# An Anonymous but Accountable Proxy Multi-signature Scheme

He Du, Jian Wang
Nanjing University of Aeronautics and Astronautics / College of Computer Science and Technology, Nanjing, P.R. China
Email: {duhe, wangjian}@nuaa.edu.cn

***Abstract*—in a proxy multi-signature, the proxy signer can sign a message on behalf of more than one original signer. However, in some cases, the proxy or the original signers do not want to leak the proxy signer's identity. In addition, when something is wrong with the signed message, the identity of the proxy signer has to be revealed. So the problem of anonymous but accountable has been proposed. Using the technology of pseudonym and secret sharing, in this paper, a simple method was proposed to reach the target of anonymous but accountable. In the scheme, the identity of the proxy signer was anonymous, but when t or more original signers want to reveal the proxy signer's identity, the proxy signer's identity would be revealed and be demonstrated.**

***Index Terms*—anonymous but accountable, proxy multi-signature, secret sharing**

## I. INTRODUCTION

As an important part of cryptography, digital signature draws lots of attention [1-3]. In 2000, Yi et al. proposed two proxy multi-signature schemes [4]. In a proxy multi-signature, the proxy signer can sign a message on behalf of more than one original signer. Based on Yi et al.'s scheme, in 2001, Lee et al. propose a new proxy multi-signature scheme with strong property [5]. Then, lots of proxy multi-signatures were proposed [6-11].

In order to verify the proxy multi-signature, the verifier needs get the proxy and all the original signers' public keys. Obviously, the verification procedure would expose the identity of the proxy signer. In 2006, Han et al proposed a scheme to solve the anonymous but accountable problem of a proxy signer [12]. But in the scheme, they use a trust third party to reveal the real proxy signer. In 2012, Du et al. proposed a method to reach the target of anonymous but accountable property [13]. But, when it comes to the proxy multi-signature scheme, the method was not appropriately.

In this paper, using the technology of pseudonym and secret sharing, a simple method was proposed to reach the target of anonymous but accountable in a proxy multi-signature scheme. In the scheme, the identity of the proxy signer was anonymous, but when t or more original signers want to reveal the proxy signer's identity, the proxy signer's identity would be revealed and be demonstrated. Compared with the protocol of Du et al., in our scheme, the original signers need not to renew their secret keys after revealing the identity of the proxy signer.

The rest of the paper is organized as follows. Section 2 gives some preliminaries employed later in this paper. Section 3 proposes the scheme of the special proxy multi-signature. We analysis the security of the proposed scheme in section 4 and finally conclude in Section 5.

## II. PRELIMINARIES

Because bilinear maps and the secret sharing play the key roles in our scheme, we would describe basic properties of them, and then also list some related mathematical hard problems.

### A. Bilinear Pairings

Let $G_1$ be a cyclic additive group and $G_2$ be a cyclic multiplicative group of the same large prime order $q$. We also assume that the discrete logarithm problems ($DLP$) in both $G_1$ and $G_2$ are hard to solve.

A map $\hat{e}: G_1 \times G_1 \to G_2$ is an admissible bilinear map [14] [15] if it satisfies the three following properties:

1. Computable: $\forall P, Q \in G_1$, there exists an efficient algorithm to compute $\hat{e}(P,Q)$;

2. Bilinear: $\forall P, Q \in G_1$ and $\forall a, b \in Z_q^*$, we have $\forall P, Q \in G_1, \hat{e}(aP, bQ) = \hat{e}(P,Q)^{ab}$;

3. Non-degenerate: for any point $P \in G_1$, $\hat{e}(P,Q) = 1$ for all $Q \in G_1$ if $P = O$.

Related mathematical hard problems

Here, we list some related mathematical hard problems in $G_1$ that we have used later:

1. Discrete Logarithm Problem ($DLP$): Given two group elements $P$ and $Q$, to find an integer $a$, such that $Q = aP$ whenever such an integer exists.

2. Decisional Diffie-Hellman Problem ($DDHP$): For $a, b, c \in Z_q^*$, given $P, aP, bP, cP$, to decide whether $c = ab \bmod q$.

3. Computational Diffie-Hellman Problem ($CDHP$): for $a, b \in Z_q^*$, given $P, aP, bP$, compute $abP$.

When the $DDHP$ is easy but the $CDHP$ is hard on the group $G_1$, we call $G_1$ as Gap Diffie-Hellman ($GDH$) group. Such groups can be found on super singular


doi:10.4304/jsw.8.8.1867-1874

elliptic curves or hyper elliptic curves over finite field,

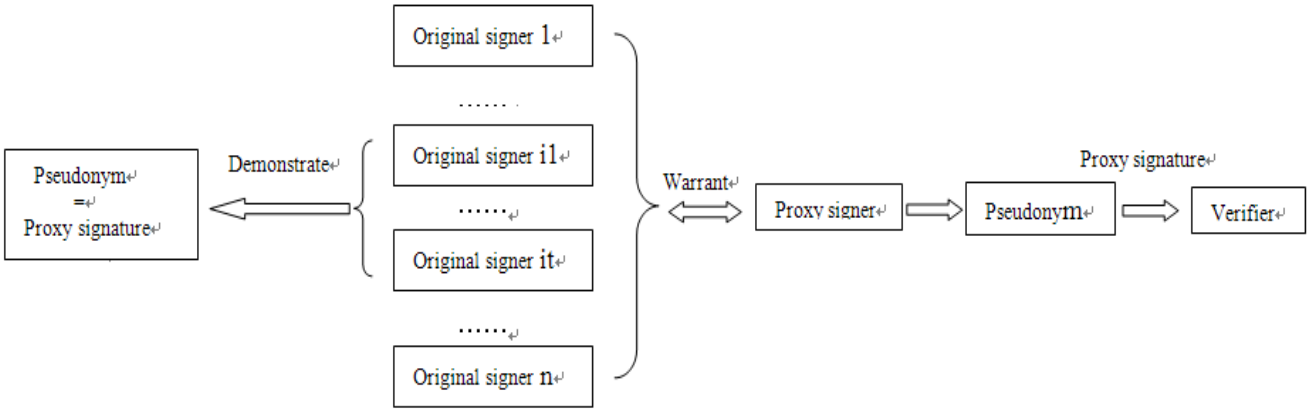


Figure 1. Brief description of the scheme

and the bilinear parings can be derived from the Weil or Tate pairing.

