d)^{-1}$ for every proxy signer j; if the equation is correct, then the whole procedure continues, otherwise the result indicates the proxy signer j is not honest, the procedure may abort. Secondly, the algorithm run by the proxy signer p computes $Q_{i,1}=\prod_{j=1}^{t}\left(Q_{i,j,1}\right)^{L_j}g_p^{-sk_p}$ and $Q_{i,2}=\prod_{j=1}^{t}\left(Q_{i,j,2}\right)^{L_j}$, and sets $Q_{i,3}=g^{r_i}$, where $L_j=\prod_{d=1,d\neq j}^{t}\frac{d}{d-j}$ is the corresponding Lagrange interpolation coefficient, then the algorithm outputs a traceable threshold proxy signature $(w_i, p\sigma_i=(Q_{i,1}, Q_{i,2}, Q_{i,3}))$ on $\mathcal{M}$. Additionally, if the number of proxy signers participating in signing are less than t, the procedure aborts.

TTPS.*TProxyVerify*: A signature receiver verifies the traceable threshold proxy signature $(w_i, p\sigma_i=(Q_{i,1}, Q_{i,2}, Q_{i,3}))$ on $\mathcal{M}$. The algorithm inputs $(\mathcal{M}, pk_i, PK_{\{1, 2, \ldots, n, n+1\}/i}, w_i, p\sigma_i)$, and then computes the equation

$$e\left(Q_{i,1}, g\right)\cdot e(g_p, pk_p)\cdot e\left(Q_{i,2}, \tau'\cdot\prod_{d\in M}\tau_d\right)^{-1}=\prod_{j=1}^{n+1}e(g_j, pk_j)\cdot e(\theta, Q_{i,3})^{H\left(pk_i, PK_{\{1,2,\ldots,n,n+1\}/i}, w_i\right)},$$

where pk_p is the public key of the proxy signer p collecting distributed proxy signatures. If the equation is correct, then the algorithm outputs the boolean value *accept*, otherwise the algorithm outputs the boolean value *reject*.

TTPS.*TProxyTrace*: The trusted authority traces the t proxy signers participating in signing by the traceable threshold proxy signature (w_i, $p\sigma_i$) on $\mathcal{M}$. Then the following steps are finished:

1. For the public key of the potential proxy signer p collecting the distributed proxy signatures ($p \in \{1, 2, \ldots, t\}$), the algorithm computes the following equation:

$$e(g_p, pk_p) = \frac{\prod_{j=1}^{n+1} e(g_j, pk_j) \cdot e(\theta, Q_{i,3})^{H\left(pk_i, PK_{\{1,2,\ldots,n,n+1\}/i}, w_i\right)}}{e\left(Q_{i,1}, g\right) \cdot e\left(Q_{i,2}, \tau' \cdot \prod_{d \in M} \tau_d\right)^{-1}},$$

   If the above equation is correct, then the algorithm outputs the public key of the proxy signer p collecting the distributed proxy signatures and runs into the next step, otherwise the algorithm outputs $\perp$.

2. By the proxy signer p collecting the distributed proxy signatures of other $t-1$ proxy signers, the algorithm gets all distributed proxy signatures $p\sigma_{i,j}$=($Q_{i,j,1}$, $Q_{i,j,2}$) of t proxy signers ($j \in \{1, 2, \ldots, t\}$). And the following two steps are finished:

(a) The algorithm computes $Q_{i,1}'' = \prod_{j=1}^{t} \left(Q_{i,j,1}\right)^{L_j}$ and $Q_{i,2}' = \prod_{j=1}^{t} (Q_{i,j,1})^{L_j}$, where $L_j = \prod_{d=1, d\neq j}^{t} \frac{d}{d-j}$ is the corresponding lagrange interpolation coefficient, and then checks the two equations $e(Q_{i,1}', g) = e(Q_{i,1}', g) \cdot e(g_p, pk_p)$ and $Q_{i,2}' = Q_{i,2}$. If the above two equations are correct, then the algorithm continues, otherwise the results indicate the real proxy signer p is not honest, the procedure may abort and the algorithm outputs $\perp$.

(b) The algorithm verifies each distributed proxy signature $p\sigma_{i,j}$ by the equation, $B_{i,j} = e(Q_{i,j,2}, g) \cdot e(Q_{i,j,2}, \tau' \cdot \prod_{d \in M} \tau_d)^{-1}$ for every real proxy signer j, then the indexed numbers of other $t-1$ proxy signers participating in signing may be confirmed, where the algorithm gets $B_{i,j}$ by the proxy signer p or, one or several of other $n-1$ proxy signers in this scheme. If the above equation is correct, then the algorithm outputs the corresponding public key of of the real proxy signer participating in signing; If the number of outputting the indexed number are less than $t-1$, the result still indicates the real proxy signer p is not honest, the algorithm outputs $\perp$.

5. ANALYSIS OF THE PROPOSED SCHEME

5.1 Correctness

Claim 5.1: In the algorithm **TTPS.*TProxyVerify***, the signature receivers may verify a traceable threshold proxy signature (w_i, $p\sigma_i$=($Q_{i,1}$, $Q_{i,2}$, $Q_{i,3}$)) on $\mathcal{M}$ by the equation

$$e\left(Q_{i,1}, g\right) \cdot e(g_p, pk_p) \cdot e\left(Q_{i,2}, \tau' \cdot \prod_{d \in M} \tau_d\right)^{-1} = \prod_{j=1}^{n+1} e(g_j, pk_j) \cdot e(\theta, Q_{i,3})^{H\left(pk_i, PK_{\{1,2,\ldots,n,n+1\}/i}, w_i\right)}.$$

Proof*:** Firstly, in **TTPS.*TProxyKeyGen, If we set $Y = \prod_{j=1, j\neq i}^{n+1} osk_{i,j}$, then we may compute $Y = \prod_{j=1}^{t} \left(psk_{i,j}\right)^{L_j}$ according to the Shamir's sharing scheme [28] and the Xiong *et al.*'s threshold signature scheme [32], where $psk_{i,j} = \prod_{p=1, p\neq i}^{n+1} s_{i,p,j}$ and $L_j = \prod_{d=1, d\neq j}^{t} \frac{d}{d-j}$. Then we may get the equation $e\left(\prod_{j=1}^{t}\left(psk_{i,j}\right)^{L_j}, g\right) = e\left(\prod_{j=1, j\neq i}^{n+1} osk_{i,j}, g\right)$.

