

# Proxy Signature with Revocation

Shengmin Xu<sup>1</sup>, Guomin Yang<sup>1(✉)</sup>, Yi Mu<sup>1</sup>, and Sha Ma<sup>1,2</sup>

<sup>1</sup> Centre for Computer and Information Security Research,  
School of Computing and Information Technology, University of Wollongong,  
Wollongong, NSW, Australia

{sx914,gyang,ymu,sma}@uow.edu.au

<sup>2</sup> College of Mathematics and Informatics, South China Agricultural University,  
Guangzhou 510640, Guangdong, China

**Abstract.** Proxy signature is a useful cryptographic primitive that allows signing right delegation. In a proxy signature scheme, an original signer can delegate his/her signing right to a proxy signer (or a group of proxy signers) who can then sign documents on behalf of the original signer. In this paper, we investigate the problem of proxy signature with revocation. The revocation of delegated signing right is necessary for a proxy signature scheme when the proxy signer's key is compromised and/or any misuse of the delegated right is noticed. Although a proxy signature scheme usually specifies a delegation time period, it may happen that the original signer wants to terminate the delegation before it is expired. In order to solve this problem, in this paper we propose a new proxy signature scheme with revocation. Our scheme utilises and combines the techniques in the Naor-Naor-Lotspeich (NNL) framework for broadcast encryption, the Boneh-Boyen-Goh (BBG) hierarchical identity-based encryption and the Boneh-Lynn-Shacham (BLS) short signature scheme and thereby constructing an efficient tree-based revocation mechanism. The unrevoked proxy signer only needs to generate evidences for proving that he/she is a valid proxy signer once in per revocation epoch, and the verifier does not need a revocation list in order to verify the validity of a proxy signature.

**Keywords:** Proxy signature · Revocation · Hierarchical structure

## 1 Introduction

Mambo, Usuda and Okamoto introduced the concept of proxy signatures in 1996 [16, 17]. In a proxy signature scheme, an original signer is allowed to delegate his signing power to a designated person called the proxy signer, and then the proxy signer is able to sign the message on behalf of the original signer.

There are four types of delegation in proxy signature. Mambo et al. [16] proposed three of them in their seminal work: full delegation, partial delegation and delegation by warrant. In the full delegation, the original signer just gives his signing key to the proxy signer as the proxy signing key. Thus, the proxy signer has the same signing ability as the original signer so that the real

signer of a signature is indistinguishable. To overcome this drawback, partial delegation was proposed, in which the original signer and the proxy signer work together to derive the proxy signing key that consists of partial private keys of the original signer and the proxy signer. Partial delegation is further classified into proxy-unprotected delegation and proxy-protected delegation [11]. In proxy-unprotected partial delegation, the original signer can derive the proxy signing key without the interaction with the proxy signer, but the proxy signer cannot derive the proxy signing key without the help from the original signer. In the case of proxy-protected partial delegation, the proxy signing key needs the contribution of both the proxy signer and the original signer. However, in the partial delegation, the proxy signer has unlimited signing ability. To conquer this problem, delegation by warrant has been proposed. The original signer signs a warrant that certifies the legitimacy of the proxy signer. Kim et al. [10] later proposed a new type of proxy delegation called partial delegation with warrant combining advantages of partial delegation and delegation with warrant.

Besides, proxy signature can be categorized into proxy multi-signature scheme and multi-proxy signature scheme. In a proxy multi-signature scheme [13, 22], a designed proxy signer can generate the signature on behalf of two or more original signers. In the case of multi-proxy signature scheme [12, 21], it allows a group of original signers to delegate the signing capability to a designated group of proxy signers.

### 1.1 Motivation of This Work

In this paper, we focus on proxy signature with revocation. Although there are many research works on proxy signature, only few of them deal with proxy revocation. It is necessary to address the problem of proxy revocation in proxy signature when the proxy signer is compromised. Moreover, in reality, the proxy signer may also misuse the delegated signing right. In such situations, the original signer should have a way to revoke the signing right delegated to the proxy signer even when the delegation has not expired. One straightforward solution to address this problem is to let the original signer publish a revocation/black list and a verifier needs to check the list before verifying a proxy signature. One limitation of such an approach is that the verifier needs to obtain the latest revocation list before verifying a proxy signature. Another problem brought by this approach is that a proxy signature generated before the proxy signer is revoked also becomes invalid. Ideally, such proxy signatures should still be considered valid since the proxy signer is not revoked when the signature is generated.

In [20], Sun suggested that the revocation problem can be solved by using a timestamp and proposed a proxy signature which allows the verifier to trace the proxy signer. However, the proposed scheme has some security issues. As pointed out in [4], an attacker can easily forge a proxy signature.

Another solution proposed in the literature to address the problem is utilizing a trusted third party. Das et al. [4] and Lu et al. [15] proposed some proxy signature schemes with revocation where a trusted third party called the

authentication server (AS) is used to provide the immediate revocation. However, a trusted third party is a very strong assumption. Hence, such a solution is not very practical in real applications.

The third solution that has been proposed by Seo et al. [19] and Liu et al. [14] is to use a third party called SEcurity Mediator (SEM) which is a partially trusted online server. In such a solution, the original signer divides the delegation into two parts and gives these two parts to the proxy signer and the SEM, respectively. When the proxy signer wants to generate a proxy signature, he/she must get the assistance from the SEM. Thus, the SEM works as a certifier to authenticate the signing ability of every proxy signer. Such a solution is not practical either since whenever the proxy signer wants to generate a proxy signature, he/she needs to contact the SEM which is a bottleneck of the system.

## 1.2 Our Result

In this paper, we introduce a novel proxy signature scheme with revocation. Compared with the previous solutions, our scheme has the following advantages.

- Our scheme does not need any third party. In addition, the verifier does not need to obtain the revocation list in order to verify a proxy signature. Instead he/she only needs to know the current revocation epoch in order to verify a proxy signature.
- The original signer can revoke a set of proxy signers in each revocation epoch. An unrevoked proxy signer only needs to generate once in each revocation epoch a proof which shows his/her valid proxy signing right.
- Our scheme explicitly includes the revocation epoch in signature verification, and hence, the verifier only denies signatures generated by a proxy signer after his/her proxy signing right is revoked. The signatures generated before revocation will remain valid.

## 1.3 Outline of Paper

The rest of this paper is organized as follows. Some preliminaries are presented in Sect. 2. The formal security models for our scheme is described in Sect. 3. The proposed proxy signature with revocation scheme is detailed in Sect. 4. We analyze the proposed scheme in Sect. 5. Finally, some concluding remarks are given in Sect. 6.

# 2 Preliminaries

In this section, we provide some background knowledge used in this paper.

## 2.1 Bilinear Map

Let  $\mathbb{G}$  and  $\mathbb{G}_T$  denote two cyclic multiplicative groups of prime order  $p$  and  $g$  be a generator of  $\mathbb{G}$ . The map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  is said to be an admissible bilinear map if the following properties hold.

1. Bilinearity: for all  $u, v \in \mathbb{G}$  and  $a, b \in \mathbb{Z}_p$ ,  $e(u^a, v^b) = e(u, v)^{ab}$ .
2. Non-degeneration:  $e(g, g) \neq 1$ .
3. Computability: it is efficient to compute  $e(u, v)$  for any  $u, v \in \mathbb{G}$ .

We say that  $(\mathbb{G}, \mathbb{G}_T)$  are bilinear groups if there exists a bilinear map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  as above.

## 2.2 Complexity Assumptions

**Definition 1** (*Computational Diffie-Hellman (CDH) problem*). Given  $g, g^a, g^b \in \mathbb{G}$  for some unknown  $a, b \in \mathbb{Z}_p$ , the computational Diffie-Hellman (CDH) problem is to compute  $g^{ab} \in \mathbb{G}$ .

**Definition 2** (*Computational Diffie-Hellman (CDH) assumption*). The  $(t, \epsilon)$ -CDH assumption holds in group  $\mathbb{G}$  if no algorithm with running time  $t$  has probability at least  $\epsilon$  in solving the CDH problem.