### B. Secret Sharing

Share generation algorithm for generic shareholder

The dealer D first picks a polynomial $f(x)$ randomly: $f(x)=a_0+a_1x+a_2x^2+\mathrm{L}\ +a_{t-1}x^{t-1}$. All coefficients $a_0,a_1,\mathrm{L}\ ,a_{t-1}$ are in a finite field F and $a_0$ is made as the secret s. Then D computes:

$$s_1=f(x_1),s_2=f(x_2),\mathrm{K}\ ,s_n=f(x_n) \qquad (1)$$

$x_i(1\le i\le n)$ is the shareholder's ID.

At last, the algorithm outputs a list of n shares $(s_1,s_2,\mathrm{K}\ ,s_n)$. Each shareholder $P_i$ is distributed a share $s_i$ secretly, $1\le i\le n$.

Secret reconstruction algorithm

In Shamir's (t, n)-secret sharing scheme, the reconstruction process can be described as follow:

$$f(x_i)=y_i\Rightarrow\begin{pmatrix}1 & x_1 & x_1^2 & \mathrm{L} & x_1^{t-1}\\ 1 & x_2 & x_2^2 & \mathrm{L} & x_2^{t-1}\\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M}\\ 1 & x_t & x_t^2 & \mathrm{L} & x_t^{t-1}\end{pmatrix}\begin{pmatrix}a_0\\ a_1\\ \mathrm{M}\\ a_{t-1}\end{pmatrix}=\begin{pmatrix}y_1\\ y_2\\ \mathrm{M}\\ y_t\end{pmatrix}\Rightarrow a_0=s \qquad (2)$$

$x_i$ is the shareholder's ID, $y_i$ is the piece, s is the secret.

## III. The Scheme

In an anonymous but accountable proxy multi-signature scheme, the original signers use their standard signature algorithm to sign a warrant which includes the type of the information delegated, the original signer's identity, proxy signer' s pseudonym (which is correlated to proxy signer's real identity/public key) and the period of delegation, etc. As a result of the interaction between the proxy and original signers, the proxy signer generates a proxy private key. After that, the proxy signer can sign any messages according to the warrant. When necessary, (t or more) original signers can join together to reveal or demonstrate the real identity/public key of the proxy signer. The procedure can be seen as FIGURE 1.

### A. Definition

In an anonymous but accountable proxy multi-signature scheme, there are n original signers $O_1,\mathrm{K}\ ,O_n$ and a proxy signer $P$.

**Definition 1.** An anonymous but accountable proxy multi-signature scheme is a tuple AAPMS=(Setup, Sign, Veri, PMSGen, PMSSign, PMSVeri, Reveal, Demonstration).

- Setup is used to produce private/public key pairs as usual. The public and private key pairs for the original and the proxy signers are $(pk_{O1},sk_{O1}),\mathrm{K}\ ,(pk_{On},sk_{On}),(pk_P,sk_P)$.
- Sign is a probabilistic polynomial-time signature issuing algorithm, which takes input message m, signer's private key $sk$, outputs a signature δ on message m.
- Veri is a deterministic verification algorithm. On input signer's public key, message m and a candidate signature δ for m, the algorithm outputs 1 if δ is a valid signature on m for the entity, and outputs 0 otherwise.
- PMSGen is a protocol used between the proxy signer and the original signers. The original signers and proxy signer take as input their private keys $sk_{O1},\mathrm{K}\ ,sk_{On}$ and $sk_P$. As a result of the interaction, the proxy signer outputs a proxy signing key $sk_{pro}$ which the proxy signer uses to produce an anonymous but accountable proxy multi-signature on behalf of the original signers.
- PMSSign is a (possibly) randomized algorithm. It takes as input the proxy signing key $sk_{pro}$ and the message $m\in\{0,1\}^*$, and then outputs an anonymous but accountable proxy multi-signature on the message m on behalf of the original signers.
- PMSVeri is a deterministic algorithm. It takes input the public keys $Q_{pseu}$ and $Q_{O1},\mathrm{K}\ ,Q_{On}$, $Q_{pseu}$ is the proxy signer's pseudonym, $Q_{O1},\mathrm{K}\ ,Q_{On}$ are the original signers' public keys. The message m and a candidate anonymous but accountable proxy multi-

signature $Sig_{pms}$ for m are also takes as input. The algorithm outputs 1 if $Sig_{pms}$ is valid for m by the proxy signer's pseudonym on behalf of the designated original signers, and outputs 0 otherwise.

- Reveal is used to reveal the real identity/public key of proxy signer when necessary.
- Demonstration is used to demonstrate the real identity/public key of proxy signer when necessary.

*B. Security Model*

Adversaries can be classified into seven types:

***Type I*** This type of adversary only has the public keys of the original signers and proxy signer. He aims to forge a signature of a warrant w of the original signer or forge a proxy signature of a message m with respect to the original signer and the proxy signer.

