3-2018

A new revocable and re-delegable proxy signature and its application

Shengmin XU
Singapore Management University, smxu@smu.edu.sg

Guomin YANG

Yi MU

Xu S, Yang G, Mu Y. A new revocable and re-delegable proxy signature and its application. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(2): 380–399 Mar. 2018. DOI 10.1007/s11390-018-1825-4

A New Revocable and Re-Delegable Proxy Signature and Its Application

Shengmin Xu, Guomin Yang, *Member*, *IEEE*, and Yi Mu, *Senior Member*, *IEEE*

Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong Wollongong 2500, Australia

E-mail: sx914@uowmail.edu.au; {gyang, ymu}@uow.edu.au

Received January 9, 2017; revised November 28, 2017.

Abstract With the popularity of cloud computing and mobile Apps, on-demand services such as on-line music or audio streaming and vehicle booking are widely available nowadays. In order to allow efficient delivery and management of the services, for large-scale on-demand systems, there is usually a hierarchy where the service provider can delegate its service to a top-tier (e.g., countrywide) proxy who can then further delegate the service to lower level (e.g., region-wide) proxies. Secure (re-)delegation and revocation are among the most crucial factors for such systems. In this paper, we investigate the practical solutions for achieving re-delegation and revocation utilizing proxy signature. Although proxy signature has been extensively studied in the literature, no previous solution can achieve both properties. To fill the gap, we introduce the notion of revocable and re-delegable proxy signature that supports efficient revocation and allows a proxy signer to re-delegate its signing right to other proxy signers without the interaction with the original signer. We define the formal security models for this new primitive and present an efficient scheme that can achieve all the security properties. We also present a secure on-line revocable and re-delegate vehicle ordering system (RRVOS) as one of the applications of our proposed scheme.

Keywords revocation, (re)delegation, proxy signature

1 Introduction

Due to the popularity of cloud computing and smart mobile devices, on-demand services, such as real-time video and music services[1-2], are becoming more and more popular nowadays. In order to provide better (e.g., customized) services to customers located at different regions, the service provider usually will delegate the right to provide the service to some local companies who are either subsidiaries or proxies that are authorized by the service provider. Moreover, there may exist a hierarchy in the delegation list, e.g., a country-wide delegatee can further delegate the right to regional delegatees. Such a multi-level delegation structure is more efficient and practical for very large international on-demand service applications.

In order to ensure that only the legitimate delegatees can provide the service (e.g., due to copyright or service charge related issues), we should allow the end users to efficiently verify whether the service provided by a company is genuine or not. On the other hand, it is possible that a legitimate delegatee may abuse the delegated right (e.g., providing unauthorized services) or refuse to pay the subscription fee. In such circumstances, the service provider or an upper-tier (e.g., country-wide) proxy should be able to revoke a misbehaving regional delegatee.

Proxy signature[3-4] provides a solution for signing right delegation and hence allows the service provider to authorize the right of providing the prescribed services to legitimate delegatees (i.e., proxy signers). One fundamental security requirement of proxy signature is that without the delegation, a proxy signer cannot produce a valid proxy signature. Hence, end users can easily verify the legitimacy of a service by verifying the proxy signature generated by the proxy signer. However, traditional proxy signature schemes, which will be reviewed shortly in Subsection 1.1, cannot achieve both re-delegation and efficient revocation simultaneously, which is the problem we aim to solve in this paper.

Regular Paper

1.1 Proxy Signature

Mambo *et al.*[3-4] introduced the first proxy signature scheme in 1996. A proxy signature scheme consists of three entities: original signer, proxy signer, and verifier. An original signer can generate a proxy signing key, which will be sent to a proxy signer. The proxy signer can then use this proxy signing key to sign messages on behalf of the original signer.

The delegation in a proxy signature scheme can be classified into four types. The seminal work of Mambo *et al.*[3] proposed three of them: full delegation, partial delegation, and delegation by warrant. In the full delegation, the original signer just gives its secret key to the proxy signer as the proxy signing key. Hence, the proxy signer and the original signer have the same signing ability and there is no non-repudiation. To conquer this drawback, partial delegation was also proposed. In the partial delegation, the proxy signing key is derived from the private key of the original signer and the public or secret key of the proxy signer. There are two approaches to realize partial delegation, namely, the proxy-unprotected and the proxy-protected delegation[5]. In the case of the proxy-unprotected partial delegation, the proxy signing key combines the secret key of the original signer and the public key of the proxy signer. Thus, the original signer can derive the proxy signing key without interacting with the proxy signer, but the proxy signer cannot obtain the proxy signing key without the help from the original signer, which leads to the problem that the original signer can sign messages on behalf of the proxy signer. To overcome the problem in the proxy-unprotected partial delegation, the proxy-protected partial delegation was proposed, which requires the generation of the proxy signing key from the secret keys of both the original signer and the proxy signer. However, the partial delegable proxy signature still suffers the problem that the proxy signer has unlimited signing ability. To overcome this drawback, the concept of delegation by warrant was proposed. In the delegation by warrant, the original signer signs a warrant which certifies the legitimacy of the proxy signer. To combine the advantages of the partial delegation and the delegation by warrant, Kim *et al.*[6] introduced a novel type of proxy delegation called partial delegation with warrant. In recent decades, many studies on proxy signature have been proposed based on the RSA[5,7] or Diffie-Hellman[8-10] assumptions.

Proxy signature schemes can also be categorized into proxy multi-signature, multi-proxy signature, and proxy re-delegation schemes. In a proxy multi-signature scheme[11-12], a designated proxy signer can generate a proxy signature on behalf of two or more original signers. A multi-proxy signature scheme[7,13] allows a group of original signers to delegate the signing capability to a designated group of proxy signers. In the case of proxy re-delegation[14], it allows the proxy signers to delegate the signing right to other proxy signers on behalf of the original signer.

1.2 Motivation

In this paper, we focus on designing proxy signature with both proxy revocation and proxy re-delegation that are important for many applications mentioned above. Although there are many research studies on proxy signature, only few of them deal with revocation or re-delegation. Moreover, there is no proxy scheme in the literature that can achieve both properties.

Proxy revocation is a critical issue when the proxy signer is compromised. Furthermore, in reality, the proxy signer may also misuse the delegated signing right. In such situations, the original signer should be able to revoke the proxy signing key delegated to the proxy signer even before the delegation expires. One straightforward solution to address this problem is to let the original signer publish a white list or a black list, and a verifier needs to check the list before verifying a proxy signature. However, this solution is not practical since the verifier needs to keep on updating the white or black list before verifying a proxy signature. Another limitation of such an approach is that the proxy signatures generated before the revocation also become invalid. Ideally, such proxy signatures should still be considered valid since the proxy signer is not revoked when the signature was generated.

One solution proposed in the literature to address the revocation problem is utilizing the time-stamp[13]. However, the proposed scheme has some security issues. As pointed out in [15], an attacker can easily forge a proxy signature.

Another solution for proxy revocation is to use a third party. Seo *et al.*[16] and Liu *et al.*[17] proposed to use a third party called security mediator (SEM) which is a semi-trusted on-line server. The original signer divides the proxy delegation into two parts and gives these two parts to the proxy signer and SEM, respectively. When the proxy signer wants to generate a proxy signature, it needs the assistance from SEM which works as a certifier to authenticate the signing ability of every proxy signer. Similarly, Das *et al.*[15]

and Lu *et al.*[18] proposed some revocable proxy signature schemes where a trusted third party called the authentication server (AS) is used to provide the immediate revocation. Such solutions are impractical either since whenever the proxy signer wants to generate a proxy signature, it must contact the third party (SEM or AS) which is a bottleneck of these systems.

Proxy re-delegation[19] is a useful property in proxy signature, when a proxy signer wishes to re-delegate a subset of their signing rights to other users. For example, in the applications of on-demand service or software manufacturing, the software or service provider can delegate the right to sell the service or software to country-wide proxies who can then further delegate the right to lower-level proxies.