## 2.3 Digital Signature Scheme

A digital signature scheme consists of three algorithms [6]:

**Key generation**  $\mathcal{G}(1^k)$ : it inputs a security parameter  $k$  and outputs in polynomial time a pair  $(pk, sk)$  of matching public and secret keys.

**Signature**  $\mathcal{S}_{sk}(m)$ : it produces a signature  $\sigma \leftarrow \mathcal{S}_{sk}(m)$  for a message  $m$  using the secret key  $sk$ .

**Verification**  $\mathcal{V}_{pk}(m, \sigma)$ : it tests whether  $\sigma$  is a valid signature for message  $m$  using the public key  $pk$ . The algorithm outputs either 1 (valid) or 0 (invalid).

## 2.4 Security Model for Existential Unforgeability

The de facto security notion is existential unforgeability under adaptive chosen message attacks [6] which is defined using the following game.

**Setup:** The challenger runs  $\mathcal{G}$ . It gives the adversary the resulting public key  $pk$  and keeps the private key  $sk$  to itself.

**Signing Query** ( $\mathcal{O}_{\mathcal{EU}_S}$ ): The adversary issues signing queries  $m_1, \dots, m_q$ . To each query  $m_i$ , the challenger responds by running  $\mathcal{S}$  to generate a signature  $\sigma_i$  of  $m_i$  and sending  $\sigma_i$  to the adversary. These queries may be asked adaptively so that each query  $m_i$  may depend on the replies to  $m_1, \dots, m_{i-1}$ . A database  $D_{\mathcal{EU}_S}$  to record the messages have been signed.

**Output:** Finally the adversary outputs a pair  $(m^*, \sigma^*)$ . The adversary wins if  $\sigma^*$  is a valid signature of  $m^*$  according to  $\mathcal{V}$  and  $m^*$  is not among the messages  $D_{\mathcal{EU}_S}$  appeared during the query phase.

**Definition 3.** A signature scheme is  $(t, q, \epsilon)$  existentially unforgeable under adaptive chosen message attacks if no  $t$ -time adversary  $\mathcal{A}_{\mathcal{EU}}$  making at most  $q$  signing queries has advantage at least  $\epsilon$  in the above game. For any PPT adversary  $\mathcal{A}_{\mathcal{EU}}$  involved in the experiment hereafter, we have  $\text{Adv}_{\mathcal{A}_{\mathcal{EU}}}^{\text{eu-cma}}(\lambda) = \Pr[\text{Expt}_{\mathcal{A}_{\mathcal{EU}}}^{\text{eu-cma}}(\lambda) = 1] \in \text{negl}(\lambda)$ .

Experiment $\text{Exp}_{\mathcal{A}_{\mathcal{EU}}}^{\text{eu-cma}}(\lambda)$	Oracle $\mathcal{O}_{\mathcal{EU}_S}(m)$
$(pk, sk) \leftarrow \text{Gen}(1^\lambda); D_{\mathcal{EU}_S} \leftarrow \emptyset$	$\sigma \leftarrow \text{Sign}(sk, m)$
$\sigma \leftarrow \mathcal{A}_{\mathcal{EU}}^{\mathcal{O}_{\mathcal{EU}_S}}(m)$	$D_{\mathcal{EU}_S} \leftarrow D_{\mathcal{EU}_S} \cup m$
$(m^*, \sigma^*) \leftarrow \mathcal{A}_{\mathcal{EU}}(pk, \mathcal{O}_{\mathcal{EU}_S})$	Return $\sigma$
If $\text{Ver}(pk, m^*, \sigma^*) = 1$ , and	
$m^* \notin D_{\mathcal{EU}_S}$ return 1 else return 0	

## 2.5 Boneh-Lynn-Shacham Short Signature Scheme

BLS Short Signature Scheme was proposed in [3]. We use this short signature as a primitive to provide authentication in our hierarchical revocation algorithm. Some details of the BLS short signature are given below.

**Keygen:** The public key is  $(\mathbb{G}, \mathbb{G}_T, q, g, y, \mathcal{H}_1)$  and secret key is  $s$ , where  $y = g^s$  and  $\mathcal{H}_1 : \{0, 1\}^* \rightarrow \mathbb{G}$  is a hash function.

**Sign:** The signature for message  $m$  is  $\sigma = h^s$ , where  $h = \mathcal{H}_1(m)$ .

**Verify:** Check whether the equation  $e(\sigma, g) = e(\mathcal{H}_1(m), y)$  holds.

This scheme has been proven to be secure against adaptive chosen-message attacks in the random oracle model assuming the CDH problem is hard.

## 2.6 Boneh-Boyen-Goh Hierarchical Identity-Based Encryption

Hierarchical identity-based encryption (HIBE) is a generalization of identity-based encryption and mirrors an organizational hierarchy. An identity at level  $k$  of the hierarchy tree can issue private keys to its descendant identities, but cannot decrypt messages intended for other identities. Boneh et al. [2] described the first HIBE scheme where the size of ciphertext does not depend on the depth of the receiver in the hierarchy. This HIBE scheme will be modified as an important part in our hierarchical revocation algorithm. The BBG HIBE scheme, which has five algorithms, is reviewed below.

**Setup:** The master public key is  $(\mathbb{G}, \mathbb{G}_T, g, g_1, g_2, \{h_i\}_{i=0}^\ell)$  and master secret key is  $g_2^\alpha$ , where  $\ell$  is the number of levels in the hierarchy,  $g_1 = g^\alpha$  and  $\alpha \in \mathbb{Z}_p$  is a random number and  $h_0, h_1, \dots, h_\ell \in \mathbb{G}$ .

**Keygen:** Given master secret key  $msk$  and an identity  $id = (I_1, \dots, I_k)$ , it will choose a random numbers  $r \in \mathbb{Z}_p$  and generate the private key  $d_{id} = (D_1, D_2, K_{k+1}, \dots, K_\ell)$ .  $D_1$  and  $D_2$  are decryption keys.  $(K_{k+1}, \dots, K_\ell)$  is the delegation part and it is used to derive decryption keys for descendant identities.

$$D_1 = g_2^\alpha \cdot (h_0 \cdot \prod_{i=1}^k h_i^{I_i})^r, D_2 = g^r, K_i = h_i^r \text{ for } i = k+1, \dots, \ell.$$

**Derive:** Given the private key  $d_{id}$  and an identity  $id' = (I_1, \dots, I_k, I_{k+1})$  that is the descendant of  $id = (I_1, \dots, I_k)$ , it chooses a random number  $r \in \mathbb{Z}_p$  and outputs a private key  $d_{id'} = (D'_1, D'_2, K'_{k+2}, \dots, K'_\ell)$  for  $id'$ .

$$d_{id'} = (D_1 \cdot K_{d+1}^{I_{k+1}} \cdot (h_0 \cdot \prod_{i=1}^{k+1} h_i^{I_i})^{r'}, D_2 \cdot g^{r'}, K_{k+2} \cdot h_{k+2}^{r'}, \dots, K_\ell \cdot h_\ell^{r'}).$$

**Encrypt:** Given the master public key  $mpk$ , an identity  $id = (I_1, \dots, I_d)$  and a message  $m$ , it outputs a ciphertext  $C = (C_0, C_1, C_2)$  by choosing a random number  $s \in \mathbb{Z}_p$  and computing the following elements

$$C_0 = m \cdot e(g_1, g_2)^s, \quad C_1 = g^s, \quad C_2 = (h_0 \cdot h_1^{I_1} \cdots h_d^{I_d})^s.$$

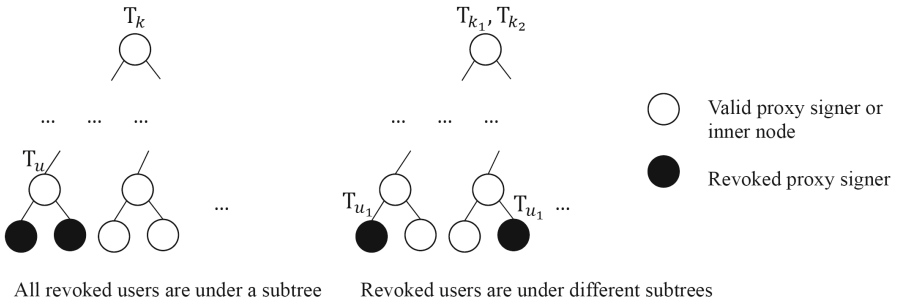
**Decrypt:** It returns  $M = C_0 \cdot e(C_1, D_1)^{-1} \cdot e(C_2, D_2)$ .