***Type II*** This type of adversary has not only the public keys of the original signers and proxy signer but also the private key of proxy signer. He aims to forge a delegation signature of a warrant w that he chooses from the original signers. Note that once he can get the signature of a warrant w, he can forge any proxy signature on any message.

***Type III*** This type of adversary has not only the public keys of the original signer and proxy signer but also the private keys of some original signers. He aims to forge a valid proxy signature.

From above we can find that if an AAPMS scheme is unforgeable against Type II and Type III adversaries, it is also unforgeable against Type I adversary.

***Type IV*** This type of adversary has only the public keys of the original signer. He aims to get the true identity of the proxy signer from the delegation file or the proxy signature.

***Type V*** This type of adversary aims to convince others that the proxy signer has made a proxy signature on behalf of the original signer(s) who has not delegated to the proxy signer.

***Type VI*** This type of adversary only has the public key of the original signers. He aims to malign an irrespective entity as the corresponding proxy signer.

***Type VII*** This type of adversary not only has the public key of the original signers but also the proxy signer.

From above we can find that if an AAPMS scheme is unforgeable against Type VII adversary, it is also unforgeable against Type VI adversary.

**Definition 2.** We define the AAPMS scheme's anonymous but accountable property as follows.

Let $o_1,\mathrm{K},o_n$ be the original signer, $p$ be the proxy signer and $v$ be the verifier, $D$ be the dealer.

**Setup** PKG runs the algorithm to obtain the secret key and public key pairs representing the keys of the proxy signer and original signers, respectively.

**Pre-delegation** Proxy signer sends the variation of random number with corresponding signature to the original signer.

**Delegation signature generation** The original signer computes delegation key and chooses the warrant $w$. Then he sends them to proxy signer with the corresponding signatures. $w$ includes the $Q_{pseu}$, $Q_{pseu}$ is dependent on a data which can be got from the interaction between the original signers and the dealer.

**Anonymous but accountable signature generation** $p$ chooses a message m, and outputs an anonymous but accountable proxy multi-signature on m.

**Verification** Finally, $v$ uses the original signer's public keys and proxy signer's pseudonym to check the anonymous but accountable signature.

**Reveal/demonstrate** if the original signers needed, they can join together to provide the corresponding evidence to reveal who is the real proxy signer.

A proxy multi-signature scheme is said to has the property of anonymous but accountable if no one can found the real proxy signer before the t or more original signers provide evidence to reveal who is the real proxy signer.

*C. Our Scheme*

Our scheme is based on Cao and Cao's identity-based proxy multi-signature scheme [8]. Let $O_1,\mathrm{K},O_n$ be the original signers and $P$ be the proxy signer designated by $O_1,\mathrm{K},O_n$. For $i\in\{1,\mathrm{K},n\}$, $O_i$ has an identity $ID_i$, $P$ has an identity $ID_P$.

**Setup:** Assume k is a security parameter. $G_1$ is a cyclic additive group of prime order $q$, with a generator $P$, $G_2$ is a cyclic multiplicative group of the same order $q$, and $e:G_1\times G_1\to G_2$ is a bilinear map. PKG chooses a master-key $s\in Z_q^*$ and computes $P_{pub}=sP$. Chooses hash functions $H_1,H_2,H_3:\{0,1\}^*\to G_1$, and hash function $H_4:\{0,1\}^*\to Z_q^*$.

**Extract:** Given a user's identity $ID\in\{0,1\}^*$, computes $Q_{ID}=H_1(ID)\in G_1$ and the associated private key $d_{ID}=sQ_{ID}\in G_1$.

**Sign:** Given a private key $d_{ID}$, in order to sign a message $m\in\{0,1\}^*$,Randomly picks $r\in Z_q^*$, and computes $U=rP\in G_1$, then computes $H=H_2(m\|U)\in G_1$. Computes $V=d_{ID}+rH\in G_1$. The signature on $m$ is $\sigma=<U,V>$.

**Veri:** To verify a signature $\sigma=<U,V>$ on message $m$ for an identity $ID$, the verifier first computes $Q_{ID}=H_1(ID)$, and $H'=H_2(m\|U)$. Then accepts the signature if $e(P,V)=e(P_{pub},Q_{ID})e(U,H')$ and rejects it otherwise.

**PMSGen:**

(1) Delegation generation:

To delegate the signing capability to $P$, the

original signers and the proxy signer do the following to make the signed warrant w. The warrant w specifies the necessary proxy details, such as the identity information of the original signers and the pseudonym of proxy signer, the type of the information delegated, and the period of delegation.

The proxy signer selects n random number $r_{pi} \in Z_q^*$ and computes $R_{pi} = r_{pi}P$ , $i \in \{1,\mathrm{K},n\}$ . Then he sends a selected $R_{pi}$ and the corresponding signature to the original signers.

Each original signer $ID_i$ selects a random number $r_{oi} \in Z_q^*$ and sends it to the dealer $D$ in a secure channel.

$D$ computes $r_o = r_{o1} + r_{o2} + \mathrm{K} + r_{on} \in Z_q^*$ , $R_O = r_O P$ and sends $R_O$ to original signers, sends $r_O$ to the proxy signer in a secure channel . Then $D$ selects a polynomial $f(x) = r_o + a_1 x + a_2 x^2 + \mathrm{K} + a_{t-1}x^{t-1}$ , $t$ is designated number. Then, $D$ computes $f(ID_i), i \in \{1,\mathrm{K},n\}$ and sends them to the corresponding original signer in a secure channel.

Each original signer computes $Q_{pseu} = R_O + Q_P + R_{Pi}$ as the proxy signer's pseudonym.