In a re-delegable proxy signature, the proxy signer can be further classified into the mediate proxy signer (re-delegator) and the end-node proxy signer (delegatee). A mediate proxy signer is able to generate proxy signing keys for other proxy signers, while an end-node proxy signer does not generate proxy signing keys for the others. The list of signers from the original signer to the delegatee is called a delegation chain.

1.3 Our Result

In this paper, we introduce a novel revocable and re-delegable proxy signature scheme. Compared with the previous related work, our scheme has the following advantages.

• Our scheme is more practical than the previous solutions since our scheme does not need any third party and provides both proxy revocation and proxy re-delegation. Also, the verifier does not need to obtain the revocation list to verify a proxy signature. Instead, it only needs to know the current revocation epoch to verify a proxy signature.

• Our scheme achieves efficient revocation. The original signer can revoke a set of proxy signers in each revocation epoch. A non-revoked proxy signer only needs to generate once in each revocation epoch a proof which shows its validity.

• Our scheme explicitly includes the revocation epoch in signature verification. Hence, the verifier only rejects the signatures generated by a proxy signer after its proxy signing right is revoked. The signatures generated before revocation will remain valid.

As an application of our proxy signature scheme, we present a secure vehicle hiring protocol that allows end users to safely book vehicles that are authorized by a trusted authority or its legitimate proxies. Some companies, for example, Uber, Grab, and Lyft, provide applications in mobile devices that allow customers to book vehicles online. There are some existing studies[20-23] addressing the revocation or delegation problem in vehicular ad hoc networks (VANETs). However, there is no existing solution addressing both problems. In this work, we introduce a solution to address both problems and also the concept and technique for allowing re-delegation that makes our solution more scalable and practical.

1.4 Differences with [24]

An extended abstract of this paper appears in the Proceeding of ACISP 2016[24], where we introduced a hierarchical revocation scheme and a proxy signature scheme with efficient revocation. In this paper, we extend the original proxy signature to achieve both revocation and re-delegation. To realize this goal, we extend and modify the models and the scheme in [24].

We extend the original scheme to realize revocable and re-delegable proxy signature Γ = (Setup, KeyGen, Delegation, Revocation, Sign, Verify). In Γ.Delegation, we append some extra information to determine the relationship between the delegator and the delegatee, which allows the delegatee to gain a delegation chain to prove the validity of the signer. Similarly, in Γ.Revocation, we allow every mediator proxy signer to generate a revocation list to revoke its delegatee(s). Γ.Sign and Γ.Verify are also modified to handle the delegation chain compared to the original revocable proxy signature scheme.

We also modify the security models in the original paper[24] to define stronger security. In each model, we define a new oracle called corrupt oracle, which allows the adversary to corrupt the original, mediate signers to obtain their secret keys. Due to the modification of the security models and the extra requirement of proxy re-delegation, we also rewrite the security proofs.

1.5 Outline of the Paper

The rest of this paper is organized as follows. Some preliminaries are presented in Section 2. The formal definition and security models for our scheme are described in Section 3. The proposed proxy signature scheme and its security proof are given in Section 4. We then present a secure vehicle ordering system as an application of our proposed scheme in Section 5. Finally, some concluding remarks are given in Section 6.

2 Preliminaries

In this section, we will briefly review some basic backgrounds used in this paper, including pairings, complexity assumptions, and some other schemes to construct our revocable and re-delegable proxy signature.

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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ as above.

2.2 Complexity Assumptions

Our scheme relies on the classical computational Diffie-Hellman (CDH) problem, whose details are described as follows.

Definition 1 (CDH Problem). *Let $\mathcal{G}$ be a group with a generator g. The computational Diffie-Hellman problem is as follows: given (g, g^a, g^b), for random $a, b \in \mathbb{Z}_p^*$, then compute g^{ab}. We say algorithm $\mathcal{A}$ has advantage ϵ in solving this problem if*

$$\mathsf{Adv}_{\mathcal{A}}^{cdh}(\lambda) = \Pr(g^{ab} \leftarrow \mathcal{A}(g, g^a, g^b) : a, b \xleftarrow{\mathrm{R}} \mathbb{Z}_p^*) \geqslant \epsilon.$$

Definition 2 (CDH Assumption). *We say it satisfies the computational Diffie-Hellman assumption if for any polynomial-time algorithm, the advantage in solving the CDH problem is negligible.*

2.3 Revocation Mechanism

Naor *et al.*[25] introduced a subset cover framework for broadcast encryption. This framework is based on the complete subtree (CS) method and the subset difference (SD) method. Halevy and Shamir[26] proposed a new method called layered subset difference (LSD) to improve the key distribution in the SD method. Later, Dodis and Fazio[27] pointed out that HIBE schemes can be based on the above methods. In this subsection, we will briefly introduce the SD method.

The SD method as shown in Fig.1 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 $S_{k_1,u_1}, ..., S_{k_m,u_m}$. Each valid user belongs to at least one set, and the number of sets m satisfies $m \leqslant 2|\mathcal{R}| - 1$. Let T_{x_j} denote the subtree rooted at x_j.

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.

2.4 Digital Signature Scheme

A digital signature scheme[28] is a triple of probabilistic polynominal time (PPT) algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

- $\Sigma.\mathsf{KeyGen}(1^\lambda) \to (pk, sk)$. It inputs a security parameter λ and outputs in PPT a pair (pk, sk) of matching the public and secret key.
- $\Sigma.\mathsf{Sign}_{sk}(m) \to \sigma$. It produces a signature σ for a message m using the secret key sk.
- $\Sigma.\mathsf{Verify}_{pk}(m, \sigma) \to [0, 1]$. It tests whether σ is a valid signature for message m using the public key pk.

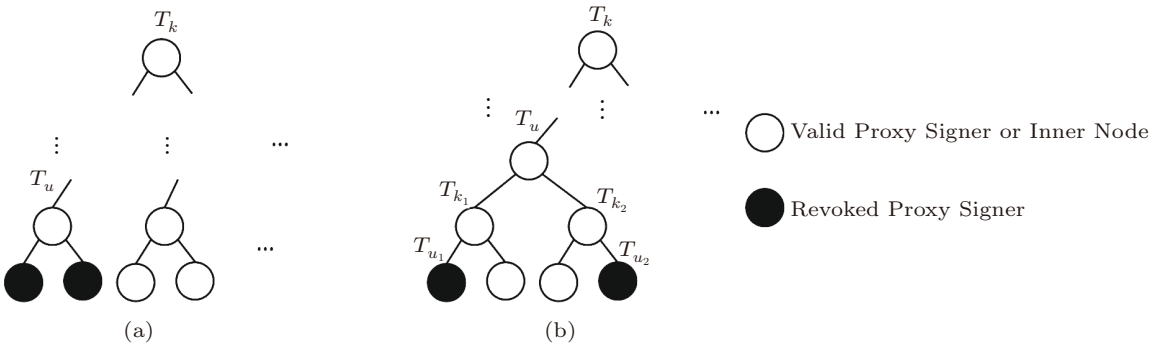

Fig.1. SD method. (a) All revoked proxy signers under a subtree. (b) Revoked proxy signer under different subtrees.

The algorithm outputs either 1 (valid) or 0 (invalid).

2.5 Security Model for Existential Unforgeability

We assume the digital signature scheme σ is existential unforgeability under a chosen-message attack. Thus, given public key pk, the adversary cannot forge any message without knowing the secret key sk. Therefore, the hard problem is given pk, the adversary cannot forge any message under this public key. Existential unforgeability[29] under an adaptive chosen-message attack is defined using the following game.

• *Setup.* The challenger runs $\Sigma.\mathsf{KeyGen}$. It gives the adversary the resulting public key pk and keeps the private key sk to itself.

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

• *Output.* Finally the adversary outputs a pair (m^*, s^*). The adversary wins if s^* is a valid signature of m^* according to $\Sigma.\mathsf{Verify}$ and m^* is not among the pairs m_i generated during the query phase.