Secondly, the following computation may be made in **TTPS.*TProxyVerify***:

$$
\begin{aligned}
& e\left(Q_{i,1}, g\right)\cdot e(g_p, pk_p)\cdot e\left(Q_{i,2}, \tau'\cdot\prod_{d\in M}\tau_d\right)^{-1} \\
&= e\left(\prod_{j=1}^{t}\left(Q_{i,j,1}\right)^{L_j}\cdot g_p^{-sk_p}, g\right)\cdot e(g_p, pk_p)\cdot e\left(\prod_{j=1}^{t}\left(Q_{i,j,2}\right)^{L_j}, \tau'\cdot\prod_{d\in M}\tau_d\right)^{-1} \\
&= e\left(\prod_{j=1}^{t}\left(Q_{i,j,1}\right)^{L_j}, g\right)\cdot e\left(\prod_{j=1}^{t}\left(Q_{i,j,2}\right)^{L_j}, \tau'\cdot\prod_{d\in M}\tau_d\right)^{-1} \\
&= e\left(\prod_{j=1}^{t}\left(psk_{i,j}\right)^{L_j}, g\right) = e\left(\prod_{j=1, j\neq i}^{n+1} osk_{i,j}, g\right) = e\left(\prod_{j=1, j\neq i}^{n+1}\left(g_j^{sk_j}\cdot g_i^{\frac{sk_i}{n}}\cdot\theta^{\frac{r_i\cdot H\left(pk_i, PK_{\{1,2,\dots,n,n+1\}/i}, w_i\right)}{n}}\right), g\right) \\
&= \prod_{j=1, j\neq i}^{n+1} e\left(g_j, g^{sk_j}\right)\cdot e\left(g_i, g^{sk_i}\right)\cdot e\left(\theta, g^{r_i\cdot H\left(pk_i, PK_{\{1,2,\dots,n,n+1\}/i}, w_i\right)}\right) = \prod_{j=1}^{n+1} e\left(g_j, pk_j\right)\cdot e\left(\theta, Q_{i,3}\right)^{H\left(pk_i, PK_{\{1,2,\dots,n,n+1\}/i}, w_i\right)}.
\end{aligned}
$$

Thus, the claim follows.

5.2 Efficiency

In this paper, we compare the proposed scheme (the scheme of Section 4) with the threshold proxy signature scheme proposed by Huang *et al.* [11], the traceable certificateless threshold proxy signature scheme proposed by Yang *et al.* [34], the threshold proxy signature scheme proposed by Hu *et al.* [12] and the unidirectional threshold proxy re-signature scheme proposed by Hu *et al.* [38] by Table 1. From the description of Table 1, we may know that the signature length of our proposed scheme is shorter than those of three of the other four schemes, and although our proposed scheme increases some computation costs, our proposed scheme is constructed in the standard model and has the full threshold proxy signature traceability.

Table 1. Comparison of five schemes.

	The length of signature	Model	Assumptions	Traceability
Scheme [11]	$2\cdot\|Z_q\|+t\cdot\|ID\|+\|w\|$	/	DLP and OWHF	partial
Scheme [34]	$(2\cdot t+4)\cdot\|G_1\|+\|w\|$	random oracle model	CDH and CHF	full
Scheme [12]	$2\cdot\|G_1\|+\|Z_q\|+t\cdot\|ID\|+\|w\|$	/	DLP and OWHF	partial
Scheme [38]	$3\cdot\|G_1\|$	standard model	CDH and CHF	no
Our Scheme	$3\cdot\|G_1\|+\|w\|$	standard model	CDH and CHF	full

Caption: $|Z_q|$ represents the length of element in Z_q, $|ID|$ represents the length of identity, $|w|$ represents the length of warrant, $|G_1|$ represents the length of element in G_1, DLP represents discrete logarithm problem, OWHF represents one way hash function, CHF represents cryptographic hash function.

5.3 Unforgeability

Theorem 5.1: The scheme of Section 4 is (j, ε, q_e, q_s)-*secure*, assuming that the (h', ε')-CDH assumption holds in G_1, where:

$$\varepsilon' = \frac{\left[12\cdot(n_m+1)\cdot q_s - 3\cdot q_e\right]\cdot\varepsilon}{64\cdot q\cdot(n_m+1)^2\cdot q_s^2},\ q_e < 4\cdot(n_m+1)\cdot q_s,$$
$$h' = h + O\Big(q_e\cdot\big(6\cdot C_{mul}+9\cdot C_{exp}\big)+q_s\cdot\Big[\big(4\cdot n_m+4\cdot|U|+9\big)\cdot C_{mul}+\big(4\cdot|U|+18\big)\cdot C_{exp}\Big]\Big),$$

and q_e is the maximal number of proxy signing key queries, q_s is the maximal number of signature queries, n_m is the length of message represented as bit strings, $|U|$ is the size of U which is the set of no corrupted users, C_{mul} and C_{exp} are respectively the time for a multiplication and an exponentiation in G_1.