At last, the warrant used to AAPMS is $w = (Q_{ID1},\mathrm{K},Q_{IDn},Q_{pseu},scope,etc)$

-For $1 \le i \le n$ , $O_i$ randomly chooses $r_i \in Z_q^*$ , and computes $U_i = r_i P \in G_1$ , broadcasts $U_i$ to the other $n-1$ original signers.

- For $1 \le i \le n$ , $O_i$ computes $U = \sum_{i=1}^{n} U_i \in G_1$ , $V_i = d_i + r_i H$ , $H = H_2(w \| U) \in G_1$ , $w$ is the warrant.

- For $1 \le i \le n$ , $O_i$ sends $(w,U_i,V_i)$ to the proxy signer $P$ .

(2) Delegation verification:

After receiving all $(w,U_i,V_i)$ , $i \in \{1,\mathrm{K},n\}$ , $P$ computes $U' = \sum_{i=1}^{n} U_i, (i = 1,\mathrm{K},n)$ , then checks if $e(P,V_i) = e(P_{pub},Q_i)e(U_i,H')$ , $H' = H_2(w \| U) \in G_1$ . for $1 \le i \le n$ , where $Q_i = H(ID_i)$ . If $(w,U_i,V_i)$ is valid, he accepts it as a valid delegation and continues; otherwise, he requests a valid one from $O_i$ , or terminates the protocol.

(3) Proxy secret key generation: If $P$ accepts all the delegations $(w,U_i,V_i)$ $(1 \le i \le n)$ , he computes the proxy key as $sk_p = \sum_{i=1}^{n} V_i + H_4(Q_{pseu} \| w \| U)(d_{IDP} + r_O + r_{pi})$ .

**PMSSign:** Using $sk_p$ , $P$ can sign the message $m$ under w on behalf of the original signers $O_1,\mathrm{K},O_n$ as follows:

-Randomly picks $r_p \in Z_q^*$ , and computes $U_p = r_p P, H_p = H_3(Q_{pseu} \| w \| m \| U_p)$ .

-Computes $V_P = sk_p + r_p H_p$ .

The proxy signature for message m on behalf of the original signers $O_1,\mathrm{K},O_n$ is $\sigma = (w,U_p,V_p,U)$ .

**PMSVeri:** To verify a proxy multi-signature $\sigma = (w,U_p,V_p,U)$ for message m under a warrant w, the verifier operates as follows:

-Checks whether or not the message m conforms to the warrant w. If not, stop. Otherwise, continue.

- Checks whether or not the proxy signer $P$ is authorized by the n original signers $O_1,\mathrm{K},O_n$ in the warrant w. If not, stop. Otherwise, continue.

- Computes $H_p' = H_3(Q_{pseu} \| w \| m \| U_p), H' = H_2(w \| U)$ , then checks if

$$e(P,V_p) = e(P_{pub}, \sum_{i=1}^{n} Q_i + H_4(Q_{pseu} \| w \| U)Q_{pseu})e(U,H')e(U_p,H_p')$$

If this holds, then accepts it, otherwise, rejects it.

**Reveal:** When the proxy signer's public key need to be revealed, t or more original signers can use the shares got from D to recover the shared secret $r_0$ (method can be found from the part of **Preliminaries**) , computes $R_O = r_O P$ . Then they can use $R_O$ and $R_{pi}$ to reveal the identity of the proxy signer $Q_{pseu} = R_O + Q_p + R_{pi}$ .

**Demonstration:** If the original signers have to demonstrate the identity of the proxy signer, t or more original signers can provide their shares to reconstruct the secret $r_O$ . Then $R_O$ is demonstrated by $R_O = r_O P$ . $R_{pi}$ can be demonstrated by the proxy signer's corresponding standard signature. So, $Q_{pseu} = R_O + Q_p + R_{pi}$ is demonstrated.

## IV. Analysis of Our Scheme

**Theorem 1:** The above proxy multi-signature is existentially unforgeable against adaptively chosen-message attacks and adaptively chosen-warrant attacks in the random oracle model if the Computational Diffie-Hellman Problem in GDH groups is hard to solve.

**Proof.** Our scheme is based on Cao and Cao's proxy multi-signature scheme [8], which is provably secure in the random oracle model assuming the Computational Diffie-Hellman problem is hard in gap Diffie-Hellman groups. We just omit it.

**Theorem 2.** Our scheme can stand against Type IV adversary.

From the scheme above, we can find that our scheme reaches the target of anonymous but accountable.

**Proof.** We suppose that the adversary would get the identity of the proxy signer from the proxy signature. Then he must has got $Q_{pseum} = R_O + Q_P + R_{Pi}$ . But, it is impossible, because the adversary can not get $R_O$ from the original signer or the dealer. He cannot demonstrate it.

**Theorem 3.** Our scheme can stand against TypeV adversary.

**Proof.** We suppose that the adversary want to convince others that the proxy signer has made a proxy signature on behalf of the original signers who have not delegated to the proxy signer. Because the scheme is provably secure in the random oracle model assuming the Computational Diffie-Hellman problem is hard in gap

Diffie-Hellman groups, the adversary can not forge the corresponding warrant w, which has been proved in Cao and Cao's scheme.

**Theorem 4.** Our scheme can stand against TypeVII adversary.

**Proof.** We suppose the adversary want to malign Q . as the proxy signer. In order to demonstrate Q. were the proxy signer, the adversary has to output a number $r_O$, where $R_O = r_O P$. But it is impossible except he has solved the Discrete Logarithm Problem. Or the adversary has got more than t secret keys of original signers.

Because that if an AAPMS scheme is unforgeable against Type VII adversary, it is also unforgeable against Type VI adversary, so we just omit the proof.

We compared our scheme with Han et al and Du et al's scheme as bellow.

TABLE I.