Definition 3. *A signature scheme is (t, q, ϵ) existentially unforgeable under an adaptive chosen-message attack if no t-time adversary $\mathcal{A}_{\mathcal{EU}}$ making at most q signing queries has advantage at least ϵ in the above game. For any PPT adversary $\mathcal{A}_{\mathcal{EU}}$ involved in the experiment in Fig.2, we have* $\mathsf{Adv}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathcal{EU}}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathcal{EU}}}(\lambda) = 1) \in \mathsf{negl}(\lambda)$, *where* $\mathsf{negl}(\lambda)$ *denotes the set of negligible functions.*

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

Fig.2. Security model for EU-CMA (extensional unforgeability under choose message attack).

The Boneh-Lynn-Shacham (BLS) signature scheme was proposed in [30], which is a widely used digital signature scheme. It has been proven secure against adaptive chosen-message attacks in the random oracle model assuming that the CDH problem is hard. We use BLS short signature as the building block to realize our revocable and re-delegable proxy signature.

2.6 Hierarchical Revocation Scheme

Xu *et al.*[24] introduced a novel tree-based hierarchical revocation scheme which is derived from Boneh-Boyen-Goh hierarchical identity-based encryption scheme[31] and the subset difference method in the Naor-Naor-Lotspiech framework for the broadcast encryption[25]. The revocation is efficient due to the tree-based hierarchical structure. This scheme keeps a white list to reject all the revoked proxy signers, and the size of this revocation list is $O(|\mathcal{R}|)$, where $|\mathcal{R}|$ is the number of the revoked user. The hierarchical revocation scheme $\Delta = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Derive}, \mathsf{Encode}, \mathsf{Verify})$ consists of the following PPT algorithms.

$\Delta.\mathsf{Setup}(1^\lambda, 1^\ell) \rightarrow (pp, pk_o, sk_o)$: given a security parameter λ and a maximum level ℓ of the full binary tree, it outputs the public parameter pp, the public key pk_o, and the secret key sk_o of the original signer $\mathcal{O}$.

• Generate parameters for bilinear map

$$(e, \mathbb{G}, \mathbb{G}_T, g, p),$$

and randomly choose parameters

$$\{h_i\}_{i=0}^{\ell}, \text{where } h_0, h_1, ..., h_\ell \in \mathbb{G}.$$

• Run $(pk_o, sk_o) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$ to generate the public key and secret key pair for the original signer $\mathcal{O}$①. Our proposed scheme utilizes the BLS short signature[30] as building block, thus $pk_o = g^{x_o}$ and $sk_o = x_o$, where x_o is a random value in $\mathbb{Z}_p$. For notational simplicity, unless otherwise specified, we will hereafter refer to a key pair (pk, sk) as a key pair of the BLS short signature scheme.

• Select an injective function $\mathcal{H}$ and a hash function $\mathcal{H}_1$.

$$\mathcal{H} : \{0,1\}^{\leqslant \ell} \rightarrow \mathbb{Z}_p^*, \mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}.$$

• Set public parameter pp as

$$(e, \mathbb{G}, \mathbb{G}_T, g, p, \{h_i\}_{i=0}^{\ell}, \mathcal{H}, \mathcal{H}_1).$$

$\Delta.\mathsf{ProxyKeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$: given a security parameter λ, it outputs the public and secret key pair $(pk_i, sk_i) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$ for the proxy signer $\mathcal{P}_i$.

①Following [24], we assume the system is set up by the original signer. However, we should note that we can also put the key generation of the original signer as a separate function.

$\Delta.\mathsf{KeyGen}(pk_o, sk_o, pk_i, w_i, id) \rightarrow d_{id}$: given a public key pk_o and a secret key sk_o of the original signer $\mathcal{O}$, a proxy signer $\mathcal{P}_i$'s public key pk_i, a warrant w_i and a label value id in the hierarchy, it outputs a hierarchical private key d_{id}, where the private key d_{id} includes the decryption key (D_1, D_2) and the delegation key $(K_2, ..., K_{\ell-|id|+1})$.

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

$\Delta.\mathsf{Derive}(pk_o, id, d_{id}, id') \rightarrow d_{id'}$: given a public key pk_o of the original signer $\mathcal{O}$, a label value id and its hierarchical private key d_{id} and a label value id', where id' is a descendant of id in the hierarchy, i.e., $id' = id \| I_1, ..., I_d$, it outputs another hierarchical private key $d_{id'}$ for id'②.

$$\begin{aligned} &d_{id'} = (D_1', D_2'),\\ &D_1' = D_1 \times \prod_{i=1}^{d} K_{i+1}^{\mathcal{H}(I_i)},\\ &D_2' = D_2. \end{aligned}$$

$\Delta.\mathsf{Encode}(id, id') \rightarrow C$: given a label value id and another label value id' which is a descendant of id, it outputs an encoding value C.

$$C = h_0 \times h_1^{\mathcal{H}(id)} \times h_2^{\mathcal{H}(I_1)} \times \cdots \times h_{d+1}^{\mathcal{H}(I_d)}.$$

$\Delta.\mathsf{Verify}(pk_o, pk_i, w_i, id, C, d_{id'}) \rightarrow [0, 1]$: given a public key pk_o of the original signer $\mathcal{O}$, a proxy signer $\mathcal{P}_i$'s public key pk_i and its warrant w_i, a label value id, an encoding value C (with regard to id and id') and a hierarchical private key $d_{id'}$, it outputs 1 (valid) if the following equation holds.

$$e(g, D_1') = e(pk_o, \mathcal{H}_1(id, w_i, pk_i)) \times e(C, D_2').$$

Otherwise, it returns 0.

This scheme has been proven to be key robust in the random oracle model[24] assuming the CDH assumption is hard. The details of the security model are described below.

Security Model for Hierarchical Revocation Algorithm. The security model for hierarchical revocation algorithm is called key robustness. 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 ($\mathcal{O}_{\mathcal{A}_\mathcal{G}}$). 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), ..., $(id_{i-1}, w_{i-1}, pk_{i-1})$. A database $D_{\mathcal{A}_\mathcal{G}}$ records all the messages that have been queried.

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

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

Hash Query ($\mathcal{O}_{\mathcal{A}_\mathcal{H}}$). 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, ϵ) 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 ϵ in the above game. For any PPT adversary $\mathcal{A}$ involved in the experiment in Fig.3, we have* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{key\text{-}robust}}(\lambda) = \Pr(\mathsf{Exp}_{\mathcal{A}}^{\mathsf{key\text{-}robust}}(\lambda, \ell) = 1) \in \mathsf{negl}(\lambda)$.

3 Formal Definitions and Security Models

In this section, we will demonstrate the syntax of our revocable and re-delegable proxy signature scheme and its formal security models. Here, we provide the details of some notations, and they will be used in this section.

- $\mathcal{N}$ is the set of the proxy signers, and $|\mathcal{N}|$ is the number of the proxy signers.
- $\mathcal{R}$ is the set of the revoked proxy signers, and $|\mathcal{R}|$ is the number of the revoked proxy signers.
- $\mathcal{R}_t$ is the set of the revoked proxy signers under the revocation epoch t.
- $\ell \in \mathbb{Z}$ is the maximum level of the tree and $|\mathcal{N}| = 2^\ell$.

②The output $d_{id'}$ of $\Delta.\mathsf{Derive}$ also belongs to the owner of d_{id}, and thus he/she does not need to add the delegation key in $d_{id'}$ (i.e., the delegation key produced in $\Delta.\mathsf{KeyGen}$ is for allowing the generation of $d_{id'}$).

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

Fig.3. Security model of key robustness.

• $w \in \mathbb{Z}$ is the warrant value for the signing right delegation.

• $\mathcal{P}_{\mathsf{a}_i}$ stands for the immediate ancestor of the proxy signer $\mathcal{P}_i$, which means the proxy signer $\mathcal{P}_{\mathsf{a}_i}$ is the proxy re-delegator who generates the proxy signing key for $\mathcal{P}_i$.

• $\mathcal{P}_{\mathsf{m}_i}$ represents a set of all previous delegators of the proxy signer $\mathcal{P}_i$ including the original signer $\mathcal{O}$.

3.1 Revocable and Re-Delegable Proxy Signature

Our revocable and re-delegable proxy signature scheme allows the original signer and all mediate proxy signers to revoke their descendant proxy signers. A revocable and re-delegable proxy signature has the following six PPT algorithms $\Gamma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Delegate}, \mathsf{Revoke}, \mathsf{Sign}, \mathsf{Verify})$.