This scheme has been proven to be selective-ID secure in the standard model and fully secure in the random oracle model.

## 2.7 Naor-Naor-Lotspiech Framework for Broadcast Encryption

Naor et al. [18] introduced a subset cover framework for broadcast encryption. This framework is based on complete subtree (CS) method and subset difference (SD) method. Halevy and Shamir [7] proposed a new method called layered subset difference (LSD) to improve the key distribution in the SD method. Later, Dodis and Fazio [5] pointed out that HIBE schemes can base on the above methods. In this section, we will briefly introduce the SD method.

The SD method works like a white list and we call it a revocation list in this paper. Each user is assigned to a leaf node in the tree and given the private keys of all co-path nodes from the root to the leaf. Let  $\mathcal{N}$  denote all the users and  $\mathcal{R}$  the revoked users. This method will group the valid users ( $\mathcal{N} \setminus \mathcal{R}$ ) into  $m$  sets  $S_{k_1, u_1}, \dots, S_{k_m, u_m}$ . Each valid user belongs to at least one set, the number of set  $m$  satisfies  $m \leq 2|\mathcal{R}| - 1$ . Let  $T_{x_j}$  denote the subtree rooted at  $x_j$ .



**Fig. 1.** The SD method

The subset  $S_{k_i, u_i}$  is defined as follows.  $T_{k_i}$  is called the primitive root.  $T_{u_i}$  is called the secondary root, and  $T_{u_i}$  is a descendant of  $T_{k_i}$ . The valid users in the set  $S_{k_i, u_i}$  consists of the leaves of  $T_{k_i}$  that are not in  $T_{u_i}$ . Thus, each user may belong to more than one set.

### 3 Formal Definitions and Security Models

In this section, we will introduce the syntax of a hierarchical revocation algorithm and a proxy signature with revocation and their formal security models. Here, we provide the details of some notations that will be used in this section.

- $\mathcal{N}$  is the set of proxy signers, and  $|\mathcal{N}|$  is the number of proxy signer.
- $\mathcal{R}$  is the set of revoked proxy signers, and  $|\mathcal{R}|$  is the number of revoked proxy signer.  $\mathcal{R}_t$  is the set of revoked proxy signers in the revocation epoch  $t$ .
- $\ell \in \mathbb{Z}$  is the maximum level of the tree and  $|\mathcal{N}| \leq 2^\ell$ .
- $id \in \{0, 1\}^{\leq \ell}$  is the label value for each node in the tree.
- $prefix(id) \in \{0, 1\}^{\leq \ell}$  is the set of label values which are the prefix of  $id$ .
- $w \in \mathbb{Z}$  is a warrant for signing right delegation.
- $d_{id_i} = (D_{i,1}, D_{i,2}, K_{i,1}, \dots, K_{i,\ell-|id_i|+1})$  is the hierarchical private key for  $id_i$ .

#### 3.1 Hierarchical Revocation Scheme

This hierarchical revocation scheme is derived from the Boneh-Boyen-Goh hierarchical identity based encryption scheme (BBG HIBE) [2] and is an essential part of our proxy signature with revocation scheme. This scheme keeps a white list to reject all the revoked proxy signers and the size of this revocation list is  $O(|\mathcal{R}|)$  since we use the Subset Difference (SD) method in the Naor-Naor-Lotspiech framework [18]. This scheme can be described using the following algorithms.

**Setup**( $1^\lambda, 1^\ell$ ): Given a security parameter  $\lambda$  and a maximum level  $\ell$  of the complete binary tree, it outputs the system parameter  $param$ , the master secret key  $msk$  and the master public key  $mpk$ .

**Keygen**( $w_i, pk_i, msk, id$ ): Given a proxy signer's warrant  $w_i$  and his/her public key  $pk_i$ , master secret key  $msk$ , the master public key  $mpk$  and the label value  $id$  in the tree, it outputs a hierarchical private key  $d_{id}$ , where  $d_{id}$  includes the decryption key and delegation key as shown in the HIBE scheme reviewed above.

**Derive**( $mpk, id, d_{id}, id'$ ): Given master public key  $mpk$ , a label value  $id$  and its hierarchical private key  $d_{id}$  and a label value  $id'$ , which is a descendant of  $id$  in the tree structure, it outputs another hierarchical private key  $d_{id'}$  for  $id'$ .

**Encode**( $mpk, id, id'$ ): Given master public key  $mpk$ , a label value  $id$  and another label value  $id'$  which is a descendant of  $id$ , it outputs an encoding value  $C$ .

**Verify**( $mpk, w_i, pk_i, id, C, d_{id'}$ ): Given master public key  $mpk$ , a proxy signer's warrant  $w_i$  and his/her public key  $pk_i$ , a label value  $id$ , an encoding value  $C$  (with regards to  $id$  and  $id'$ ) and a hierarchical private key  $d_{id'}$ , it outputs either 1 or 0.

**Security Model for Hierarchical Revocation Algorithm.** We propose a security notion called key robustness to define the security of our hierarchical revocation algorithm. The security model is defined using the following game:

**Setup:** The challenger runs *Setup*. It gives the adversary the resulting of master public key  $mpk$  and keeps the master private key  $msk$  to itself.

**Keygen Query** ( $\mathcal{O}_{\mathcal{A}_G}$ ): The adversary issues up to  $q_G$  key generations queries  $\{(id_i, w_i, pk_i)\}_{i=1}^{q_G}$ . To each  $(id_i, w_i, pk_i)$ , the challenger responds by running *Keygen* to generate a result  $d_{id_i}$  for  $(id_i, w_i, pk_i)$  and sending  $d_{id_i}$  to the adversary. These queries may be asked adaptively so that each query  $(id_i, w_i, pk_i)$  may depend on the replies to  $(id_1, w_1, pk_1), \dots, (id_{i-1}, w_{i-1}, pk_{i-1})$ . A database  $D_{\mathcal{A}_G}$  records all the messages that have been queried.

**Output:** Finally the adversary outputs  $(id^*, id^{*'}, w^*, C^*, pk^*, d_{id^{*'}}^*)$  such that  $C^*$  is an encoding with regards to  $id^*$  and  $id^{*'}$ . The adversary wins if  $(id^{*'}, w^*, pk^*)$  or  $(prefix(id^{*'}), w^*, pk^*)$  has not appeared in any *Keygen queries*, and  $(mpk, w^*, pk^*, id^*, C^*, d_{id^{*'}}^*)$  can pass the verification.

In the random oracle model, we have an additional oracle called hash oracle:

**Hash Query** ( $\mathcal{O}_{\mathcal{A}_H}$ ): The adversary issues hash queries  $\{(id_i, w_i, pk_i)\}_{i=1}^{q_H}$ . To each  $(id_i, w_i, pk_i)$ , the challenger responds by returning a random element in the range of the hash function  $\mathcal{H}_1$ . The same result is returned if the same input is queried for more than one time.

**Definition 4.** A hierarchical revocation scheme is  $(t, q_H, q_G, \epsilon)$  key robust if no  $t$ -time adversary  $\mathcal{A}$  making at most  $q_H$  hash queries and  $q_G$  keygen queries has advantage at least  $\epsilon$  in the above game. For any PPT adversary  $\mathcal{A}$  involved in the experiment hereafter, we have  $\text{Adv}_{\mathcal{A}}^{\text{key-robust}}(\lambda) = \Pr[\text{Expt}_{\mathcal{A}}^{\text{key-robust}}(\lambda, \ell) = 1] \in \text{negl}(\lambda)$ .