COMPARISON

| Property | Han et al's scheme | Du et al's scheme | Our scheme |
|---|---|---|---|
| TTP | Need | Not need | Not need |
| Security proof | No | Yes | Yes |
| Change the key | Not need | Need | Not need |

## V. CONCLUSION

Privacy has become one of the most important human rights of the modern age. Using the technology of pseudonym, in this paper, we propose a simple secure anonymous but accountable method for proxy signer in a proxy signature scheme. Unless the corresponding evidence has been provided by the original signer(s), no one can find the proxy signer's real identity or public key. In the scheme, the identity of the proxy signer was anonymous, but when t or more original signers want to reveal the proxy signer's identity, the proxy signer's identity would be revealed and be demonstrated.

## APPENDIX A SECURITY PROOF

If there exists a type *II* adversary $A_{II}$ can breaks the standard signature scheme, then there exists an algorithm $\mathcal{B}$ which is able to use $A_{II}$ to solve an instance of the CDH problem with a non-negligible probability.. Short signature is proved to be secure against existential forgery on adaptive chosen-message attacks (in the random oracle model) assuming the CDHP (Computational Diffie-Hellman Problem) is hard, we just omit the proof.

If there exists a type *III* adversary $A_{III}$ can breaks the proxy multi-signature scheme, then there exists an algorithm $\mathcal{B}$ which is able to use $A_{III}$ to solve an instance of the *CDH* problem with a non-negligible probability.

**Proof.** Algorithm $\mathcal{C}$ is given $X = xP \in G_1$ and $Y = yP \in G_1$. Its goal is to output $xY = xyP \in G_1$. Algorithm $\mathcal{C}$ simulates the challenger and interacts with adversary $\mathcal{A}$ as follows:

Setup: Algorithm $\mathcal{C}$ initializes $\mathcal{A}$ with $P_{pub} = X$ as a systems overall public key.

extraction-queries: At any time $\mathcal{A}$ can query the random oracle $PK$. To respond to these queries, $\mathcal{C}$ maintains a list $L_1$ of tuples $(i, Q_i, b_i, c_i)$ as explained below. The list is initially empty. When an identity $i$ is submitted to the $PK$ oracle, algorithm $\mathcal{C}$ responds as follows:

(1) If the query $Q_i$ already appears on the list $L_1$ in some tuple $(i, Q_i, b_i, c_i)$, then algorithm $\mathcal{C}$ responds with $Q_i$.

(2) Otherwise, $\mathcal{C}$ generates a random coin $c \in \{0,1\}$ such that $\Pr[c = 0] = 1/(q_E + n)$.

(3) Algorithm $\mathcal{C}$ picks a random $b_i \in Z_q^*$. If $c = 0$, computes $Q_i = b_i Y$, If $c = 1$, computes $Q_i = b_i P$.

(4) Algorithm $\mathcal{C}$ adds the tuple $(i, Q_i, b_i, c_i)$ to the list $L_1$ and responds to $\mathcal{A}$ with $Q_i$.

$H_2$ -queries: At any time $\mathcal{A}$ can query the random oracle $H_2$. To respond this query, $\mathcal{C}$ maintains a list $L_2$ of tuples $(e_i, m_i, v_i, s_i)$, as explained below. The list is initially empty. When a message $m_i$ is submitted to the $H_2$ oracle, algorithm $\mathcal{C}$ responds as follows:

(1) If the query $m_i$ already appears on the list $L_2$ in some tuple $(e_i, m_i, v_i, s_i)$ then algorithm $\mathcal{C}$ responds with $H_2(m_i) = e_i$.

(2) Otherwise, $\mathcal{C}$ generates a random coin $s_i \in \{0,1\}$ such that $\Pr[s_i = 0] = 1/2$.

(3) Algorithm $\mathcal{C}$ picks a random $v_i \in Z_q^*$. If $s_i = 0$, computes $e_i = v_i P_{pub}$, If $s_i = 1$, computes $e_i = v_i P$.

(4) Algorithm $\mathcal{C}$ adds the tuple $(e_i, m_i, v_i, s_i)$ to the list $L_2$ and responds to $\mathcal{A}$ with $H_2(m_i) = e_i$.

$H_3$ -queries: At any time $\mathcal{A}$ can query the random oracle $H_3$. To respond to queries to $H_3$ oracle, $\mathcal{C}$ maintains a list $L_3$ of tuples $(Q_i, w_i, m_i, R_i, \eta_i)$ as explained below. When a tuple $(Q_i, w_i, m_{i,i})$ is submitted to the $H_3$ oracle, algorithm $\mathcal{C}$ responds as follows:

(1) If the query tuple already appears on the list $L_3$ in some tuple $(Q_i, w_i, m_i, R_i, \eta_i)$, then algorithm $\mathcal{C}$ responds with $H_3(Q_i \| w_i \| m_i \| R_i) = \eta_i P \in G_1$.

(2) Otherwise, algorithm $\mathcal{C}$ picks $\eta_i \in Z_q^*$ at random, stores the tuple $(Q_i, w_i, m_i, R_i, \eta_i)$ in the list $L_3$ and responds to $\mathcal{A}$ with $H_3(Q_i \| w_i \| m_i \| R_i) = \eta_i P \in G_1$.

$H_4$ -queries: At any time $\mathcal{A}$ can query the random oracle $H_4$. To respond to queries to $H_4$ oracle, $\mathcal{C}$ maintains a list $L_4$ of tuples $(Q_i, w_i, u_i)$ as explained below. When a tuple $(Q_i, w_i)$ is submitted to the $H_4$ oracle, algorithm $\mathcal{C}$ responds as follows:

(1) If the query tuple already appears on the list $L_4$ in some tuple $(Q_i, w_i, u_i)$, then algorithm $\mathcal{C}$ responds with $H_4(Q_i \| w_i) = u_i \in Z_q^*$.