$\Gamma.\mathsf{Setup}(1^\lambda, 1^\ell) \rightarrow (pp, pk_o, sk_o)$. Given a system security parameter λ 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 public parameter pp, the public key pk_o, and the secret key sk_o of the original signer $\mathcal{O}$.

$\Gamma.\mathsf{KeyGen}(1^\lambda) \rightarrow (pk_i, sk_i)$. Given a system security parameter λ, it outputs a pair of public and secret key (pk_i, sk_i) for the proxy signer $\mathcal{P}_i$.

$\Gamma.\mathsf{Delegation}(pk_i, pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, w_i, T_{\mathsf{a}_i}) \rightarrow T_i$. Given the public key pk_i of the proxy signer $\mathcal{P}_i$, the public key pk_{a_i} and the secret key sk_{a_i} of the proxy signer $\mathcal{P}_i$'s immediate proxy ancestor, a warrant w_i for the proxy signer $\mathcal{P}_i$, and the delegated tag of its immediate proxy ancestor T_{a_i}, it outputs the delegated tag T_i for the proxy signer $\mathcal{P}_i$.

$\Gamma.\mathsf{Revocation}(pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, t, \mathcal{R}_t) \rightarrow RL_t$. Given the public key pk_{a_i} and the secret key sk_{a_i} of the mediate proxy signer $\mathcal{P}_{\mathsf{a}_i}$, 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 for its descendant proxy signers.

$\Gamma.\mathsf{Sign}(sk_i, T_i, RL_t, M) \rightarrow \sigma$. Given the secret key sk_i and the delegated tag T_i of the proxy signer $\mathcal{P}_i$, the revocation list RL_t under revocation epoch t and a message M, it outputs a proxy signature σ.

$\Gamma.\mathsf{Verify}(pk_i, pk_o, t, M, \sigma) \rightarrow [0, 1]$. Given the public key pk_i of the proxy signer $\mathcal{P}_i$, the public key pk_o of the original signer $\mathcal{O}$, the revocation epoch t, the message M, and the proxy signature σ, it outputs either 1 (valid) or 0 (invalid).

3.2 Security Models for Revocable and Re-Delegable Proxy Signature

To define the unforgeability of our revocable and re-delegable proxy signature scheme, according to the classification of Huang *et al.*[32], their continuing work[33] and the work of Xu *et al.*[24], we divide the adversaries into the following four types.

Type I. This type of adversary $\mathcal{A}_\mathrm{I}$ represents the normal adversary who has the public parameter pp, the public key of the original signer pk_o, and public keys of all proxy signers $\{pk_i\}_{i=1}^{|\mathcal{N}|}$.

Type II. This type of adversary $\mathcal{A}_\mathrm{II}$ stands for any mediate proxy signer including the original signer who has the public parameter pp, the public key of the original signer pk_o, public keys of all proxy signers $\{pk_i\}_{i=1}^{|\mathcal{N}|}$, and a corrupt oracle to corrupt any mediate proxy signer including the original signer and some proxy signers. The goal of adversary is to forge a valid proxy signature w.r.t. an uncorrupted proxy signer.

Type III. This type of adversary $\mathcal{A}_\mathrm{III}$ represents the proxy signer who has the public parameter pp, the public key of the original signer pk_o, public keys of all proxy signers $\{pk_i\}_{i=1}^{|\mathcal{N}|}$, and a corrupt oracle to corrupt any proxy signer and some mediate proxy signers including the original signer. The goal of adversary is to forge a valid proxy signature without a valid delegation.

Type IV. This type of adversary $\mathcal{A}_\mathrm{IV}$ stands for a revoked proxy signer who has the public parameter pp,

the public key of the original signer pk_o, public keys of all proxy signers $\{pk_i\}_{i=1}^{|\mathcal{N}|}$, and a corrupt oracle to corrupt any proxy signer and some mediate proxy signers including the original signer. The goal of adversary is to forge a valid proxy signature after it has been revoked.

One can find that if our revocable and re-delegable scheme is secure against type II (or type III or type IV) adversary, our scheme is also secure against type I adversary. Below we give the formal security models. 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.

1) *Security Model for Adversary* $\mathcal{A}_{\mathrm{II}}$. $\mathcal{A}_{\mathrm{II}}$ represents the original signer or any mediate proxy signer who wants to generate a valid proxy signature for any proxy signer without knowing its secret key. The security model is defined using the following game.

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

Corrupt Query ($\mathcal{O}_{\mathrm{II_C}}$). The adversary issues up to q_C corrupt queries. Upon receiving a public key pk corresponding to any mediate proxy signer including the original signer or some proxy signers, the challenger reveals the matching secret key sk of the public key pk. These queries may be asked adaptively so that each query may depend on the replies to all previous queries. A database $\mathcal{D}_{\mathrm{II_C}}$ records all the queried messages.

Signing Query ($\mathcal{O}_{\mathrm{II_S}}$). The adversary issues up to q_S signing queries $(w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$, where $pk \notin \mathcal{R}_t$. The challenger responds by running $T \leftarrow \Gamma.\mathsf{Delegation}(pk_i, pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, w_i, T_{\mathsf{a}_i})$ to obtain the delegated tag T, the revocation algorithm $RL \leftarrow \Gamma.\ \mathsf{Revocation}(pk_{\mathsf{a}_i},\ sk_{\mathsf{a}_i}, t, \mathcal{R}_t)$ to gain the revocation list RL_t for all proxy mediators, and $\sigma \leftarrow \Gamma.\mathsf{Sign}(sk, T, RL_t, M)$ to obtain the proxy signature σ. After that, the challenger sends σ to the adversary. These queries may be asked adaptively so that each query may depend on the replies to all previous queries. A database $D_{\mathrm{II_S}}$ records all the information of queries.

Output. Finally, the adversary outputs $(w^*,\ pk^*,\ M^*,\ t^*,\ \{pk_i^*,\ \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*)$. The adversary wins if pk^* has not been corrupted, $(pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}})$ does not appear in $D_{\mathrm{II_S}}$, and $(pk^*, pk_o, t^*, M^*, \sigma^*)$ can pass the verification.

Definition 5. *A proxy signature scheme is* (t, q_C, q_S, ϵ) *existentially unforgeable under type* II *adaptive chosen-message attacks if no* t*-time adversary* $\mathcal{A}_{\mathrm{II}}$ *making at most* q_C *corrupt queries and* q_S *signing queries has advantage at least* ϵ *in the above game. For any PPT adversary* $\mathcal{A}_{\mathrm{II}}$ *involved in the experiment in Fig.4, we have* $\mathsf{Adv}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{II}}}(\lambda) = \Pr(\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{II}}}(\lambda, \ell) = 1) \in \mathsf{negl}(\lambda)$.

2) *Security Model for Adversary* $\mathcal{A}_{\mathrm{III}}$. $\mathcal{A}_{\mathrm{III}}$ stands for a malicious proxy signer, who wants to generate a proxy signature without knowing at least one of all the delegated tags of its proxy mediators. 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 public parameter pp, and the public keys of the original signer pk_o and proxy signers $\{pk_i\}_{i=1}^{|\mathcal{N}|}$.