***Proof*:** Let **TTPS** be a traceable (t, n) threshold proxy signature scheme on n+1 users. Additionally, let $\mathcal{A}$ be an $(h, \varepsilon, q_e, q_s)$-adversary attacking **TTPS**. From the adversary $\mathcal{A}$, we construct an algorithm $\mathcal{B}$, for $(g, g^a, g^b)\in G_1$, the algorithm $\mathcal{B}$ is able to use $\mathcal{A}$ to compute g^{ab}. Thus, we assume the algorithm $\mathcal{B}$ can solve the CDH with probability at least ε' and in time at most h', contradicting the (h', ε')-CDH assumption. Based on the Boldyreva *et al.*'s work [1], such a simulation may be created in the following way:

Setup: The algorithm $\mathcal{B}$ inputs a security parameter 1^k. And the algorithm $\mathcal{B}$ sets l_m=4q_s, and randomly chooses $k_m\in Z_{l_m}$, with $0\le k_m\le n_m$. We will assume that $l_m\cdot(n_m+1) < q$ and $q_e<4\cdot(n_m+1)\cdot q_s$ for the given values of q_e, q_s, n_m. Then the algorithm $\mathcal{B}$ chooses $x'\in Z_{l_m}$ and a vector $X=(x_i)$ of length n_m, with $x_i\in Z_{l_m}$ for all i, and chooses $y'\in Z_q$ and a vector $Y=(y_i)$ of length n_m, with $y_i\in Z_q$ for all i. And let $\mathcal{M}\subseteq\{1, 2,\dots, n_m\}$ be the set of index β such that $\mathcal{M}[\beta]=1$, where $\mathcal{M}[\beta]$ is the β-th bit of $\mathcal{M}$. To make the notation easier to follow, the following two functions are defined for a message $\mathcal{M}$:$F(\mathcal{M})$= $F(\mathcal{M}) = x'+\sum_{\beta\in M} x_\beta - l_m\cdot k_m$, $J(\mathcal{M})=y'+\sum_{\beta\in M} y_\beta$. Also, to make our description simpler, we assume that the indexed numbers of all no corrupted users are $\{1, 2, \dots, |U|\}$, and the indexed numbers of all corrupted users are $\{|U|+1, |U|+2, \dots, |\Omega|\}$, so we have that the no corrupted user $u_i^*\in U$ with $i\in\{1, 2, \dots, |U|\}$, and the corrupted user $d_i\in D$ with $i\in\{|U|+1, |U|+2, \dots, |\Omega|\}$, where Ω is a set which includes n+1 users, the user $u_i\in\Omega$ with $i\in\{1, 2, \dots, n, n+1\}$; D is a set which includes all corrupted users with $D\subset\Omega$; U is a set which includes all no corrupted users with $U\subseteq\Omega$; $|\Omega|=n+1$ and $|\Omega|-|U|=|D|=t$. Additionally, to make our proof easier to understand, we set $u_1^*\in U$ as a challenger in the security game.

Then the algorithm $\mathcal{B}$ constructs the public parameters by making the following assignments. The algorithm $\mathcal{B}$ sets $g_1=g^b$ and $b\in Z_q$($\mathcal{B}$ doesn't know b), and randomly chooses $\ell_i\in Z_q$ for all i with $i\in\{2, 3, \dots, |U|\}$, sets $g_i=g_1^{\ell_i}$ for all i with $i\in\{2, 3, \dots, |U|\}$, and randomly chooses $g_j\in G_1$ with $j\in\{|U|+1, |U|+2, \dots, n+1\}$, and then generates a $(n+1)$-length vector $\Upsilon=(g_i)$ with $i\in\{1, 2, \dots, n+1\}$. And then $\mathcal{B}$ chooses $\ell_0\in Z_q$ and sets $\theta=g_1^{\ell_0}\cdot g$, and computes $\tau'=g_1^{-l_m\cdot k_m+x'}\cdot g^{y'}$, $\tau_\beta=g_1^{x_\beta}\cdot g^{y_\beta}$, τ_β, with $1\le\beta\le n_m$. Then all public parameters $(G_1, G_2, e, g, \theta, \Upsilon, \tau', (\tau_\beta), t, n)$ are outputted to the adversary $\mathcal{A}$, where $\tau'\cdot\prod_{\beta\in M}\tau_\beta=g^{F(\mathcal{M})}\cdot g^{J(\mathcal{M})}$. And the algorithm $\mathcal{B}$ sets $pk_1=g^a$ as the public key of u_1^* with $a\in Z_q$ ($\mathcal{B}$ doesn't know a), and sets $pk_i= g^{sk_i}= g^{a\cdot\ell_i}$as the public key of no corrupted user u_i^* for all i with $i\in\{2, 3,\dots, |U|\}$, and runs the algorithm ***KeyGen*** to generate all corrupted users' public/private key pairs, $(pk_j=g^{sk_j}, sk_j\in Z_q)$ ← ***KeyGen***(*parameters*), with $j\in\{|U|+1, |U|+2, \dots, n+1\}$. Lastly, all corrupted users' public/private key pairs, the public keys of all no corrupted users are passed to the adversary $\mathcal{A}$.