(2) Otherwise, algorithm $\mathcal{C}$ picks $u_i \in Z_q^*$ at random, stores the tuple $(Q_i, w_i, u_i)$ in the list $L_4$ and responds to $\mathcal{A}$ with $H_4(Q_i \| w_i) = u_i \in Z_q^*$.

Delegation signature queries:

If $\mathcal{A}$ requests to interact with one of the original signers (i.e., $Q_i$), $\mathcal{A}$ creates a warrant *w*, and requests $Q_i$ to sign the warrant *w*. $\mathcal{C}$ queries *w* to its signing oracle $O_s(d_i, \cdot)$. Upon receiving an answer $\delta$, it forwards $(w, \delta)$ to $\mathcal{A}$ and adds the warrant w to *Warro*.

If $\mathcal{A}$ requests to interact with $Q_p$, $Q_p$ playing the role of the proxy signer, the original signers are $Q_i$, $(i = 1, \mathrm{K}, n)$. $\mathcal{A}$ outputs warrant *w* and computes the signature $\delta_i = s_i$ for warrant *w* under secret key $d_i$, $(i = 1, \mathrm{K}, n)$. Then he sends $(w, \delta_1, \mathrm{K}, \delta_n)$ to $\mathcal{C}$. After receiving $(w, \delta_1, \mathrm{K}, \delta_n)$, $\mathcal{C}$ checks the validity of $\delta_i = s_i$, $(i = 1, \mathrm{K}, n)$. If so, $\mathcal{C}$ adds the warrant *w* $(i = 1, \mathrm{K}, n)$ to list *Warrp*.

Proxy multi-signature queries: $\mathcal{A}$ requests a proxy multi-signature on $(w, m)$, where $Q_p$ is the proxy signer. $\mathcal{C}$ responds this query as follows:

(1) If *w* is not in the list *Warrp*, $\mathcal{C}$ outputs "failure" and halts. Otherwise, $\mathcal{C}$ has $\delta_i = s_i$ $(i = 1, \mathrm{K}, n)$, where $\delta_i$ is the signature on warrant *w* under the secret key $d_i$ $(i = 1, \mathrm{K}, n)$.

(2) $\mathcal{C}$ recovers the previously defined value $PK(p) = Q_p \in G_1$ and $PK(i) = Q_i$ $(i = 1, \mathrm{K}, n)$ from the list $L_1$.

(3) Makes query *w* to $H_2$ oracle. $\mathcal{C}$ recovers the corresponding $(w_i, v, s)$ from $L_2$. If $s = 1$, then outputs ''failure" and halts. Otherwise, it means that $H_2(w)$ was previously defined to be $vP_{pub}$.

(4) Makes query $(Q_p, w)$ to $H_4$ oracle and return $H_4(ID_p \| w) = u$.

(5) Picks $r_1, r_2 \in Z_q^*$ at random and defines $s_p = r_1 P_{pub} \in G_1$, $R_p = r_2 P_{pub} \in G_1$.

(6) Defines the hash value $H_3(Q_i \| w_i \| m_i)$ as $r_2^{-1}(r_1 P - P\prod_{i=1}^{n} Q_i - uQ_p) \in G_1$ ($\mathcal{C}$ outputs "failure" and halts if $H_3$ turns out to be already defined for the input $(Q_i, w_i, m_i)$).

(7) $\delta = (w, Q_p, R_p, s_p)$ is a valid proxy signature on message *m* by $Q_p$ on behalf of $Q_i$ $(i = 1, \mathrm{K}, n)$ under the warrant *w*.

(8) $\mathcal{C}$ adds $(w, m)$ to the list $IVPMS_{qu}$, and forwards $\delta$ to $\mathcal{A}$.

$E_1$ : $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}$'s key extraction queries.

$E_2$ : $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}$'s standard signature queries.

$E_3$ : $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}$'s proxy multi-signature queries.

$E_6$ : $\mathcal{A}$ generates a valid and nontrivial proxy multi-signature forgery, where $Q_p$ is the proxy signer.

$E_7$ : Event $E_6$ occurs, and, in addition, $c_p = 0$, $c_i = 1$ $(i = 1, \mathrm{K}, n)$ and $s^* = 1$. Here $c_p$ and $c_i$ are the c-component of the tuple on the list $L_1$. $s^*$ is the s-component of the tuple on the list $L_2$.

$E_8$ : A generates a valid and nontrivial proxy multi-signature forgery and $Q_k$ plays the role of one of the original signers.

$E_9$ : Event $E_8$ occurs, and, in addition, $c_k = 0$, $c_i = 1$ $(i = 1, \mathrm{K}, k-1, k+1, \mathrm{K}, n)$, $c_p = 1$ and $s^* = 1$. Here $c_p$ and $c_i$ $(i = 1, \mathrm{K}, n)$ are the c-component of the tuple on the list $L_1$. $s^*$ is the s-component of the tuple on the list $L_2$.

**Output**: Eventually, $\mathcal{A}$ halts. It either concedes failure, in which case so does $\mathcal{C}$, or it returns a forgery.