Corrupt Query ($\mathcal{O}_{\mathrm{III_C}}$). The adversary issues up to q_C corrupt queries. Upon receiving a public key pk corresponding to any end-node proxy signer or some

Oracle $\mathcal{O}_{\mathrm{II_C}}(pk)$ $\mathcal{D}_{\mathrm{II_C}} \leftarrow \mathcal{D}_{\mathrm{II_C}} \cup pk$ Return sk	Oracle $\mathcal{O}_{\mathrm{II_S}}(w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$ $RL \leftarrow \emptyset$ For each $pk_i \in pk_{\mathsf{m}}$ $T \leftarrow \Gamma.\mathsf{Delegation}(pk_i, pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, w_i, T_{\mathsf{a}_i})$ For each $(pk, \mathcal{R}_t) \in \{pk_i, \mathcal{R}_t)\}_{i\in\mathsf{m}}$ $RL \leftarrow \Gamma.\mathsf{Revocation}(pk, sk, t, \mathcal{R}_t)$ $RL_t \leftarrow RL_t \cup RL$ $\sigma \leftarrow \Gamma.\mathsf{Sign}(sk, T, RL_t, M)$ $\mathcal{D}_{\mathrm{II_S}} \leftarrow \mathcal{D}_{\mathrm{II_S}} \cup (w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$ Return σ
Experiment $\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{II}}}(\lambda, \ell)$ $(pp, pk_o, sk_o) \leftarrow \Gamma.\mathsf{Setup}(1^\lambda, 1^\ell)$ $(\{pk_i, sk_i\}_{i=1}^{\lvert\mathcal{N}\rvert}) \leftarrow \Gamma.\mathsf{KeyGen}$ $\mathcal{D}_{\mathrm{II_C}}, \mathcal{D}_{\mathrm{II_S}} \leftarrow \emptyset(1^\lambda)$; $sk \leftarrow \mathcal{O}_{\mathrm{II_C}}(pk)$ $\sigma \leftarrow \mathcal{O}_{\mathrm{II_S}}(w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$ $(w^*, pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*)$ $\leftarrow \mathcal{A}_{\mathrm{II}}(pp, pk_o, \{pk_i\}_{i=1}^{\lvert\mathcal{N}\rvert}, \mathcal{O}_{\mathrm{II_C}}, \mathcal{O}_{\mathrm{II_S}})$ If $(pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}) \notin \mathcal{D}_{\mathrm{II_S}}$, and $\Gamma.\mathsf{Verify}(pk^*, pk_o, t^*, M^*, \sigma^*) = 1$ return 1 else return 0	

Fig.4. Security model for adversary $\mathcal{A}_{\mathrm{II}}$.

mediate proxy signers including the original signer, the challenger responses the related secret key sk. These queries may be asked adaptively so that each query may depend on the replies to all previous queries. A database $\mathcal{D}_{\mathrm{III_C}}$ records all the queried messages.

Delegation Query ($\mathcal{O}_{\mathrm{III_D}}$). The adversary issues up to q_D delegation queries. To each delegation query (pk, pk_{a}, w), the challenger responds by running the algorithm $T \leftarrow \Gamma.\mathsf{Delegation}(pk, pk_{\mathsf{a}}, sk_{\mathsf{a}}, w_i, T_{\mathsf{a}})$ to gain the delegated tag T and the challenger sends T to the adversary. These queries may be asked adaptively. A database $D_{\mathrm{III_D}}$ records all the delegation queries.

Revocation Query ($\mathcal{O}_{\mathrm{III_R}}$). The adversary issues up to q_R revocation queries $(pk, t, \mathcal{R}_t)$. To each query, the challenger responds by executing $RL \leftarrow \Gamma.\mathsf{Revocation}$ $(pk, sk, t, \mathcal{R}_t)$ to acquire the revocation list RL_t for revocation epoch t. Then the challenger sends RL_t to the adversary. These queries may be asked adaptively. Notice that we assume there is only one $(pk, \mathcal{R}_t)$ pair for each t.

Signing Query ($\mathcal{O}_{\mathrm{III_S}}$). The adversary makes up to q_S signing queries to the challenger. For each $(w, pk, M,$ $t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$ where $pk \notin \mathcal{R}_t$, the challenger responds by running the delegation $T \leftarrow \Gamma.\ \mathsf{Delegation}(pk_i, pk_{\mathsf{a}_i},$ $sk_{\mathsf{a}_i}, w_i, T_{\mathsf{a}_i})$ algorithm to get delegated tag T, the revocation algorithm $RL \leftarrow \Gamma.\ \mathsf{Revocation}(pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, t, \mathcal{R}_t)$ to obtain the revocation list RL_t for all proxy mediators $\{pk_i\}_{i\in\mathsf{m}}$, and $\sigma \leftarrow \Gamma.\mathsf{Sign}(sk, T, RL_t, M)$ algorithm to get the proxy signature σ. After that, the challenger sends σ to the adversary. These queries may be asked adaptively so that each query may depend on the replies to all previous queries. A database $D_{\mathrm{III_S}}$ records all the information of queries.

Output. Finally, the adversary outputs (w^*, pk^*, M^*, t^*, $\{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*$). The adversary wins if the immediate proxy ancestor pk_{a}^* has not been corrupted, $(pk^*, pk_{\mathsf{a}}^*, w^*)$ has not been queried to delegation oracle, $(pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}})$ has not been queried to signing oracle, and $(pk^*, pk_o, t^*, M^*, \sigma^*)$ can pass verification.

Definition 6. *A proxy signature scheme is* $(t, q_C, q_D, q_R, q_S, \epsilon)$ *existentially unforgeable under type* III *adaptive chosen-message attacks if no* t*-time adversary* $\mathcal{A}_{\mathrm{III}}$ *making at most* q_C *corrupt queries,* q_D *delegation queries,* q_R *revocation queries and* q_S *signing queries has advantage at least* ϵ *in the above game. For any PPT adversary* $\mathcal{A}_{\mathrm{III}}$ *involved in the experiment in Fig.*5*, we have* $\mathsf{Adv}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{III}}}(\lambda) = \mathsf{Pr}(\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{III}}}(\lambda) = 1) \in \mathsf{negl}(\lambda)$.

3) *Security Model for Adversary* $\mathcal{A}_{\mathrm{IV}}$. $\mathcal{A}_{\mathrm{IV}}$ represents revoked proxy signers who want to generate a valid proxy signature. The security model is defined

Oracle $\mathcal{O}_{\mathrm{III_C}}(pk)$ $\mathcal{D}_{\mathrm{III_C}} \leftarrow \mathcal{D}_{\mathrm{III_C}} \cup pk$ Return sk	Oracle $\mathcal{O}_{\mathrm{III_S}}(w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$ $RL \leftarrow \emptyset$ For each $pk_i \in pk_{\mathsf{m}}$ $T \leftarrow \Gamma.\mathsf{Delegation}(pk_i, pk_{\mathsf{a}_i},\ sk_{\mathsf{a}_i}, w_i, T_{\mathsf{a}_i})$ For each $(pk, \mathcal{R}_t) \in (pk_{\mathsf{a}_i}, \mathcal{R}_t)$ $RL \leftarrow \Gamma.\mathsf{Revocation}(pk_{\mathsf{a}_i},\ sk_{\mathsf{a}_i}, t, \mathcal{R}_t)$ $RL_t \leftarrow RL_t \cup RL$ $\sigma \leftarrow \Gamma.\mathsf{Sign}(sk, T, RL_t, M)$ $\mathcal{D}_{\mathrm{III_S}} \leftarrow \mathcal{D}_{\mathrm{III_S}} \cup (w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$ Return σ
Oracle $\mathcal{O}_{\mathrm{III_D}}(pk, pk_{\mathsf{a}}, w)$ $T \leftarrow \Gamma.\mathsf{Delegation}(pk, pk_{\mathsf{a}}, sk_{\mathsf{a}}, w_i, T_{\mathsf{a}})$ $\mathcal{D}_{\mathrm{III_D}} \leftarrow \mathcal{D}_{\mathrm{III_D}} \cup (pk, pk_{\mathsf{a}}, w)$ Return T	
Oracle $\mathcal{O}_{\mathrm{III_R}}(pk, t, \mathcal{R}_t)$ $RL \leftarrow \Gamma.\mathsf{Revocation}(pk, sk, t, \mathcal{R}_t)$ Return RL	

Experiment $\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{III}}}(\lambda, \ell)$
$(pp, pk_o, sk_o) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
$(\{pk_i, sk_i\}_{i=1}^{|\mathcal{N}|}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\mathcal{D}_{\mathrm{III_C}}, \mathcal{D}_{\mathrm{III_D}}, \mathcal{D}_{\mathrm{III_S}} \leftarrow \emptyset$
$sk \leftarrow \mathcal{O}_{\mathrm{III_C}}(pk)$
$T \leftarrow \mathcal{O}_{\mathrm{III_D}}(pk, pk_{\mathsf{a}}, w)$
$RL \leftarrow \mathcal{O}_{\mathrm{III_R}}(pk, t, \mathcal{R}_t)$
$\sigma \leftarrow \mathcal{O}_{\mathrm{III_S}}(pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$
$(w^*, pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*) \leftarrow \mathcal{A}_{\mathrm{III}}(pp, pk_o, \{pk_i\}_{i=1}^{|\mathcal{N}|}, \mathcal{O}_{\mathrm{III_C}}, \mathcal{O}_{\mathrm{III_D}}, \mathcal{O}_{\mathrm{III_R}}, \mathcal{O}_{\mathrm{III_S}})$
If $pk_{\mathsf{a}}^* \notin \mathcal{D}_{\mathrm{III_C}}$, $(pk^*, pk_{\mathsf{a}}^*, w^*) \notin \mathcal{D}_{\mathrm{III_D}}$, $(pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}) \notin \mathcal{D}_{\mathrm{III_S}}$, and
$\Gamma.\mathsf{Verify}(pk^*, pk_o, t^*, M^*, \sigma^*) = 1$ return 1 else return 0

Fig.5. Security model for adversary $\mathcal{A}_{\mathrm{III}}$.