Oracle $\mathcal{O}_{\mathcal{A}_H}(id, w, pk)$ Return $\mathcal{H}_1(id, w, pk)$	Oracle $\mathcal{O}_{\mathcal{A}_G}(id, w, pk)$ $D_{\mathcal{A}_G} \leftarrow D_{\mathcal{A}_G} \cup (id, w, pk)$ Return <i>keygen</i> ( $w, pk, msk, id$ )
Experiment $\text{Expt}_{\mathcal{A}}^{\text{key-robust}}(\lambda, \ell)$	
$(mpk, msk) \leftarrow \text{Setup}(1^\lambda, 1^\ell); D_{\mathcal{A}_G} \leftarrow \emptyset$	
$H_i \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{A}_H}}(id_i, w_i, pk_i); d_{id_i} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{A}_G}}(id_i, w_i, pk_i)$	
$(id^*, id^{*'}, w^*, C^*, pk^*, d_{id^{*'}}^*) \leftarrow \mathcal{A}(mpk, \mathcal{O}_{\mathcal{A}_H}, \mathcal{O}_{\mathcal{A}_G})$	
If $\text{Verify}(mpk, w^*, pk^*, id^*, C^*, d_{id^{*'}}^*) \rightarrow 1$ , $(id^{*'}, w^*, pk^*) \notin D_{\mathcal{A}_G}$ , and $(prefix(id^{*'}), w^*, pk^*)^* \notin D_{\mathcal{A}_G}$ return 1 else return 0	

### 3.2 Proxy Signature with Revocation

In our scheme, there are two parties: an original signer  $\mathcal{O}$  and a group of proxy signers  $\mathcal{P}_i$  for  $i = 1, \dots, |\mathcal{N}|$ . A proxy signature scheme with revocation can be described as a collection of the following algorithms:

**Setup**( $1^\lambda, 1^\ell$ ): Given a system security parameter  $\lambda$  and a maximum level of the complete binary tree that defines the maximum number of the proxy signers  $|\mathcal{N}| = 2^\ell$ , it outputs the system parameters  $\mathcal{Y}$ .



**Keygen**( $1^\lambda, \mathcal{Y}$ ): Given a system security parameter  $\lambda$  and the system parameters  $\mathcal{Y}$ , it outputs a pair of public and secret key  $(pk, sk)$ . The original signer runs this algorithm to generate its own public  $pk_o$  and security key  $sk_o$ . The proxy signers runs this function to generate its own public  $pk_i$  and security key  $sk_i$ .

**Delegation**( $\mathcal{Y}, w_i, pk_i, pk_o, sk_o$ ): Given a system parameters  $\mathcal{Y}$ , the warrant  $w_i$  and public key  $pk_i$  of the proxy signer  $\mathcal{P}_i$  and the public key  $pk_o$  and secret key  $sk_o$  of original signer, it generates the delegated information  $I_i$ .

**Revocation**( $\mathcal{Y}, sk_o, t, \mathcal{R}_t$ ): Given a system parameters  $\mathcal{Y}$ , the secret key  $sk_o$  of original signer, the current revocation epoch  $t$  and the set of revoked proxy signers  $\mathcal{R}_t$ , it outputs a revocation list  $RL_t$  under the revocation epoch  $t$ .

**Sign**( $\mathcal{Y}, RL_t, sk_i, I_i, M$ ): Given a system parameters  $\mathcal{Y}$ , the revocation list  $RL_t$  under revocation epoch  $t$ , the secret key  $sk_i$  and delegated information  $I_i$  of proxy signer  $\mathcal{P}_i$  and a message  $M$ , it outputs a proxy signature  $\sigma$ .

**Verify**( $\mathcal{Y}, t, pk_i, pk_o, M, \sigma$ ): Given a system parameter  $\mathcal{Y}$ , the revocation epoch  $t$ , public key  $pk_i$  of proxy signer, public key  $pk_o$  of original signer, the message  $M$  and the proxy signature  $\sigma$ , it outputs either 1 or 0.

**Security Models for Proxy Signature with Revocation.** To define the unforgeability of our proxy signature scheme with revocation, according to the classification of Huang et al. [8] and their continuing work [9], we divide the adversaries into the following four types<sup>1</sup>:

1. *Type I*: This type of adversary  $\mathcal{A}_I$  has public parameter  $\mathcal{Y}$ , public key of original signer  $pk_o$ , and public keys of all proxy signers  $\{pk_i\}_{i=1}^{|\mathcal{N}|}$ .
2. *Type II*: This type of adversary  $\mathcal{A}_{II}$  has public parameter  $\mathcal{Y}$ , public key of original signer  $pk_o$ , public keys of all proxy signers  $\{pk_i\}_{i=1}^{|\mathcal{N}|}$ , and the secret key of original signer  $sk_o$ .
3. *Type III*: This type of adversary  $\mathcal{A}_{III}$  has public parameter  $\mathcal{Y}$ , public key of original signer  $pk_o$ , public keys of all proxy signers  $\{pk_i\}_{i=1}^{|\mathcal{N}|}$ , and secret keys of all proxy signers  $\{sk_i\}_{i=1}^{|\mathcal{N}|}$ .
4. *Type IV*: This type of adversary  $\mathcal{A}_{IV}$  has public parameter  $\mathcal{Y}$ , public key of original signer  $pk_o$ , public keys of all proxy signers  $\{pk_i\}_{i=1}^{|\mathcal{N}|}$ , and the secret key and delegated information of all revoked proxy signers  $\{sk_i, I_i\}_{i \in \mathcal{R}}$ <sup>2</sup>.

One can find that if our proxy signature scheme is secure against *Type II* (or *Type III* or *Type IV*) adversary, our scheme is also unforgeable against *Type I* adversary. Below we give the formal security models.

**Security Model for Adversary  $\mathcal{A}_{II}$ .** Adversary  $\mathcal{A}_{II}$  represents original signer, who wants to generate a valid proxy signature without knowing the secret key of the proxy signer. The security model is defined using the following game:

<sup>1</sup> In all the security models, we assume that there is only one set of revoked signers  $\mathcal{R}_{t_i}$  for each revocation epoch  $t_i$ .

<sup>2</sup> For achieving the backward security [1], we need the time stamp server to generate the time certificate for each proxy signature.

**Setup:** The challenger generates  $|\mathcal{N}| + 1$  public key and secret key pairs and assigns them to the original signer and proxy signers. Then it gives the adversary the system parameter  $\mathcal{Y}$ , the public keys of original signer  $pk_o$  and proxy signers  $\{pk_i\}_{i=1}^{|\mathcal{N}|}$ , secret key of original signer  $sk_o$ , and keeps the secret keys of all proxy signers  $\{sk_i\}_{i=1}^{|\mathcal{N}|}$  to itself.

**Signing Query** ( $\mathcal{O}_{\mathcal{IIS}}$ ): The adversary issues signing queries  $\{(w_i, pk_i, M_i, t_i, \mathcal{R}_{t_i})\}_{i=1}^q$  where  $pk_i \notin \mathcal{R}_{t_i}$ . The challenger responds by running *Delegation* algorithm to get delegated information  $I_i$ , *Revocation* algorithm to get revocation list  $RL_{t_i}$ , and *Sign* algorithm to get the proxy signature  $\sigma_i$ . After that, the challenger sends  $\sigma_i$  to the adversary. These queries may be asked adaptively so that each query  $(w_i, pk_i, M_i, t_i, \mathcal{R}_{t_i})$  may depend on the replies to all previous queries. A database  $D_{\mathcal{IIS}}$  records all the information of queries. If  $pk_i \in \mathcal{R}_{t_i}$ , the challenger rejects the query.

**Output:** Finally, the adversary outputs  $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}, \sigma^*)$ . The adversary wins if  $pk^*$  is one of the proxy signer public keys that have been given,  $(w^*, pk^*, t^*, \mathcal{R}_{t^*}, M^*)$  does not appear in  $D_{\mathcal{IIS}}$ , and  $(\mathcal{Y}, t^*, pk^*, pk_o, M^*, \sigma^*)$  can pass the verification.