(1)If $\mathcal{A}$ outputs a forgery of the form $(w^*, M^*, R_p^*, s_p^*)$, where $Q_p$ is the proxy signer, $(w^*, M^*) \notin PMS_{qu}$, and

$$e(s_p^*, P) = e(Q_p, \prod_{i=1}^{n} H_2(w^*) \cdot P_{pub}) e(Q_p, H_4(Q_p \| w^*) \cdot P_{pub}) e(R_p^*, H_3(Q_p \| w^* \| M^* \| R_p^*))$$

.(i.e., $\mathcal{A}$ forges a valid proxy multi-signature on message $M^*$ under the warrant $w^*$). The $\mathcal{C}$ recovers $(p, Q_p, b_p, c_p)$ and $(i, Q_i, b_i, c_i)$ $(i = 1, \mathrm{K}, n)$ from the list $L_1$. Algorithm $\mathcal{C}$ proceeds only if $c_p = 0$ and $c_i = 1$ $(i = 1, \mathrm{K}, n)$. Otherwise, $\mathcal{C}$ outputs "failure" and stop. Then we know that $Q_p = b_p Y$ and $Q_i = b_i P$, $(i = 1, \mathrm{K}, n)$. Next, algorithm $\mathcal{C}$ recovers $(e^*, M^*, v^*, s^*)$ from $L_2$, $(Q_p, R^*, w^*, \eta^*)$ from the list $L_3$ $(H_3(Q_p \| R^* \| w^*)) = \eta^* P$ and $(Q_i, w, u)$ from list $L_4$ $(H_4(Q_i \| w^*) = u^*)$. If $s^* = 0$, then $\mathcal{C}$ outputs "failure" and stop. Otherwise, we know $s^* = 1$, and hence $H_2(M^*)$ was defined to be $v^* P \in G_1$. Then we can deduce

$$e(s_p^*, P) = e(P\prod_{i=1}^{n} b_i P, \prod_{i=1}^{n} (v^* P \cdot P_{pub})) e(b_p Y, P_{pub})^{u^*} e(R_p^*, \eta^* P).$$

$\mathcal{C}$ calculates and outputs the required *xY* as $b_p^{-1} u^{*-1} (s_p^* - \prod_{i=1}^{n} [(v^* P \cdot P_{pub}) b_i P] - \eta^* R_p^*)$.

(2)If $\mathcal{A}$ outputs a forgery of the form $(w^*, M^*, R_p^*, s_p^*)$, where $Q_p$ is the proxy signer, $Q_k$ is one of the original signers, and $(w^*, M^*) \notin PMS_{qu}$, and

$$e(s_p^*, P) = e(Q_k, H_2(w^*) \cdot P_{pub}) e(Q_p, \prod_{i=1}^{n} H_2(w^*) \cdot P_{pub}) e(Q_p, H_4(Q_p \| w^*) \cdot P_{pub}) e(R_p^*, H_3(Q_p \| w^* \| M^* \| R_p^*))$$

$(i \neq k)$ .(i.e., $\mathcal{A}$ forges a valid proxy multi-signature

and $Q_K$ plays the role of one of the original signers.) $\mathfrak{C}$ recovers $(p,Q_p,b_p,c_p)$ and $(i,Q_i,b_i,c_i)$ $(i=1,\mathrm{K},n)$ from the list $L_1$. Algorithm $\mathfrak{C}$ proceeds only if $c_p=0$ and $c_i=1$ $(i=1,\mathrm{K},n)$. Otherwise, $\mathfrak{C}$ outputs "failure" and stop. Then we know that $Q_k=b_kY$ and $Q_i=b_iP$, $(i=1,\mathrm{K},k-1,k+1,\mathrm{K},n)$. Next, algorithm $\mathfrak{C}$ recovers $(e^*,M^*,v^*,s^*)$ from $L_2$, $(Q_p,R^*,w^*,\eta^*)$ from the list $L_3$ $(H_3(Q_p\|R^*\|w^*))=\eta^*P$ and $(Q_i,w,u)$ from list $L_4$ $(H_4(Q_i\|w^*)=u^*)$. If $s^*=0$, then $\mathfrak{C}$ outputs "failure" and stop. Otherwise, we know $s^*=1$, and hence $H_2(M^*)$ was defined to be $v^*P\in G_1$. Then we can deduce

$$e(s_p^*,P)=e(b_KY,H_2(w^*)\cdot P_{pub})$$
$$e(P\prod_{i=1}^{n}b_iP,\prod_{i=1}^{n}(v^*P\cdot P_{pub}))e(b_pP,P_{pub})^{u^*}e(R_p^*,\eta^*P)$$

$\mathfrak{C}$ calculates and outputs the required $xY$ as

$$b_k^{-1}(s_p^*-v^*P\cdot P_{pub}-\prod_{i=1}^{n}[(v^*P\cdot P_{pub})b_iP]-u^*b_pP_{pub}-\eta^*R_p^*)$$

Adversary $A_{III}$ succeeds if events $(E_1,E_2,E_3,E_6,E_7)$ happen, the probability $\Pr[E_1\wedge E_2\wedge E_3\wedge E_6\wedge E_7]$ can be decomposed as

$$\Pr[E_1\wedge E_2\wedge E_3\wedge E_6\wedge E_7]$$
$$=\Pr[E_1]\Pr[E_2\mid E_1]\Pr[E_3\mid E_1\wedge E_2]$$
$$\Pr[E_6\mid E_1\wedge E_2\wedge E_3]\Pr[E_7\mid E_1\wedge E_2\wedge E_3\wedge E_6]$$
$$\geq\frac{1}{4}(1-q_s(q_{H_2}+q_s)2^{-k})(1-q_{PMS}(q_{H_3}+q_{PMS})2^{-k})$$
$$(1-\frac{1}{q_E+n})^{q_E+n}(\frac{1}{q_E+n})\Pr[E_6\mid E_1\wedge E_2\wedge E_3]$$