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 pp and the public keys of the original signer pk_o and proxy signers $\{pk_i\}_{i=1}^{|\mathcal{N}|}$.

Corrupt Query ($\mathcal{O}_{\mathrm{IV_C}}$). The adversary issues up to q_C corrupt queries. Upon receiving a public key pk corresponding to any end-node proxy signer or some mediate proxy signer including the original signer, the challenger reveals the related secret key sk. These queries may be asked adaptively so that each query may depend on the replies to all previous queries. A database $\mathcal{D}_{\mathrm{IV_C}}$ records all the queried messages.

Delegation Query ($\mathcal{O}_{\mathrm{IV_D}}$). The adversary issues up to q_D delegation queries. To each delegation query (pk, pk_{a}, w), the challenger responds by running the algorithm $T \leftarrow \Gamma.\mathsf{Delegation}(pk, pk_{\mathsf{a}}, sk_{\mathsf{a}}, w_i, T_{\mathsf{a}})$ to gain the delegated tag T and the challenger sends T to the adversary. These queries may be asked adaptively.

Revocation Query ($\mathcal{O}_{\mathrm{IV_R}}$). The adversary issues up to q_R revocation queries $(pk, t, \mathcal{R}_t)$. To each query, the challenger responds by executing the revocation algorithm $RL \leftarrow \mathsf{Revocation}(pk, t, \mathcal{R}_t)$ to acquire the revocation list RL_t for revocation epoch t. Then the challenger sends RL_t to the adversary. These queries may be asked adaptively. Notice that we assume there is only one $(pk, \mathcal{R}_t)$ pair for each t.

Signing Query ($\mathcal{O}_{\mathrm{IV_S}}$). The adversary sends up to q_S signing queries to the challenger. For each $(w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$ where $pk \notin \mathcal{R}_t$, the challenger responds by running the delegation algorithm $T \leftarrow \Gamma.\mathsf{Delegation}(pk_i, pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, w_i, T_{\mathsf{a}_i})$ to get delegated tag T, $RL \leftarrow \Gamma.\mathsf{Revocation}(pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, t, \mathcal{R}_t)$ to gain revocation list RL_t for all proxy mediators $\{pk_i\}_{i\in\mathsf{m}}$, and the signing algorithm $\sigma \leftarrow \Gamma.\mathsf{Sign}(sk, T, RL_t, M)$ to get the proxy signature σ. After that, the challenger sends σ to the adversary. These queries may be asked adaptively so that each query may depend on the replies to all previous queries. A database $D_{\mathrm{IV_S}}$ records all the information of queries.

Output. Finally, the adversary outputs $(w^*, pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*)$. The adversary wins if $(pk^*, pk_o, t^*, M^*, \sigma^*)$ can pass the verification, pk^* is in the revoked signer set $\mathcal{R}_{t^*}$, and its immediate proxy ancestor pk_{a}^* and delegation query $(pk^*, pk_{\mathsf{a}}^*, w^*)$ have not been corrupted and queried to delegation oracle.

Definition 7. *A proxy signature scheme is* (t, q_C, q_D, q_R, q_S, ϵ)*-strongly existentially unforgeable under type* IV *adaptive chosen-message attacks if no t-time adversary $\mathcal{A}_{\mathrm{IV}}$ making at most q_C corrupt queries, q_D delegation queries, q_R revocation queries and q_S signing queries has advantage at least ϵ in the above game. For any PPT adversary $\mathcal{A}_{\mathrm{IV}}$ involved in the experiment in Fig.6, we have* $\mathsf{Adv}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{IV}}}(\lambda) = \mathsf{Pr}(\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{IV}}}(\lambda) = 1) \in \mathsf{negl}(\lambda)$.

Oracle $\mathcal{O}_{\mathrm{IV_C}}(pk)$
$\mathcal{D}_{\mathrm{IV_C}} \leftarrow \mathcal{D}_{\mathrm{IV_C}} \cup pk$
Return sk

Oracle $\mathcal{O}_{\mathrm{IV_D}}(pk, pk_{\mathsf{a}}, w)$
$T \leftarrow \Gamma.\mathsf{Delegation}(pk, pk_{\mathsf{a}}, sk_{\mathsf{a}}, w_i, T_{\mathsf{a}})$
Return T

Oracle $\mathcal{O}_{\mathrm{IV_R}}(pk, t, \mathcal{R}_t)$
$RL \leftarrow \Gamma.\mathsf{Revocation}(pk, sk, t, \mathcal{R}_t)$
$\mathcal{D}_{\mathrm{IV_R}} \leftarrow \mathcal{D}_{\mathrm{IV_R}} \cup (pk, t, \mathcal{R}_t)$
Return RL

Oracle $\mathcal{O}_{\mathrm{IV_S}}(w, pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$
$RL \leftarrow \emptyset$
For each $pk_i \in pk_{\mathsf{m}}$
$T \leftarrow \Gamma.\mathsf{Delegation}(pk_i, pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, w_i, T_{\mathsf{a}_i})$
For each $(pk, \mathcal{R}_t) \in (pk_{\mathsf{a}_i}, \mathcal{R}_t)$
$RL \leftarrow \Gamma.\mathsf{Revocation}(pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, t, \mathcal{R}_t)$
$RL_t \leftarrow RL_t \cup RL$
$\sigma \leftarrow \Gamma.\mathsf{Sign}(sk, T, RL_t, M)$
Return σ

Experiment $\mathsf{Exp}^{\mathsf{eu\text{-}cma}}_{\mathcal{A}_{\mathrm{IV}}}(\lambda, \ell)$
$(pp, pk_o, sk_o) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$
$(\{pk_i, sk_i\}_{i=1}^{|\mathcal{N}|}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\mathcal{D}_{\mathrm{IV_C}}, \mathcal{D}_{\mathrm{IV_R}} \leftarrow \emptyset$
$sk \leftarrow \mathcal{O}_{\mathrm{IV_C}}(pk)$
$T \leftarrow \mathcal{O}_{\mathrm{IV_D}}(pk, pk_{\mathsf{a}}, w)$
$\sigma \leftarrow \mathcal{O}_{\mathrm{IV_S}}(pk, M, t, \{pk_i, \mathcal{R}_t\}_{i\in\mathsf{m}})$
$(w^*, pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*) \leftarrow \mathcal{A}_{\mathrm{IV}}(pp, pk_o, \mathcal{O}_{\mathrm{IV_C}}, \mathcal{O}_{\mathrm{IV_D}}, \mathcal{O}_{\mathrm{IV_R}}, \mathcal{O}_{\mathrm{IV_S}})$
If $pk^* \in \mathcal{R}_{t^*}, pk_{\mathsf{a}}^* \notin \mathcal{D}_{\mathrm{IV_C}}$, $(pk^*, pk_{\mathsf{a}}^*, w^*) \notin \mathcal{D}_{\mathrm{IV_D}}$, and $\Gamma.\mathsf{Verify}(pk^*, pk_o, t^*, M^*, \sigma^*) = 1$
return 1 else return 0

Fig.6. Security model for adversary $\mathcal{A}_{\mathrm{IV}}$.