**Definition 5.** A proxy signature scheme is  $(t, q, \epsilon)$  existentially unforgeable under Type-II adaptive chosen message attacks if no  $t$ -time adversary  $\mathcal{A}_{\mathcal{II}}$  making at most  $q$  signing queries has advantage at least  $\epsilon$  in the above game. For any PPT adversary  $\mathcal{A}_{\mathcal{II}}$  involved in the experiment hereafter, we have  $\text{Adv}_{\mathcal{A}_{\mathcal{II}}}^{\text{eu-cma}}(\lambda) = \Pr[\text{Expt}_{\mathcal{A}_{\mathcal{II}}}^{\text{eu-cma}}(\lambda, \ell) = 1] \in \text{negl}(\lambda)$ .

---

Oracle  $\mathcal{O}_{\mathcal{IIS}}(w, pk, M, t, \mathcal{R}_t)$   
 $I \leftarrow \text{Delegation}(\mathcal{Y}, w, pk, pk_o, sk_o)$ ;  $RL_t \leftarrow \text{Revocation}(\mathcal{Y}, sk_o, t, \mathcal{R}_t)$   
 $\sigma \leftarrow \text{Sign}(\mathcal{Y}, RL_t, sk, I, M)$ ;  $D_{\mathcal{IIS}} \leftarrow D_{\mathcal{IIS}} \cup (w, pk, M, t, \mathcal{R}_t)$   
Return  $\sigma$

---

Experiment  $\text{Exp}_{\mathcal{A}_{\mathcal{II}}}^{\text{eu-cma}}(\lambda, \ell)$   
 $(\mathcal{Y}, pk_o, sk_o, \{pk_i, sk_i\}_{i=1}^{|\mathcal{N}|}) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$ ;  $D_{\mathcal{IIS}} \leftarrow \emptyset$   
 $\sigma \leftarrow \mathcal{A}_{\mathcal{II}}^{\mathcal{O}_{\mathcal{IIS}}}(w, pk, M, t, \mathcal{R}_t)$   
 $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}, \sigma^*) \leftarrow \mathcal{A}_{\mathcal{II}}(\mathcal{Y}, pk_o, sk_o, \{pk_i\}_{i=1}^{|\mathcal{N}|}, \mathcal{O}_{\mathcal{IIS}})$   
If  $\text{Verify}(\mathcal{Y}, t^*, pk^*, pk_o, M^*, \sigma^*) = 1$ ,  $pk^* \in \{pk_i\}_{i=1}^{|\mathcal{N}|}$ , and  
 $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}) \notin D_{\mathcal{IIS}}$  return 1 else return 0

**Security Model for Adversary  $\mathcal{A}_{\mathcal{III}}$ .** Adversary  $\mathcal{A}_{\mathcal{III}}$  represents proxy signers, who want to generate the proxy signature without knowing the delegated information. The security model is defined using the following game:

**Setup:** The challenger generates  $|\mathcal{N}| + 1$  public key and secret key pairs and assigns them to original signer and proxy signers. Then it gives the adversary the system parameter  $\mathcal{Y}$ , the public keys of original signer  $pk_o$  and proxy signers  $\{pk_i\}_{i=1}^{|\mathcal{N}|}$ , secret keys of proxy signers  $\{sk_i\}_{i=1}^{|\mathcal{N}|}$ , and keeps the secret key of original signer  $sk_o$  to itself.

**Delegation Query** ( $\mathcal{O}_{III\mathcal{D}}$ ): The adversary issues up to  $q_D$  delegation queries. To each  $(w_i, pk_i)$ , the challenger responds by running *Delegation* algorithm to gain the delegated information  $I_i$  and the challenger sends  $I_i$  to the adversary. These queries may be asked adaptively. A database  $D_{III\mathcal{D}}$  records all the delegation queries.

**Revocation Query** ( $\mathcal{O}_{III\mathcal{R}}$ ): The adversary issues up to  $q_R$  revocation queries  $(t_i, \mathcal{R}_{t_i})$ . To each query, the challenger responds by executing *Revocation* algorithm to get the revocation list  $RL_{t_i}$  for revocation epoch  $t_i$ . Then the challenger sends  $RL_{t_i}$  to the adversary. These queries may be asked adaptively. Notice that we assume there is only one  $\mathcal{R}_{t_i}$  for each  $t_i$ .

**Signing Query** ( $\mathcal{O}_{III\mathcal{S}}$ ): The adversary makes up to  $q_S$  signing queries to the challenger. For each  $(w_i, pk_i, M_i, t_i, \mathcal{R}_{t_i})$  where  $pk_i \notin \mathcal{R}_{t_i}$ , the challenger gains the delegated information  $I_i$  by running *Delegation* algorithm, runs the *Revocation* algorithm to get the revocation list  $RL_{t_i}$ , and executes the *Sign* algorithm to acquire the proxy signature  $\sigma_i$ . These queries may be asked adaptively. A database  $D_{III\mathcal{S}}$  records all the signing queries.

**Output:** Finally, the adversary outputs  $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}, \sigma^*)$ . The adversary wins if  $pk^*$  is one of the proxy signer's public keys given,  $(w^*, pk^*)$  has not been queried to *Delegation* oracle,  $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*})$  has not been queried to *Signing* oracle, and  $(\mathcal{Y}, t^*, pk^*, pk_o, M^*, \sigma^*)$  can pass verification.

**Definition 6.** A proxy signature scheme is  $(t, q_D, q_R, q_S, \epsilon)$  existentially unforgeable under Type-III adaptive chosen message attacks if no  $t$ -time adversary  $\mathcal{A}_{III}$  making at most  $q_D$  delegation queries,  $q_R$  revocation queries and  $q_S$  signing queries has advantage at least  $\epsilon$  in the above game. For any PPT adversary  $\mathcal{A}_{III}$  involved in the experiment hereafter, we have  $\text{Adv}_{\mathcal{A}_{III}}^{\text{eu-cma}}(\lambda) = \Pr[\text{Exp}_{\mathcal{A}_{III}}^{\text{eu-cma}}(\lambda) = 1] \in \text{negl}(\lambda)$ .

Oracle $\mathcal{O}_{III\mathcal{D}}(w, pk)$ $I \leftarrow \text{Delegation}(\mathcal{Y}, w, pk, pk_o, sk_o)$ $D_{III\mathcal{D}} \leftarrow D_{III\mathcal{D}} \cup (w, pk)$ Return $I$	Oracle $\mathcal{O}_{III\mathcal{S}}(w, pk, M, t, \mathcal{R}_t)$ $I \leftarrow \text{Delegation}(\mathcal{Y}, w, pk, pk_o, sk_o)$ $RL_t \leftarrow \text{Revocation}(\mathcal{Y}, sk_o, t, \mathcal{R}_t)$ $\sigma \leftarrow \text{Sign}(\mathcal{Y}, RL_t, sk, I, M)$ $D_{III\mathcal{S}} \leftarrow D_{III\mathcal{S}} \cup (w, pk, M, t, \mathcal{R}_t)$ Return $\sigma$
Oracle $\mathcal{O}_{III\mathcal{R}}(t, \mathcal{R}_t)$ $RL_t \leftarrow \text{Revocation}(\mathcal{Y}, sk_o, t, \mathcal{R}_t)$ Return $RL_t$	
Experiment $\text{Exp}_{\mathcal{A}_{III}}^{\text{eu-cma}}(\lambda, \ell)$ $(\mathcal{Y}, pk_o, sk_o, \{pk_i, sk_i\}_{i=1}^{ \mathcal{N} }) \leftarrow \text{Setup}(1^\lambda, 1^\ell); D_{III\mathcal{D}}, D_{III\mathcal{S}} \leftarrow \emptyset$ $I_i \leftarrow \mathcal{A}_{III}^{\mathcal{O}_{III\mathcal{D}}}(w_i, pk_i); RL_{t_i} \leftarrow \mathcal{A}_{III}^{\mathcal{O}_{III\mathcal{R}}}(t_i, \mathcal{R}_{t_i})$ $\sigma_i \leftarrow \mathcal{A}_{III}^{\mathcal{O}_{III\mathcal{S}}}(w_i, pk_i, M_i, t_i, \mathcal{R}_{t_i})$ $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}, \sigma^*) \leftarrow \mathcal{A}_{III}(\mathcal{Y}, pk_o, \{pk_i, sk_i\}_{i=1}^{ \mathcal{N} }, \mathcal{O}_{III\mathcal{D}}, \mathcal{O}_{III\mathcal{R}}, \mathcal{O}_{III\mathcal{S}})$ If $\text{Verify}(\mathcal{Y}, t^*, pk^*, pk_o, M^*, \sigma^*) = 1$ , $pk^* \in \{pk_i\}_{i=1}^{ \mathcal{N} }$ , $(w^*, pk^*) \notin D_{III\mathcal{D}}$ , and $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}) \notin D_{III\mathcal{S}}$ return 1 else return 0	