Queries: When running the adversary $\mathcal{A}$, both proxy signing key queries and signature queries can occur. The algorithm $\mathcal{B}$ answers these in the following way:

- **ProxyKey Queries:**

(I) A no corrupted user u_1^* is a delegator and is also simulated as n proxy signers, so the simulator simulating as u_1^* needs to delegate his signing rights to n oneself and generate a proxy signing key (u_1^* is also one proxy signer collecting distributed proxy signatures). Because the algorithm $\mathcal{B}$ does not know the private key of u_1^*, the algorithm randomly chooses $r \in Z_q$, and computes $psk = pk_1^{-\frac{n}{\ell_0}} \cdot \theta^r$, $R = (pk_1^{-\frac{n}{\ell_0}} \cdot g^r)^{\frac{1}{H(pk_1, PK_{\{1\}}, w)}}$, where $\mathcal{B}$ gets the warrant w by the query of $\mathcal{A}$, $PK_{\{1\}}$ is a public key list including n public keys of u_1^*. Setting $r' = \left(r - \frac{n \cdot a}{\ell_0}\right) \cdot \frac{1}{H(pk_1, PK_{\{1\}}, w)}$, $psk = g_1^{n \cdot a} \cdot \theta^{r' \cdot H(pk_1, PK_{\{1\}}, w)}$ and $R = g^{r'}$. So psk is a valid proxy signing key for threshold self-proxy signature of u_1^*. If $\ell_0 \cdot H(pk_1, PK_{\{1\}}, w) = 0 \bmod q$, then the above computation cannot be performed and the simulator will abort; otherwise a valid proxy signing key psk and the information R are passed to the adversary $\mathcal{A}$.

(II) A corrupted user d_i is a delegator with $i \in \{|U|+1, |U|+2, \ldots, n+1\}$, so a corrupted user d_i needs to delegate his signing rights to other n true users and generate a proxy signing key(we assume another corrupted user d_p is one proxy signer collecting distributed proxy signatures with $p \neq i$ and $p \in \{|U|+1, |U|+2, \ldots, n+1\}$). The algorithm $\mathcal{B}$ does not know the private keys of all no corrupted user u_j^* with $j \in \{1, 2, \ldots, |U|\}$, but know the private keys of all corrupted users and can get the warrant w by the query of $\mathcal{A}$, so $\mathcal{B}$ randomly chooses $r \in Z_q$, and then computes

$$psk = pk_1^{-\frac{1}{\ell_0}} \cdot \prod_{j=2}^{|U|} \left(pk_j^{-\frac{\ell_j}{\ell_0}} \right) \cdot \prod_{j=|U|+1, j \neq p}^{n+1} \left(g_j^{sk_j} \right) \cdot \theta^r, \; R = \left(pk_1^{-\frac{1}{\ell_0}} \cdot \prod_{j=2}^{|U|} \left(pk_j^{-\frac{\ell_j}{\ell_0}} \right) \cdot g^r \right)^{\frac{1}{H(pk_i, PK_{\{1,2,\ldots,n,n+1\}/i}, w)}},$$

where $PK_{\{1, 2, \ldots, n, n+1\}/i}$ is a public key list including n proxy signers' public keys. Similarly, setting $r' = (r - a \cdot (\frac{1}{\ell_0} + \sum_{j=2}^{|U|} \frac{(\ell_j)^2}{\ell_0})) \cdot \frac{1}{H(pk_i, PK_{\{1, 2, \ldots, n, n+1\}/i}, w)}$, we can get that $psk = \prod_{j=1, j \neq p}^{n+1} (g_j^{sk_j}) \cdot \theta^{r' \cdot H(pk_1, PK_{\{1, 2, \ldots, n, n+1\}/i}, w)}$ and $R = g^{r'}$. So psk is a valid proxy signing key for threshold proxy signature of d_i, where we assume d_p is one proxy signer collecting distributed proxy signatures. If $\ell_0 \cdot H(pk_i, PK_{\{1, 2, \ldots, n, n+1\}}, w) = 0 \bmod q$, then the above computation cannot be performed and the simulator will abort; otherwise a valid proxy signing key psk and the information R are passed to the adversary $\mathcal{A}$.

(III) A no corrupted user u_1^* is a delegator, so the simulator simulating as u_1 needs to delegate his signing rights to other n true users and generate a proxy signing key(we assume one corrupted user d_p is one proxy signer collecting distributed proxy signatures with $p \in \{|U|+1, |U|+2, \ldots, n+1\}$). The algorithm $\mathcal{B}$ does not know the private keys of all no corrupted user, but know the private keys of all corrupted users and can get the warrant w by the query of $\mathcal{A}$, so $\mathcal{B}$ randomly chooses $r \in Z_q$, and then computes

$$psk = pk_1^{-\frac{1}{\ell_0}} \cdot \prod_{j=2}^{|U|} \left(pk_j^{-\frac{\ell_j}{\ell_0}} \right) \cdot \prod_{j=|U|+1, j \neq p}^{n+1} \left(g_j^{sk_j} \right) \cdot \theta^r, \; R = \left(pk_1^{-\frac{1}{\ell_0}} \cdot \prod_{j=2}^{|U|} \left(pk_j^{-\frac{\ell_j}{\ell_0}} \right) \cdot g^r \right)^{\frac{1}{H(pk_1, PK_{\{2,\ldots,n,n+1\}}, w)}},$$

where $PK_{\{2, \ldots, n, n+1\}}$ is a public key list including n proxy signers' public keys. Similarly, psk is a valid proxy signing key for threshold proxy signature of u_1^*, where we assume d_p is one proxy signer collecting distributed proxy signatures. If $\ell_0 \cdot H(pk_1, PK_{\{2, \ldots\ldots, n, n+1\}}, w)=0 \bmod q$, then the above computation cannot be performed and the simulator will abort; otherwise a valid proxy signing key psk and the information R are passed to the adversary $\mathcal{A}$.