4 Proposed Scheme

In this section, inspired by the hierarchical revocation scheme[24], we construct a revocable and re-delegable proxy signature scheme. The idea of our scheme is that all the proxy signers can delegate their signing rights to other proxy signers by creating a new tree structure revocation list. A proxy signer needs to prove that all of its ancestral proxy signers and itself are unrevoked. Our revocable and re-delegable proxy signature has six PPT algorithms.

$\boldsymbol{\Gamma}.\mathbf{Setup}(\mathbf{1}^{\boldsymbol{\lambda}}, \mathbf{1}^{\boldsymbol{\ell}}) \rightarrow (\boldsymbol{pp}, \boldsymbol{pk_o}, \boldsymbol{sk_o})$. The original signer $\mathcal{O}$ sets up the system by running $\Delta.\mathsf{Setup}$:

$$(pp, pk_o, sk_o) \leftarrow \Delta.\mathsf{Setup}(1^{\lambda}, 1^{\ell}),$$

where the public and secret key pair (pk_o, sk_o) is generated by executing $\Sigma.\mathsf{KeyGen}$.

The public parameter pp is:

$$pp = (e, \mathbb{G}, \mathbb{G}_T, g, p, \{h_i\}_{i=0}^{\ell}, \mathcal{H}, \mathcal{H}_1).$$

$\boldsymbol{\Gamma}.\mathbf{KeyGen}(\mathbf{1}^{\boldsymbol{\lambda}}) \rightarrow (\boldsymbol{pk_i}, \boldsymbol{sk_i})$. The proxy signer $\mathcal{P}_i$ generates key pair (pk_i, sk_i) by running $\Delta.\mathsf{ProxyKeyGen}$:

$$(pk_i, sk_i) \leftarrow \Delta.\mathsf{ProxyKeyGen}(1^{\lambda}),$$

where the public and secret key pair (pk_i, sk_i) is derived from $\Sigma.\mathsf{KeyGen}$.

$\boldsymbol{\Gamma}.\mathbf{Delegation}(\boldsymbol{pk_i}, \boldsymbol{pk_{a_i}}, \boldsymbol{sk_{a_i}}, \boldsymbol{w_i}, \boldsymbol{T_{a_i}}) \rightarrow \boldsymbol{T_i}$. The mediate proxy signer $\mathcal{P}_{\mathsf{a}_i}$ (or the original signer $\mathcal{O}$) generates the delegated tag T_i for its immediate descendant proxy signer $\mathcal{P}_i$.

• A warrant w_i is an explicit description of the delegation relation.

• T_{a_i} is the delegated tag for the proxy signer $\mathcal{P}_{\mathsf{a}_i}$ or the original signer $\mathcal{O}$ ($T_o = \emptyset$).

• $\mathcal{P}_{\mathsf{a}_i}$ assigns to $\mathcal{P}_i$ an available leaf v_i of label $\langle v_i \rangle$. Let $x_0 = \epsilon, x_1, ..., x_{\ell-1}, x_\ell = v_i$ be the path from the root ϵ of T to v_i. For $j = 0$ to ℓ, $\mathcal{P}_{\mathsf{a}_i}$ does the followings:

1) consider the sub-tree T_{x_j} rooted at node x_j, and let copath_{x_j} be the co-path from x_j to v_i;

2) for each node $\omega \in \mathsf{copath}_{x_j}$, since x_j is an ancestor of ω, $\langle x_j \rangle$ is a prefix of $\langle \omega \rangle$ and we denote by $\langle \omega \rangle = \langle x_j \rangle \| \omega_{\ell_1} ... \omega_{\ell_2} \in \{0,1\}^* \times \{0,1\}^{\ell_2 - \ell_1 + 1}$, for some $\ell_1 \leqslant \ell_2 \leqslant \ell$, the suffix of $\langle \omega \rangle$ coming right after $\langle x_j \rangle$. Compute the hierarchical private key d_w by running $\Delta.\mathsf{KeyGen}$:

$$d_w \leftarrow \Delta.\mathsf{KeyGen}(pk_{\mathsf{a}_i}, sk_{\mathsf{a}_i}, pk_i, w_i, \langle x_j \rangle).$$

Parse $d_w = (D_{\omega,1}, D_{\omega,2}, K_{\omega,\ell_2-\ell_1+3}, ..., K_{\omega,\ell})$. Set $h_\omega = \mathcal{H}_1(x_j, \omega_i, pk_i)^{sk_{\mathsf{a}_i}}$, elements $D_{\omega,1}, D_{\omega,2}$ and $K_{\omega,\ell_2-\ell_1+3}, ..., K_{\omega,\ell}$ are:

$$\begin{aligned} D_{\omega,1} &= h_\omega \times (h_0 \times h_1^{\mathcal{H}(\langle x_j \rangle)} \times h_2^{\mathcal{H}(\langle \omega_{\ell_1} \rangle)} \times \cdots \times \\ &\qquad h_{\ell_2-\ell_1+2}^{\mathcal{H}(\langle \omega_{\ell_2} \rangle)})^r, \\ D_{\omega,2} &= g^r, \\ K_{\omega,\ell_2-\ell_1+3}, ..., K_{\omega,\ell} &= h_{\ell_2-\ell_1+3}^r, ..., h_\ell^r. \end{aligned}$$

• $\mathcal{P}_i$ gains the delegated tag T_i:

$$T_i \leftarrow T_{\mathsf{a}_i} \cup (pk_{\mathsf{a}_i}, pk_i, w_i, \langle v_i \rangle, \{\{d_\omega\}_{\omega \in \mathsf{copath}_{x_j}}\}_{j=0}^{\ell}).$$

$\boldsymbol{\Gamma}.\mathbf{Revocation}(\boldsymbol{pk_{a_i}}, \boldsymbol{sk_{a_i}}, \boldsymbol{t}, \boldsymbol{\mathcal{R}_t}) \rightarrow \boldsymbol{RL_t}$. The mediate proxy signer $\mathcal{P}_{\mathsf{a}_i}$ generates the revocation list for its own immediate descendant proxy signers.

• Using the subset difference covering algorithm[25], we find a cover of unrevoked user set $\mathcal{N} \setminus \mathcal{R}_t$ as the union of disjoint subsets of the form set

$$\{S_{k_1,u_1}, ..., S_{k_m,u_m}\}$$

with $m \leqslant 2 \times |\mathcal{R}| - 1$.

• For $i = 1$ to m, we do the followings.

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 $\langle x_{u_i} \rangle$ can be written as:

$$\langle x_{u_i} \rangle = \langle x_{k_i} \rangle \| u_{i,\ell_{i,1}} ... u_{i,\ell_{i,2}}.$$

2) Compute an encoding value C_i of S_{k_i,u_i} as a group element by running $\Delta.\mathsf{Encode}$:

$$C_i \leftarrow \Delta.\mathsf{Encode}(\langle x_{k_i} \rangle, \langle x_{u_i} \rangle).$$

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

3) The proxy ancestor $\mathcal{P}_{\mathsf{a}_i}$ (or the original signer $\mathcal{O}$) generates a signature Θ_i:

$$\Theta_i \leftarrow \Sigma.\mathsf{Sign}_{sk_{\mathsf{a}_i}}(C_i, g^t).$$

• Return the revocation list RL_t:

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

$\boldsymbol{\Gamma}.\mathbf{Sign}(\boldsymbol{sk_i}, \boldsymbol{T_i}, \boldsymbol{RL_t}, \boldsymbol{M}) \rightarrow \boldsymbol{\sigma}$. The proxy signer $\mathcal{P}_i$ generates a proxy signature σ for a message M. $\mathcal{P}_i$ only needs to generate the hierarchical decryption key once for the whole delegation chain in each revocation epoch.

• Suppose the delegated tag $T_i = \{T_1, T_2, ..., T_n\}$. For $k = 1$ to n, we generate the proof that shows T_k is a valid delegated tag.