**Security Model for Adversary  $\mathcal{A}_{IV}$ .** Adversary  $\mathcal{A}_{IV}$  represents proxy signers, who want to generate the proxy signature when they have been revoked. The security model is defined using the following game:

**Setup:** The challenger generates  $|\mathcal{N}| + 1$  public key and secret key pairs and assigns them to the original signer and proxy signers. Then it gives the adversary the system parameter  $\mathcal{Y}$ , the public keys of original signer  $pk_o$  and proxy signers  $\{pk_i\}_{i=1}^{|\mathcal{N}|}$ , secret keys of proxy signers  $\{sk_i\}_{i=1}^{|\mathcal{N}|}$ , and keeps the secret key of original signer  $sk_o$  to itself.

**Delegation Query** ( $\mathcal{O}_{\mathcal{IV}_D}$ ): The adversary issues up to  $q_D$  delegation queries. To each  $(w_i, pk_i)$ , the challenger responds by running *Delegation* algorithm to gain the delegated information  $I_i$  and the challenger sends  $I_i$  to the adversary. These queries may be asked adaptively.

**Revocation Query** ( $\mathcal{O}_{\mathcal{IV}_R}$ ): The adversary issues up to  $q_R$  revocation queries  $(t_i, \mathcal{R}_{t_i})$ . To each query, the challenger responds by executing *Revocation* algorithm to get the revocation list  $RL_{t_i}$  for revocation epoch  $t_i$ . Then the challenger sends  $RL_{t_i}$  to the adversary. These queries may be asked adaptively. A database  $D_{\mathcal{IV}_R}$  records all the queries. Notice that we assume there is only one  $\mathcal{R}_{t_i}$  for each  $t_i$ .

**Signing Query** ( $\mathcal{O}_{\mathcal{IV}_S}$ ): The adversary sends up to  $q_S$  signing queries to the challenger. For each  $(w_i, pk_i, M_i, t_i, \mathcal{R}_{t_i})$  where  $pk_i \notin \mathcal{R}_{t_i}$ , the challenger gains the delegated information  $I_i$  by running *Delegation* algorithm, runs the *Revocation* algorithm to get the revocation list  $RL_{t_i}$ , and executes the *Sign* algorithm to acquire the proxy signature  $\sigma_i$ . These queries may be asked adaptively. A database  $D_{\mathcal{IV}_S}$  records all the signing queries.

**Output:** Finally, the adversary outputs  $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}, \sigma^*)$ . The adversary wins if  $pk^* \in \mathcal{R}_{t^*}$ ,  $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*})$  has not been queried to *Sign* oracle, and  $(\mathcal{Y}, t^*, pk^*, pk_o, M^*, \sigma^*)$  can pass the verification.

**Definition 7.** A proxy signature scheme is  $(t, q_D, q_R, q_S, \epsilon)$ -strongly existentially unforgeable under an adaptive chosen message attack if no  $t$ -time adversary  $\mathcal{A}_{\mathcal{IV}}$  making at most  $q_D$  delegation queries,  $q_R$  revocation queries and  $q_S$  signing queries has advantage at least  $\epsilon$  in the above game. For any PPT adversary  $\mathcal{A}_{\mathcal{IV}}$  involved in the experiment hereafter, we have  $\text{Adv}_{\mathcal{A}_{\mathcal{IV}}}^{\text{eu-cma}}(\lambda) = \Pr[\text{Expt}_{\mathcal{A}_{\mathcal{IV}}}^{\text{eu-cma}}(\lambda) = 1] \in \text{negl}(\lambda)$ .

Oracle $\mathcal{O}_{\mathcal{IV}_D}(w, pk)$ $I \leftarrow \text{Delegation}(w, \mathcal{Y}, pk, pk_o, sk_o)$ Return $I$	Oracle $\mathcal{O}_{\mathcal{IV}_S}(w, pk, M, t, \mathcal{R}_t)$ $I \leftarrow \text{Delegation}(\mathcal{Y}, w, pk, pk_o, sk_o)$ $RL_t \leftarrow \text{Revocation}(\mathcal{Y}, sk_o, t, \mathcal{R}_t)$ $\sigma \leftarrow \text{Sign}(\mathcal{Y}, RL_t, sk, I, M)$ $D_{\mathcal{IV}_S} \leftarrow D_{\mathcal{IV}_S} \cup (w, pk, M, t, \mathcal{R}_t)$ Return $\sigma$
Oracle $\mathcal{O}_{\mathcal{IV}_R}(t, \mathcal{R}_t)$ $RL_t \leftarrow \text{Revocation}(\mathcal{Y}, sk_o, t, \mathcal{R}_t)$ $D_{\mathcal{IV}_R} \leftarrow D_{\mathcal{IV}_R} \cup (t, \mathcal{R}_t)$ Return $RL_t$	
Experiment $\text{Exp}_{\mathcal{A}_{\mathcal{IV}}}^{\text{eu-cma}}(\lambda, \ell)$ $(\mathcal{Y}, pk_o, sk_o, \{pk_i, sk_i\}_{i=1}^{ \mathcal{N} }) \leftarrow \text{Setup}(1^\lambda, 1^\ell); D_{\mathcal{IV}_R}, D_{\mathcal{IV}_S} \leftarrow \emptyset$ $I_i \leftarrow \mathcal{A}_{\mathcal{IV}}^{\mathcal{O}_{\mathcal{IV}_D}}(w_i, pk_i)$ $RL_{t_i} \leftarrow \mathcal{A}_{\mathcal{IV}}^{\mathcal{O}_{\mathcal{IV}_R}}(t_i, \mathcal{R}_{t_i})$ $\sigma_i \leftarrow \mathcal{A}_{\mathcal{IV}}^{\mathcal{O}_{\mathcal{IV}_S}}(w_i, pk_i, M_i, t_i, \mathcal{R}_{t_i})$ $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}, \sigma^*) \leftarrow \mathcal{A}_{\mathcal{IV}}(\mathcal{Y}, pk_o, \{pk_i, sk_i\}_{i=1}^{ \mathcal{N} }, \mathcal{O}_{\mathcal{IV}_D}, \mathcal{O}_{\mathcal{IV}_R}, \mathcal{O}_{\mathcal{IV}_S})$ If $\text{Verify}(\mathcal{Y}, t^*, pk^*, pk_o, M^*, \sigma^*) = 1$ , $pk^* \in \mathcal{R}_{t^*}$ , $(w^*, pk^*, M^*, t^*, \mathcal{R}_{t^*}) \notin D_{\mathcal{IV}_S}$ return 1 else return 0	

## 4 The Proposed Scheme

In this section, inspired by the BBG HIBE scheme [2], the NNL framework for broadcast encryption [18] and the BLS short signature [3], we will construct a hierarchical revocation scheme. Based on this revocation scheme, we will then build our proxy signature scheme with revocation.

### 4.1 Hierarchical Revocation Scheme

Our hierarchical revocation scheme consists of the following algorithms.