- **Signature Queries:**

(I) Consider a query for a threshold self-proxy signature of u_1^* on $\mathcal{M}$ and w. If $F(\mathcal{M}) \neq 0 \bmod q$, then the simulation continues and u_1^* is simulated as n proxy signers. The algorithm $\mathcal{B}$ randomly chooses $r, v \in Z_q$, and computes and outputs a threshold self-proxy signature $(w, p\sigma(Q_1, Q_2, Q_3))$ of u_1^* (u_1^* is also one proxy signer collecting distributed proxy signatures), where:

$$Q_1 = pk_1^{-\frac{n \cdot J(\mathcal{M})}{F(\mathcal{M})}} \cdot \theta^{r \cdot H\left(pk_1, PK_{\{1\}}, w\right)} \cdot \left(\tau' \cdot \prod_{\beta \in M} \tau_\beta\right)^v, Q_2 = pk_1^{-\frac{n}{F(\mathcal{M})}} \cdot g^v \text{ and } Q_3 = g^r.$$

If we set $\lambda = v - \dfrac{n \cdot a}{F(\mathcal{M})}$, then

$$p\sigma(Q_1, Q_2, Q_3) = \left(g_1^{n \cdot a} \cdot \theta^{r \cdot H\left(pk_1, PK_{\{1\}}, w\right)} \cdot \left(\tau' \cdot \prod_{\beta \in M} \tau_\beta\right)^\lambda, \; g^\lambda, \; g^r\right).$$

And $(w, p\sigma(Q_1, Q_2, Q_3))$ is passed to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ can verify it. To the adversary $\mathcal{A}$, all threshold self-proxy signatures computed by the algorithm $\mathcal{B}$ will be indistinguishable from the signatures generated by a true challenger (or the user u_1^*). As [22], $F(\mathcal{M}) \neq 0 \bmod l_m$, will ensure that a threshold self-proxy signature can be constructed.

(II) Consider a query for a threshold proxy signature of d_i on $\mathcal{M}$ and w, with $i \in \{|U|+1, |U|+2, \ldots, n+1\}$. If $F(\mathcal{M}) \neq 0 \bmod q$, then the simulation continues. Thus, the algorithm $\mathcal{B}$ randomly chooses $r, v \in Z_q$, and then computes and outputs a threshold proxy signature $(w, p\sigma(Q_1, Q_2, Q_3))$ of d_i (we assume that one corrupted user d_p is one proxy signer collecting distributed proxy signatures with $p \neq i$ and $p \in \{|U|+1, |U|+2, \ldots, n+1\}$), where

$$Q_1 = pk_1^{-\frac{J(\mathcal{M})}{F(\mathcal{M})}} \cdot \prod_{j=2}^{|U|}\left(pk_j^{-\frac{\ell_j \cdot J(\mathcal{M})}{F(\mathcal{M})}}\right) \cdot \prod_{j=|U|+1, j \neq p}^{n+1}\left(g_j^{sk_j}\right) \cdot \theta^{r \cdot H\left(pk_i, PK_{\{1,2,\ldots,n,n+1\}/i}, w\right)} \cdot \left(\tau' \cdot \prod_{\beta \in M} \tau_\beta\right)^v,$$

$$Q_2 = pk_1^{-\frac{1}{F(\mathcal{M})}} \cdot \prod_{j=2}^{|U|}\left(pk_j^{-\frac{\ell_j}{F(\mathcal{M})}}\right) \cdot g^v \text{ and } Q_3 = g^r.$$

If we set $\lambda = v - a \cdot \left(\dfrac{1}{F(\mathcal{M})} + \sum_{j=2}^{|U|} \dfrac{(\ell_j)^2}{F(\mathcal{M})}\right)$, then $(w, p\sigma=(Q_1, Q_2, Q_3))$ is a valid threshold proxy signature. To the adversary $\mathcal{A}$, all threshold proxy signatures computed by the algorithm $\mathcal{B}$ will be indistinguishable from the signatures generated by the true proxy signers. Similarly, $F(\mathcal{M}) \neq 0 \bmod l_m$, will ensure that a threshold proxy signature can be constructed.

(III) Consider a query for a threshold proxy signature of u_1^* on $\mathcal{M}$ and w. If $F(\mathcal{M})\neq 0$ mod q, then the simulation continues. Thus, the algorithm $\mathcal{B}$ randomly chooses r, $v\in Z_q$, and then computes and outputs a threshold proxy signature(w, $p\sigma=(Q_1, Q_2, Q_3)$)of u_1^*(we assume that one corrupted user d_p is one proxy signer collecting distributed proxy signatures with $p\in\{|U|+1, |U|+2, \ldots, n+1\}$), where:

$$Q_1 = pk_1^{-\frac{J(\mathcal{M})}{F(\mathcal{M})}}\cdot\prod_{j=2}^{|U|}\left(pk_j^{-\frac{\ell_j\cdot J(\mathcal{M})}{F(\mathcal{M})}}\right)\cdot\prod_{j=|U|+1,j\neq p}^{n+1}\left(g_j^{sk_j}\right)\cdot\theta^{r\cdot H\left(pk_1,PK_{\{2,\ldots,n,n+1\}},w\right)}\cdot\left(\tau'\cdot\prod_{\beta\in M}\tau_\beta\right)^{v},$$

$$Q_2 = pk_1^{-\frac{1}{F(\mathcal{M})}}\cdot\prod_{j=2}^{|U|}\left(pk_j^{-\frac{\ell_j}{F(\mathcal{M})}}\right)\cdot g^{v}\ \text{and}\ Q_3=g^r.$$

Similarly, (w, $p\sigma=(Q_1, Q_2, Q_3)$) can be verified by the adversary $\mathcal{A}$. To the adversary $\mathcal{A}$, all threshold proxy signatures computed by the algorithm $\mathcal{B}$ will be indistinguishable from the signatures generated by the true proxy signers. Similarly, $F(\mathcal{M})\neq 0$ mod l_m, will ensure that a threshold proxy signature can be constructed.

Forgery: If the algorithm $\mathcal{B}$ does not abort as a consequence of one of the queries above, the adversary $\mathcal{A}$ will, with probability at least ε, return a forgery according to the following situation.