Adversary $A_{III}$ succeeds if events $(E_1,E_2,E_3,E_8,E_9)$ happen, the probability $\Pr[E_1\wedge E_2\wedge E_3\wedge E_8\wedge E_9]$ can be decomposed as

$$\Pr[E_1\wedge E_2\wedge E_3\wedge E_8\wedge E_9]$$
$$=\Pr[E_1]\Pr[E_2\mid E_1]\Pr[E_3\mid E_1\wedge E_2]\Pr[E_8\mid E_1\wedge E_2\wedge E_3]$$
$$\Pr[E_9\mid E_1\wedge E_2\wedge E_3\wedge E_8]$$
$$\geq\frac{1}{4}(1-q_s(q_{H_2}+q_s)2^{-k})(1-q_{PMS}(q_{H_3}+q_{PMS})2^{-k})$$
$$(1-\frac{1}{q_E+n})^{q_E+n}(\frac{1}{q_E+n})\Pr[E_8\mid E_1\wedge E_2\wedge E_3]$$
$$\Pr[E_1\wedge E_2\wedge E_3\wedge E_6\wedge E_7]+\Pr[E_1\wedge E_2\wedge E_3\wedge E_8\wedge E_9]$$
$$\geq\frac{1}{4}(1-q_s(q_{H_2}+q_s)2^{-k})(1-q_{PMS}(q_{H_3}+q_{PMS})2^{-k})$$
$$(1-\frac{1}{q_E+n})^{q_E+n}(\frac{\varepsilon}{q_E+n})$$
$$\geq\frac{1}{4}(1-q_s(q_{H_2}+q_s)2^{-k})(1-q_{PMS}(q_{H_3}+q_{PMS})2^{-k})(\frac{\varepsilon}{e(q_E+n)})$$

So, here exists a type *III* adversary $A_{III}$ can breaks the proxy multi-signature scheme, then there exists an algorithm $\mathcal{B}$ which is able to use $A_{III}$ to solve an instance of the *CDH* problem with a non-negligible probability.

ACKNOWLEDGEMENT

This work was supported by Funding of Jiangsu Innovation Program for Graduate Education (CXZZ12_0161), the Fundamental Research Funds for the Central Universities. A Project Funded by the Priority Academic Programme Development of Jiangsu Higher Education Institutions.

REFERENCE

[1] Zhan Jianhong, Cui Yuanbo, Xu min et al, "An efficient and provably secure proxy signature scheme", Journal of Convergence Information Technology, Vol.6, No.7, pp.294-302, July 2011.
[2] Xuan Hong, Yu Long, "A Novel Unidirectional Proxy Re-Signature Scheme and Its Application for MANETs", Journal of Computers, Vol.7, No.7, 2012.
[3] Xiangguo Cheng, Shaojie Zhou, Lifeng Guo et al, "An ID-Based Short Group Signature Scheme", Journal of Software, Vol.8, No.3, 2013.
[4] Yi Lijiang, Bai Guoqiang and Xiao Guozhen, "Proxy multi-signature scheme: a new type of proxy signature scheme", lectronics Letters, Vol. 36, No. 6, pp.527–528, 2000.
[5] Byoungcheon Lee, Heesun Kim and Kwangjo Kim, "Strong proxy signature and its applications", Proc. of ACISP'01, [S.1.], Springer-Verlag, pp.603–608, 2001.
[6] Ji Jiahui, Li Daxing and Wang Mingqiang, "New proxy multi-signature, multi-proxy signature and multi-proxy multi-signature schemes from bilinear pairings", Chinese Journal of Computers, Vol. 27, No. 10, pp.1429–1435, 2004.
[7] Chien-Lung Hsu, Tzong-Sun Wu and Wei-Hua He, "New proxy multi-signature Scheme", Applied Mathematics and Computation, Vol. 162, No. 3, pp.1201–1206, 2005.
[8] Fan Hai-Wei, Ming Yang, "Proxy multi-signature scheme in the standard model", Advanced Materials Research, vol.433-440, pp2077-2085, 2012.
[9] Cao Feng and Cao Zhenfu, "A secure identity-based proxy multi-signature scheme", Information Sciences, Vol. 179,No. 3, pp.292–302, 2009.
[10] Biing-Feng Wang, "On the security of an identity-based proxy multi-signature scheme", IET Information Security, Vol. 4, No. 2, pp.45–48, 2010.
[11] Ying Sun,Chunxiang Xu,Yong Yu and Bo Yang, "Improvement of a proxy multi-signature scheme without random oracles", Computer Communications, Vol. 34, No. 3, pp.257–263, 2011.
[12] Song Han, Elizabeth Chang, Jie Wang and Wanquan Liu, "New proxy signatures preserving privacy and as secure as ElGamal signatures", International Journal of Information Technology, vol.3, no.1, pp.1–6 , 2006.
[13] He Du, Jian Wang, Yanan Liu, "A New Anonymous but Accountable Secure Proxy Signature Scheme", International Journal of Digital Content Technology and its Applications, Vol 6, No.9, pp.204-210, 2012.
[14] Dan Boneh, Matthew Franklin, "Identity-based encryption from the weil pairing", In Advances in Cryptology – Crypto'2001, LNCS 2139, Springer-Verlag, pp. 213–229, 2001.
[15] Bo Zhang, Qiuliang Xu, "Certificateless proxy blind signature scheme from bilinear pairings", Second International Workshop on Knowledge Discovery and Data Mining, pp. 573-576, 2009.

**He Du** born in 1984. Ph.D. candidate. His current research area includes Applied Cryptography and Network Security.

**Jian Wang** born in 1968. Ph.D., Professor. Ph.D. supervisor. His research interest includes Broadcast Encryption, Security issues on Wireless Sensor Network and Ad-hoc, Identification Authentication, Security Metrics of Software and System, etc.