**Setup** $(1^\lambda, 1^\ell) \rightarrow (param, msk, mpk)$ :

- Set system parameter  $param = (e, \mathbb{G}, \mathbb{G}_T, g, p)$ .
- The original signer  $\mathcal{O}$  has  $(pk_o, sk_o) = (g^{x_o}, x_o)$  and each proxy signer  $\mathcal{P}_i$  has  $(pk_i, sk_i) = (g^{x_i}, x_i)$ , where  $x_i \in \mathbb{Z}_p^*$ . For each  $\mathcal{P}_i$ , assign a warrant  $w_i$ .
- Set master secret key  $msk = sk_o$  and master public key  $mpk = (pk_o, \{h_i\}_{i=0}^\ell)$ , where  $h_0, h_1, \dots, h_\ell \in \mathbb{G}$ .
- Select an injective function  $\mathcal{H} : \{0, 1\}^{\leq \ell} \rightarrow \mathbb{Z}_p^*$  and a hash function  $\mathcal{H}_1 : \{0, 1\}^* \rightarrow \mathbb{G}$ .

**Keygen** $(w_i, pk_i, msk, id) \rightarrow d_{id}$ :

$$\begin{aligned} d_{id} &= (D_1, D_2, K_2, \dots, K_{\ell-|id|+1}) \\ &= (\mathcal{H}_1(id, w_i, pk_i)^{sk_o} \cdot (h_0 \cdot h_1^{\mathcal{H}(id)})^r, g^r, h_2^r, \dots, h_{\ell-|id|+1}^r). \end{aligned}$$

**Derive** $(mpk, id, d_{id}, id' = (id, I_1, \dots, I_d)) \rightarrow d_{id'}$ :

$$\begin{aligned} d_{id'} &= (D'_1, D'_2) = (D_1 \cdot \prod_{i=1}^d K_{i+1}^{\mathcal{H}(I_i)}, D_2) \\ &= (\mathcal{H}_1(id, w_i, pk_i)^{sk_o} \cdot (h_0 \cdot h_1^{\mathcal{H}(id)} \cdot h_2^{\mathcal{H}(I_1)} \dots h_{d+1}^{\mathcal{H}(I_d)})^r, g^r). \end{aligned}$$

**Encode** $(mpk, id, id' = (id, I_1, \dots, I_d)) \rightarrow C$ :  $C = h_0 \cdot h_1^{\mathcal{H}(id)} \cdot h_2^{\mathcal{H}(I_1)} \dots h_{d+1}^{\mathcal{H}(I_d)}$ .

**Verify** $(mpk, w_i, pk_i, id, C, d_{id'}) \rightarrow \{0, 1\}$ : Parse  $d_{id'} = (D'_1, D'_2)$ , return 1 if following equation is true:  $e(g, D'_1) = e(pk_o, \mathcal{H}_1(id, w_i, pk_i)) \cdot e(C, D'_2)$ .

### 4.2 Proxy Signature with Revocation

Our proxy signature with revocation scheme consists of the following algorithms.

**Setup** $(1^\lambda, 1^\ell)$ :  $\lambda \in \mathbb{N}$  is a security parameter and  $N = 2^\ell$  is the maximum number of proxy signer. Generate a bilinear map  $(e, \mathbb{G}, \mathbb{G}_T, p, g)$ . Choose randomly  $\{h_i\}_{i=0}^\ell$  from  $\mathbb{G}$ . Choose an injective functions  $\mathcal{H} : \{0, 1\}^{\leq \ell} \rightarrow \mathbb{Z}_p^*$  and two hash functions  $\mathcal{H}_i : \{0, 1\}^* \rightarrow \mathbb{G}$  ( $i = 1, 2$ ). The system parameter  $\mathcal{Y} = ((e, \mathbb{G}, \mathbb{G}_T, p, g), \{h_i\}_{i=0}^\ell, \mathcal{H}, \mathcal{H}_1, \mathcal{H}_2)$ .

**Keygen**( $1^\lambda, \mathcal{Y}$ ): original signer  $\mathcal{O}$  and each proxy signer  $\mathcal{P}_i$  run the key generation algorithm to generate their own public key and secret key pair.  $\mathcal{O}$  generates  $(pk_o, sk_o) = (x_o, g^{x_o})$  and  $\mathcal{P}_i$  generates  $(pk_i, sk_i) = (x_i, g^{x_i})$ .

**Delegation**( $\mathcal{Y}, w_i, pk_i, pk_o, sk_o$ ):  $\mathcal{O}$  generates the delegated information  $I_i$  to  $\mathcal{P}_i$ .

- A warrant  $w_i$  is an explicit description of the delegation relation.
- $\mathcal{O}$  assigns to  $\mathcal{P}_i$  an availabel leaf  $v_i$  of label  $\langle v_i \rangle$ . Let  $x_0 = \epsilon, x_1, \dots, x_{\ell-1}, x_\ell = v_i$  be the path from the root  $\epsilon$  of  $\mathbb{T}$  to  $v_i$ . For  $j = 0$  to  $\ell$ ,  $\mathcal{O}$  does the following. Consider the sub-tree  $\mathbb{T}_{x_j}$  rooted at node  $x_j$ . Let  $\text{copath}_{x_j}$  be the co-path from  $x_j$  to  $v_i$ . For each node  $\omega \in \text{copath}_{x_j}$ , since  $x_j$  is an ancestor of  $\omega$ ,  $\langle x_j \rangle$  is a prefix of  $\langle \omega \rangle$  and we denote by  $\omega_{\ell_1 \dots \ell_2} \in \{0, 1\}^{\ell_2 - \ell_1 + 1}$ , for some  $\ell_1 \leq \ell_2 \leq \ell$ , the suffix of  $\langle \omega \rangle$  coming right after  $\langle x_j \rangle$ . Choose a random  $r \in \mathbb{Z}_p^*$  and compute

$$\begin{aligned} d_w &= (D_{\omega,1}, D_{\omega,2}, K_{\omega, \ell_2 - \ell_1 + 3}, \dots, K_{\omega, \ell}) \\ &= (\mathcal{H}_1(x_j, \omega_i, pk_i)^{sk_o} \cdot (h_0 \cdot h_1^{\mathcal{H}(\langle x_j \rangle)} \cdot h_2^{\mathcal{H}(\langle \omega_{\ell_1} \rangle)} \dots h_{\ell_2 - \ell_1 + 2}^{\mathcal{H}(\langle \omega_{\ell_2} \rangle)})^r, \\ &\quad g^r, h_{\ell_2 - \ell_1 + 3}^r, \dots, h_\ell^r). \end{aligned}$$

$\mathcal{P}_i$  gains the delegated information  $I_i = (w_i, \langle v_i \rangle, \{\{d_\omega\}_{\omega \in \text{copath}_{x_j}}\}_{j=0}^\ell)$ .

**Revocation**( $\mathcal{Y}, sk_o, t, \mathcal{R}_t$ ):

- Using the SD covering algorithm, find a cover of unrevoked user set  $\mathcal{N} \setminus \mathcal{R}_t$  as the union of disjoint subsets of the form  $S_{k_1, u_1}, \dots, S_{k_m, u_m}$ , with  $m \leq 2 \cdot |\mathcal{R}| - 1$ .
- For  $i = 1$  to  $m$ , do the following.
  1. Consider  $S_{k_i, u_i}$  as the difference between sub-trees rooted at an internal node  $x_{k_i}$  and one of its descendants  $x_{u_i}$ . The label of  $x_{u_i}$  can be written as  $\langle x_{u_i} \rangle = \langle x_{k_i} \rangle \| u_{i, \ell_{i,1}} \dots u_{i, \ell_{i,2}}$ . Then, compute an encoding of  $S_{k_i, u_i}$  as a group element:

$$C_i = h_0 \cdot h_1^{\mathcal{H}(\langle x_{k_i} \rangle)} \cdot h_2^{\mathcal{H}(u_{i, \ell_{i,1}})} \dots h_{\ell_{i,2} - \ell_{i,1} + 2}^{\mathcal{H}(u_{i, \ell_{i,2}})}.$$

2.  $\mathcal{O}$  generates a signature  $\Theta_i = \text{Sign}_{sk_o}(C_i, g^t) = \mathcal{H}_2(C_i, g^t)^{sk_o}$ .