(I) The adversary $\mathcal{A}$ returns a no corrupted user u_1^*, a message $\mathcal{M}^*$, and a valid threshold self-proxy signature forgery, (w^*, $p\sigma^*=(Q_1^*, Q_2^*, Q_3^*)$) with

$$p\sigma^*=(Q_1^*, Q_2^*, Q_3^*)=\left(g_1^{n\cdot a}\cdot\theta^{r^*\cdot H\left(pk_1,PK_{\{1\}},w^*\right)}\cdot\left(\tau'\cdot\prod_{\beta\in M}\tau_\beta\right)^{\lambda^*},\ g^{\lambda^*},\ g^{r^*}\right).$$

If $\ell_0\cdot H(pk_1, PK_{\{1\}},w^*)\neq 0$ mod q or $F(\mathcal{M}^*)\neq 0$ mod q, then the algorithm $\mathcal{B}$ will abort. If $\ell_0\cdot H(pk_1, PK_{\{1\}},w^*)\neq 0$ mod q and $F(\mathcal{M}^*)=0$ mod q, then the algorithm $\mathcal{B}$ computes and outputs

$$\left(\frac{Q_1^*}{\left(g^{r^*}\right)^{H\left(pk_1,PK_{\{1\}},w^*\right)}\cdot\left(g^{\lambda^*}\right)^{J(\mathcal{M}^*)}}\right)^{\frac{1}{n}}=\left(\frac{g_1^{n\cdot a}\cdot\theta^{r^*\cdot H\left(pk_1,PK_{\{1\}},w^*\right)}\cdot\left(\tau'\cdot\prod_{\beta\in M}\tau_\beta\right)^{\lambda^*}}{\left(g^{r^*}\right)^{H\left(pk_1,PK_{\{1\}},w^*\right)}\cdot\left(g^{\lambda^*}\right)^{J(\mathcal{M}^*)}}\right)^{\frac{1}{n}}=\left(g_1^{n\cdot a}\right)^{\frac{1}{n}}=g_1^{a}=g^{ab}$$

which is the solution to the given CDH problem.

(II) The adversary $\mathcal{A}$ returns a corrupted user d_i with $i\in\{|U|+1, |U|+2\ldots\ldots n+1\}$, a message $\mathcal{M}^*$, and a valid threshold proxy signature forgery, (w^*, $p\sigma^*=(Q_1^*, Q_2^*, Q_3^*)$) with

$$p\sigma^*=(Q_1^*, Q_2^*, Q_3^*)=\left(\prod_{j=1,j\neq p}^{n+1}\left(g_j^{sk_j}\right)\cdot\theta^{r^*\cdot H\left(pk_i,PK_{\{1,2,\ldots,n,n+1\}/i},w^*\right)}\cdot\left(\tau'\cdot\prod_{\beta\in M}\tau_\beta\right)^{\lambda^*},\ g^{\lambda^*},\ g^{r^*}\right),$$

where we assume that one corrupted user d_p is one proxy signer collecting distributed proxy signatures with $p\neq i$ and $p\in\{|U|+1, |U|+2,\ldots\ldots, n+1\}$. Similarly, we can know $p\sigma^*$is the solution to the given CDH problem.

(III) The adversary $\mathcal{A}$ returns a no corrupted user u_1^*, a message $\mathcal{M}^*$, and a valid threshold proxy signature forgery, (w^*, $p\sigma^*=(Q_1^*, Q_2^*, Q_3^*)$) with

$$p\sigma^*=(Q_1^*, Q_2^*, Q_3^*)=\left(\prod_{j=1, j\neq p}^{n+1}\left(g_j^{sk_j}\right)\cdot\theta^{r^*\cdot H\left(pk_1, PK_{\{2,\ldots,n,n+1\}}, w^*\right)}\cdot\left(\tau'\cdot\prod_{\beta\in M}\tau_\beta\right)^{\lambda^*}, g^{\lambda^*}, g^{r^*}\right),$$

where we assume that one corrupted user d_p is one proxy signer collecting distributed proxy signatures with $p\in\{|U|+1, |U|+2, \ldots, n+1\}$. Similarly, we can know $p\sigma^*$ is the solution to the given CDH problem.

Now, we analyze the probability of the algorithm $\mathcal{B}$ not aborting. To make the analysis simpler, we will define the events E_i, E^*, η_j, η^* as E_i: $\ell_0\cdot H(*, *, w_i)\neq 0 \bmod q$, with $i=1, 2, \ldots, q_e$, where q_e is the number of proxy signing key queries; E^*: $\ell_0\cdot H(*, *, w_i)\neq 0 \bmod q$; η_j: $F(\mathcal{M}_j)\neq 0 \bmod l_m$, with $j=1, 2, \ldots, q_s$, where q_s is the number of signature queries; η^*: $F(\mathcal{M}^*)=0 \bmod q$.

So the probability of $\mathcal{B}$ not aborting is

$$\Pr(not\_abort)=\Pr\left(\bigcap_{i=1}^{q_e}\mathrm{E}_i\wedge\mathrm{E}^*\wedge\bigcap_{j=1}^{q_s}\eta_j\wedge\eta^*\right)$$
$$=\left(1-\frac{q_e}{q}\right)\cdot\frac{1}{q}\cdot\frac{3}{4}\cdot\frac{1}{4\cdot q_s}\cdot\frac{1}{n_m+1}>\left(1-\frac{q_e}{l_m\left(n_m+1\right)}\right)\cdot\frac{1}{q}\cdot\frac{3}{4}\cdot\frac{1}{4\cdot q_s}\cdot\frac{1}{n_m+1}=\frac{12\cdot\left(n_m+1\right)\cdot q_s-3\cdot q_e}{64\cdot q\cdot\left(n_m+1\right)^2\cdot q_s^2}.$$

We can get that $\varepsilon'=\dfrac{\left[12\cdot\left(n_m+1\right)\cdot q_s-3\cdot q_e\right]\cdot\varepsilon}{64\cdot q\cdot\left(n_m+1\right)^2\cdot q_s^2}$. The time complexity of the algoithm $\mathcal{B}$ is $h'=h+O(q_e\cdot(6\cdot C_{mul}+9\cdot C_{mul}+)+q_s\cdot[(4\cdot n_m+4\cdot|U|+9)\cdot C_{mul}+(4\cdot|U|+18\cdot C_{exp})])$.