Return the revocation list  $RL_t$  which is defined to be

$$RL_t = (t, \mathcal{R}_t, \{\langle x_{k_i} \rangle, \langle x_{u_i} \rangle, (C_i, \Theta_i)\}_{i=1}^m).$$

**Sign**( $\mathcal{Y}, RL_t, sk_i, I_i, M$ ): Proxy signer only needs to generate the HIBE decryption key once in each revocation epoch.

- Using  $RL_t$ , determine the set  $S_{k_l, u_l}$ , with  $l \in \{1, \dots, m\}$ , that contains the leaf  $v_i$  (this subset must exist since  $pk_i \notin \mathcal{R}_t$ ) and let  $x_{k_l}$  and  $x_{u_l}$  denote the primary and secondary roots of  $S_{k_l, u_l}$ . Since  $x_{k_l}$  is an ancestor of  $x_{u_l}$ , we can write  $\langle x_{u_l} \rangle = \langle x_{k_l} \rangle \| u_{l, \ell_1} \dots u_{l, \ell_2}$ , for some  $\ell_1 < \ell_2 < \ell$  and with  $u_{l, \kappa} \in \{0, 1\}$  for each  $\kappa \in \{\ell_1, \dots, \ell_2\}$ . The proxy signer  $\mathcal{P}_i$  computes an HIBE decryption key of the form

$$(D_{l,1}, D_{l,2}) = \left( (\mathcal{H}_1(x_{k_l}, w_i, pk_i)^{sk_o} \cdot (h_0 \cdot h_1^{\mathcal{H}(\langle x_{k_l} \rangle)} \cdot h_2^{\mathcal{H}(u_{l, \ell_1})} \dots h_{\ell_2 - \ell_1 + 2}^{\mathcal{H}(u_{l, \ell_2})})^r, g^r) \right).$$

Note that  $(D_{l,1}, D_{l,2})$  can be reused in whole revocation epoch.

– Compute  $\sigma_m = \text{Sign}_{sk_i}(M, \Omega)$  where  $\Omega = (w_i, x_{k_l}, x_{u_l}, D_{l,1}, D_{l,2}, C_l, \Theta_l)$ .

Return the proxy signature  $\sigma = (\Omega, \sigma_m)$ .

**Verify**( $\mathcal{Y}, t, pk_i, pk_o, M, \sigma$ ): Verifier checks the proxy signature.

1. Check  $\sigma_m$ : If  $\text{Verify}_{pk_i}((M, \Omega), \sigma_m) = 0$ , return 0.
2. Check  $\Theta_l$ : If  $\text{Verify}_{pk_o}((C_l, g^t), \Theta_l) = 0$ , return 0.
3. Check  $C_l$ : If  $e(g, D_{\ell_1}) = e(pk_o, \mathcal{H}_1(x_{k_l}, w_i, pk_i)) \cdot e(C_\ell, D_{\ell_2})$ , return 1. Otherwise, return 0.

## 5 Security Analysis

The proposed schemes is secure against Type-II/III/IV adversaries in the random oracle model. Please refer to the full version of the paper for the full proofs.

**Theorem 1.** *The hierarchical revocation scheme is key robust assuming that the CDH assumption holds in  $\mathbb{G}$ .*

**Theorem 2.** *The proxy signature with revocation scheme is secure against Type-II/III/IV adversaries.*

## 6 Conclusions

In this paper, we proposed a new solution for proxy signature with revocation. Compared with the previous approaches, our solution does not require any third party. In addition, the verifier does not need to access the latest revocation list in order to verify a proxy signature. We also built a novel hierarchical revocation scheme, which is of independent interest. We proved the security of the hierarchical revocation scheme and the proxy signature scheme with revocation against various types of adversaries.

**Acknowledgement.** The last author of this work is supported by the National Natural Science Foundation of China (No. 61402184).

## References

1. Ateniese, G., Song, D., Tsudik, G.: Quasi-efficient revocation of group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
2. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)

4. Das, M.L., Saxena, A., Gulati, V.P.: An efficient proxy signature scheme with revocation. *Informatica* **15**(4), 455–464 (2004)
5. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) *DRM 2002. LNCS*, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
6. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* **17**(2), 281–308 (1988)
7. Halevy, D., Shamir, A.: The LSD broadcast encryption scheme. In: Yung, M. (ed.) *CRYPTO 2002. LNCS*, vol. 2442, pp. 47–60. Springer, Heidelberg (2002)
8. Huang, X., Mu, Y., Susilo, W., Zhang, F., Chen, X.: A short proxy signature scheme: efficient authentication in the ubiquitous world. In: Enokido, T., Yan, L., Xiao, B., Kim, D.Y., Dai, Y.-S., Yang, L.T. (eds.) *EUC-WS 2005. LNCS*, vol. 3823, pp. 480–489. Springer, Heidelberg (2005)
9. Huang, X., Susilo, W., Mu, Y., Wu, W.: Proxy signature without random oracles. In: Cao, J., Stojmenovic, I., Jia, X., Das, S.K. (eds.) *MSN 2006. LNCS*, vol. 4325, pp. 473–484. Springer, Heidelberg (2006)
10. Kim, S., Park, S., Won, D.: Proxy signatures, revisited. In: *Information and Communications Security*, pp. 223–232 (1997)
11. Lee, B., Kim, H.-S., Kim, K.: Secure mobile agent using strong non-designated proxy signature. In: Varadharajan, V., Mu, Y. (eds.) *ACISP 2001. LNCS*, vol. 2119, pp. 474–486. Springer, Heidelberg (2001)
12. Lee, B., Kim, H., Kim, K.: Strong proxy signature and its applications. In: *Proceedings of SCIS*, vol. 1, pp. 603–608 (2001)
13. Li, X., Chen, K., Li, S.: Multi-proxy signature and proxy multi-signature schemes from bilinear pairings. In: Liew, K.-M., Shen, H., See, S., Cai, W. (eds.) *PDCAT 2004. LNCS*, vol. 3320, pp. 591–595. Springer, Heidelberg (2004)
14. Liu, Z., Yupu, H., Zhang, X., Ma, H.: Provably secure multi-proxy signature scheme with revocation in the standard model. *Comput. Commun.* **34**(3), 494–501 (2011)
15. Lu, E.J.-L., Hwang, M.-S., Huang, C.-J.: A new proxy signature scheme with revocation. *Appl. Math. Comput.* **161**(3), 799–806 (2005)
16. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures: delegation of the power to sign messages. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **79**(9), 1338–1354 (1996)
17. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pp. 48–57. ACM (1996)
18. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) *CRYPTO 2001. LNCS*, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
19. Seo, S.-H., Shim, K.-A., Lee, S.-H.: A mediated proxy signature scheme with fast revocation for electronic transactions. In: Katsikas, S.K., López, J., Pernul, G. (eds.) *TrustBus 2005. LNCS*, vol. 3592, pp. 216–225. Springer, Heidelberg (2005)
20. Sun, H.-M.: Design of time-stamped proxy signatures with traceable receivers. In: *IEE Proceedings-Computers and Digital Techniques*, vol. 147, no. 6, pp. 462–466 (2000)
21. Tzeng, S.-F., Yang, C.-Y., Hwang, M.-S.: A nonrepudiable threshold multi-proxy multi-signature scheme with shared verification. *Future Gener. Comput. Syst.* **20**(5), 887–893 (2004)
22. Yi, L., Bai, G., Xiao, G.: Proxy multi-signature scheme: a new type of proxy signature scheme. *Electron. Lett.* **36**(6), 527–528 (2000)



Information Security and Privacy

21st Australasian Conference, ACISP 2016, Melbourne,  
VIC, Australia, July 4-6, 2016, Proceedings, Part II

Liu, J.K.; Steinfeld, R. (Eds.)

2016, XVIII, 517 p. 46 illus., Softcover

ISBN: 978-3-319-40366-3