5.4 Traceability

Theorem 5.2: The scheme of Section 4 has the threshold proxy signature traceability so as to trace t proxy signers participating in signing when it is unforgeable (Theorem 5.1 holds).

Proof*:** The scheme of Section 4 has been proved to have the threshold proxy signature unforgeability in Section 5.3. So, in the algorithm **TTPS.*TProxyTrace, the trusted authority may certainly uncover out the proxy signer p collecting distributed proxy signatures from the equation,

$$e(g_p, pk_p)=\frac{\prod_{j=1}^{n+1}e(g_j, pk_j)\cdot e(\theta, Q_{i,3})^{H\left(pk_i, PK_{\{1,2,\ldots\ldots,n,n+1\}/i}, w_i\right)}}{e\left(Q_{i,1}, g\right)\cdot e\left(Q_{i,2}, \tau'\cdot\prod_{d\in M}\tau_d\right)^{-1}}.$$

Therefore, the proxy signer p cannot deny to have participated in signing. Additionally, the proxy signer p can verify each distributed proxy signature $p\sigma_{i,j}$ by the equation, $\underline{B}_{i,j}=e(Q_{i,j,1}, g)\cdot e(Q_{i,j,2}, \tau'\cdot\prod_{d\in M}\tau_d)^{-1}$ respectively for other $t-1$ proxy signers participated in signing. Thus, other $t-1$ proxy signers participated in signing cannot deny to have participated in signing.

6. CONCLUSIONS

In this paper, we propose a traceable (t, n) threshold proxy signature scheme in the standard model, which is based on the Waters' signature scheme and the Xiong *et al.*'s threshold signature scheme. In our proposed scheme, there is not a trusted third party of computing signature, then computing a threshold proxy signature is more flexible. Also, we show the complete analysis for security of the proposed scheme. Compared with other types of proxy signature schemes, constructing a secure and efficient traceable (t, n) threshold proxy signature scheme is very challenging [1, 9, 11-13, 17]. Thus, the work about threshold proxy signature still needs to be further progressed.

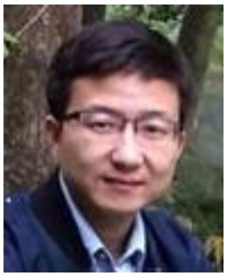
REFERENCES

1. A. Boldyreva, A. Palacio, and B. Warinschi, "Secure proxy signature schemes for delegation of signing rights," *Journal of Cryptology*, Vol. 25, 2012, pp. 57-115.
2. A. Boldyreva, A. Palacio, and B. Warinschi, "Secure proxy signature schemes for delegation of signing rights," http://eprint.iacr.org/2003/096.
3. D. Boneh and X. Boyen, "Short signatures without random oracles," *Advances in Cryptology*, Vol. 3027, 2004, pp. 56-73.
4. F. Cao and Z. F. Cao, "A secure identity-based multi-proxy signature scheme," *Computers and Electrical Engineering*, Vol. 35, 2009, pp. 86-95.
5. J. C. Cha and J. H. Cheon, "An identity-based signature from Gap Diffie-Hellman groups," *Public Key Cryptography*, LNCS, Vol. 2567, 2002, pp. 18-30.
6. F. Cao and Z. Cao, "A secure identity-based proxy multi-signature scheme," *Information Sciences*, Vol. 179, 2009, pp. 292-302.
7. G. Fuchsbauer and D. Pointcheval, "Anonymous consecutive delegation of signing rights: unifying group and proxy signatures," *Lecture Notes in Computer Science*, Vol. 5458, 2009, pp. 95-115.
8. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signature, Advances" in *Proceedings of International Conference on Cryptology-EuroCrypt*, LNCS, Vol. 1070, 1996, pp. 354-371.
9. S. J. Hwang and C. C. Chen, "New threshold-proxy threshold-signature schemes," *Computers and Electrical Engineering*, Vol. 31, 2005, pp. 69-80.
10. J. Herranz and G. Saez, "Revisiting fully distributed proxy signature schemes," http://eprint.iacr.org/2003/197.
11. H. F. Huang and C. C. Chang, "A novel efficient (t, n) threshold proxy signature scheme," *Information Sciences*, Vol. 176, 2006, pp. 1338-1349.

12. J. H. Hu and J. Z. Zhang, "Cryptanalysis and improvement of a threshold proxy signature scheme," *Computer Standards and Interfaces*, Vol. 31, 2009, pp. 169-173.
13. C. L. Hsu and T. S. Wu, "Efficient nonrepudiable threshold proxy signature scheme with known signers against the collusion attack," *Applied Mathematics and Computation*, Vol. 168, 2005, pp. 305-319.
14. Z. P. Jin and Q. Y. Wen, "Certificateless multi-proxy signature," *Computer Communications*, Vol. 34, 2011, pp. 344-352.
15. Y. S. Kim and J. H. Chang, "Provably secure proxy blind signature scheme," in *Proceedings of the 8th IEEE International Symposium on Multimedia*, 2006, pp. 998-1003.
16. Z. H. Liu, Y. P. Hu, X. S. Zhang, and H. Ma, "Provably secure multi-proxy signature scheme with revocation in the standard model," *Computer Communications*, Vol. 34, 2011, pp. 494-501.
17. R. X. Lu, Z. F. Cao, and H. J. Zhu, "A robust (k, n)+1 threshold proxy signature scheme based on factoring," *Applied Mathematics and Computation*, Vol. 166, 2005, pp. 35-45.
18. X. Li and K. Chen, "ID-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes from bilinear pairings," *Applied Mathematics and Computation*, Vol. 169, 2005, pp. 437-450.
19. S. Lal and A. K. Awasthi, "Proxy blind signature scheme," http://eprint.iacr.org/2003/072.
20. M. Mambo, K. Usuda, and E. Okamoto, "Proxy signature: delegation of the power to sign messages," *IEICE Transactions on Fundamentals of Electronics Communication and Computer Science*, Vol. E79-A, 1996, pp. 1338-1354.
21. T. Malkin, S. Obana, and M. Yung, "The hierarchy of key evolving signatures and a characterization of proxy signatures," *Advances in Cryptology-EuroCrypt*, Vol. 3027, 2004, pp. 306-322.
22. K. G. Paterson and J. C. N. Schuldt, "Efficient identity-based signatures secure in the standard model," in *Proceedings of the 11th Australasian Conference on Information Security and Privacy*, LNCS, Vol. 4058, 2006, pp. 207-222.
23. J. C. N. Schuldt, K. Matsuura, and K. G. Paterson, "Proxy signatures secure against proxy key exposure," *Public Key Cryptography*, Vol. 4939, 2008, pp. 141-161.
24. Y. Sun, C. X. Xu, and Y. Yu, "Strongly unforgeable proxy signature scheme secure in the standard model," *Journal of Systems and Software*, Vol. 84, 2011, pp. 1471-1479.
25. Y. Sun, C. X. Xu, H. Wang, and C. X. Fu, "Improved multi-proxy signature scheme without random oracles," *Chinese Journal of Electronics*, Vol. 20, 2011, pp. 200-206.
26. Y. Sun, C. X. Xu, Y. Yu, and B. Yang, "Improvement of a proxy multi-signature scheme without random oracles," *Computer Communications*, Vol. 34, 2011, pp. 257-263.
27. Z. H. Shao, "Improvement of identity-based proxy multi-signature scheme," *Journal of Systems and Software*, Vol. 82, 2009, pp. 794-800.
28. A. Shamir, "How to share a secret," *Communications of the ACM*, Vol. 22, 1979, pp. 612-613.
29. Q. Wang, Z. F. Cao, and S. B. Wang, "Formalized security mModel of multi-proxy signature schemes," in *Proceedings of the 5th International Conference on Computer and Information Technology*, 2005, pp. 668-672.

30. B. Waters, "Efficient identity-based encryption without random oracles," *Advances in Cryptology-EuroCrypt*, Vol. 3494, 2005, pp. 114-127.
31. H. Xiong, J. B. Hu, Z. Chen, and F. G. Li, "On the security of an identity based multi-proxy signature scheme," *Computers and Electrical Engineering*, Vol. 37, 2011, pp. 129-135.
32. H. Xiong, F. G. Li, and Z. G. Qin, "Certificateless threshold signature secure in the standard model," *Information Sciences*, Vol. 237, 2013, pp. 73-81.
33. C. H. Yang, S. F. Tzeng, and M. S. Hwang, "On the efficiency of non-repudiable threshold proxy signature scheme with known signers," *Journal of Systems and Software*, Vol. 73, 2004, pp. 507-514.
34. T. Yang, H. Xiong, J. B. Hu, Y. G. Wang, Y. Deng, B. Xiao, and Z. Chen, "A traceable certificateless threshold proxy signature scheme from bilinear pairings," *Web Technologies and Applications*, LNCS, Vol. 6612, 2011, pp. 376-381.
35. L. Harn and F. Wang, "Threshold signature scheme without using polynomial interpolation," *International Journal of Network Security*, Vol. 18, 2016, pp. 710-717.
36. S. J. Aboud, S. Yousef, and M. Cole, "Undeniable threshold proxy signature scheme," https://www.researchgate.net/publication/261242449.
37. S. Mashhadi, "A novel non-repudiable threshold proxy signature scheme with known signers," *International Journal of Network Security*, Vol. 15, 2013, pp. 274-279.
38. G. Hu, X. Yang, and C. Wang, "An unidirectional threshold proxy resignature scheme," *Journal of Convergence Information Technology*, Vol. 8, 2013, pp. 731-736.
39. C. C. Lee, T. C. Lin, S. F. Tzeng, and M. S. Hwang, "Generalization of proxy signature based on factorization," *International Journal of Innovative Computing, Information and Control*, Vol. 7, 2011, pp. 1039-1054.
40. C. I. T. Chen, M. H. Chang, and Y. S. Yeh, "Design of proxy signature in the digital signature algorithm (DSA)," *Journal of Information Science and Engineering*, Vol. 22, 2006, pp. 965-973.
41. C. C. Lee, M. S. Hwang, and W. P. Yang, "Untraceable blind signature schemes based on discrete logarithm problem," *Fundamenta Informaticae*, Vol. 55, 2003, pp. 307-320.
42. N. Y. Lee, T. Hwang, and C. M. Li, "(t, n) threshold untraceable signatures," *Journal of Information Science and Engineering*, Vol. 16, 2000, pp. 835-846.

Ke Gu (谷科) received his Ph.D. degree in School of Information Science and Engineering from Central South University in 2012. He is currently a Lecturer at Changsha University of Science and Technology. His research interests include cryptography, network and information security.



Yong Wang (王勇) received his Ph.D. degree in School of Information Science and Engineering from Central South University in 2011. He is currently an Associate Professor at Central South University. His research interests include information theory and computational intelligence.

Sheng Wen (文晟) received his Ph.D. degree in School of Information Science and Engineering from Central South University in 2012. He is currently a Lecturer at Deakin University. Her research interests include social media, network security and modelling.