1) Parse the delegated tag T_k:

$T_k = (pk'_{\mathsf{a}_i}, pk'_i, w'_i, \langle v_i \rangle, \{\{d_\omega\}_{\omega \in \mathsf{copath}_{x_j}}\}_{j=0}^{\ell})$.

2) Parse the revocation list RL:

$RL = (pk''_{\mathsf{a}_i}, t, \mathcal{R}_t, \{\langle x_{k_i} \rangle, \langle x_{u_i} \rangle, (C_i, \Theta_i)\}_{i=1}^{m})$.

3) Set $\Omega = \emptyset$, find the revocation list $RL \in RL_t$ where pk'_{a_i} in T_k is equal to pk''_{a_i} in the revocation list RL and then do the followings.

a) Determine set S_{k_l,u_l}, with $l \in \{1, ..., m\}$ 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 the secondary roots of S_{k_l,u_l}, respectively. Since x_{k_l} is an ancestor of x_{u_l}, we can write $\langle x_{u_l} \rangle$:

$$\langle x_{u_l} \rangle = \langle x_{k_l} \rangle \| u_{l,\ell_1} ... u_{l,\ell_2},$$

for some $\ell_1 < \ell_2 < \ell$ and with $u_{l,\kappa} \in \{0, 1\}$ for each $\kappa \in \{\ell_1, ..., \ell_2\}$.

b) The proxy signer $\mathcal{P}_i$ computes a hierarchical decryption key $d_{\langle x_{u_l} \rangle}$ for the next immediate mediate node of $\mathcal{P}_{\mathsf{a}_i}$:

$$d_{\langle x_{u_l} \rangle} \leftarrow \Delta.\mathsf{Derive}(pk'_{\mathsf{a}_i}, \langle x_{k_l} \rangle, d_{x_{k_l}}, \langle x_{u_l} \rangle).$$

Parse $d_{\langle x_{u_l} \rangle} = (D_{l,1}, D_{l,2})$. Set $h_\ell = \mathcal{H}_1(x_{k_l}, w_i, pk_i)^{sk_{\mathsf{a}_i}}$. Elements $D_{l,1}$ and $D_{l,2}$ are:

$$D_{l,1} = h_\ell (h_0 \times h_1^{\mathcal{H}(\langle x_{k_l} \rangle)} h_2^{\mathcal{H}(u_{l,\ell_1})} \times \cdots \times h_{\ell_2 - \ell_1 + 2}^{\mathcal{H}(u_{l,\ell_2})})^r,$$
$$D_{l,2} = g^r.$$

c) Set $\Omega_k = (pk'_{\mathsf{a}_i}, pk'_i, w'_i, x_{k_l}, x_{u_l}, D_{l,1}, D_{l,2}, C_l, \Theta_l)$ and $\Omega \leftarrow \Omega \cup \Omega_k$.

• Compute $\sigma_M \leftarrow \Sigma.\mathsf{Sign}_{sk_i}(M, \Omega)$.

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

$\boldsymbol{\Gamma.\mathsf{Verify}(pk_i, pk_o, t, M, \sigma) \rightarrow [0, 1]}$. The verifier checks the proxy signature.

• Check σ_M: if $\Sigma.\mathsf{Verify}_{pk_i}((M, \Omega), \sigma_M) \rightarrow 0$, return 0.

• Parse $\Omega = \{\Omega_1, ..., \Omega_n\}$. For $j = 1$ to n, check the message in Ω_j.

1) Parse the proof of delegation chain $\Omega_j = (pk'_{\mathsf{a}_i}, pk'_i, w'_i, \langle x_{k_l} \rangle, \langle x_{u_l} \rangle, D_{l,1}, D_{l,2}, C_l, \Theta_l)$.

2) Check Θ_l: if $\Sigma.\mathsf{Verify}_{pk'_{\mathsf{a}_i}}((C_l, g^t), \Theta_l) \rightarrow 0$, return 0.

3) Check C_l: if $\Delta.\mathsf{Verify}(pk'_{\mathsf{a}_i}, pk'_i, w'_i, \langle x_{u_l} \rangle, C_l, d_{\langle x_{u_l} \rangle}) \rightarrow 0$, return 0.

• Otherwise, return 1.

4.1 Efficiency Analysis

Since our revocable and re-delegable proxy signature is based on the subset difference (SD) revocation method[24-25], the size of a delegated tag is $O(\log^2 \mathcal{N})$ and the size of the revocation list is $O(\mathcal{R})$, where $\mathcal{N}$ is the number of system users and $\mathcal{R}$ is the number of revoked users.

The cost of signing is constant, and it signs the message M and a set of signatures Ω for all proxy signers in the delegation chain, where the set of signatures Ω only needs to be generated once in every revocation epoch. The cost of verification is linear in the number of proxy signers in the delegation chain. In reality, the delegation chain usually has a constant size. Therefore, our verification algorithm is efficient in practice.

4.2 Security Analysis for Adversary $\mathcal{A}_{\mathrm{II}}$

Theorem 1. *Our revocable and re-delegable proxy signature is (t, q_C, q_S, ϵ)-secure against the adversary $\mathcal{A}_{\mathrm{II}}$, assuming the signature scheme σ is (t', q', ϵ')-secure existentially unforgeable under an adaptive chosen-message attack, where $\epsilon' = 1/|\mathcal{N}| \times \epsilon$.*

Proof. Suppose $\mathcal{A}_{\mathrm{II}}$ is a forger that can break the scheme. There then exists a PPT algorithm $\mathcal{B}$ that can break the existential unforgeability of the signature scheme used by the proxy signer. Let $\mathcal{C}$ denote the challenger of $\mathcal{B}$.

• *Setup.* Algorithm $\mathcal{B}$ receives the challenge public key pk from $\mathcal{C}$ and sets the parameters as follows.

1) Generate the system parameters (pp, pk_o, sk_o).

2) Choose an uniformly random number k from the distribution $\{1, ..., |\mathcal{N}|\}$, and then set $pk_k = pk$.

3) Select $|\mathcal{N}| - 1$ random numbers $\{x_i\}_{i=1, i \neq k}^{|\mathcal{N}|} \in \mathbb{Z}_p$, the set $sk_i = x_i$ and $pk_i = g^{x_i}$ for $i = 1, ..., |\mathcal{N}|$ and $i \neq k$.

4) Return $(pp, pk_o, \{pk_i\}_{i=1}^{|\mathcal{N}|})$ to $\mathcal{A}_{\mathrm{II}}$.

• $\mathcal{Q}_{\mathrm{II_C}}$. Adversary $\mathcal{A}_{\mathrm{II}}$ issues up to q_C corrupt queries. Algorithm $\mathcal{B}$ responds to a query on message pk_i as follows.

1) If $i \neq k$, it finds the secret key sk_i corresponding to the public key pk_i.

2) If $i = k$, it aborts.

3) Return sk_i to $\mathcal{A}_{\mathrm{II}}$.

• $\mathcal{Q}_{\mathrm{II_S}}$. Adversary $\mathcal{A}_{\mathrm{II}}$ issues up to q_S signing queries. Algorithm $\mathcal{B}$ responds to a query on message $(w_i, pk_i, M_i, t_i, \{pk_j, \mathcal{R}_t\}_{j \in \mathsf{m}})$ as follows.

1) If $i = k$, execute $\Gamma.\mathsf{Delegation}$ to generate the private tag T_i, $\Gamma.\mathsf{Revocation}$ to gain the revocation list

RL_{t_i}, query Ω to the algorithm $\mathcal{C}$ to obtain σ_{M_i}, and then set the proxy signature $\sigma_i = (\Omega, \sigma_{M_i})$.

2) If $i \neq k$ and $k \notin \mathsf{m}$, the processes are the same as the above case except that signature σ_M is obtained from $\Gamma.\mathsf{Sign}$ algorithm.

3) If $i \neq k$ and $k \in \mathsf{m}$, the processes are the same as the above case except that the delegation tag for $\mathcal{P}_k$'s delegatee is obtained from algorithm $\mathcal{C}$.

4) Return σ_i to $\mathcal{A}_{\mathrm{II}}$.

• *Output.* Finally adversary $\mathcal{A}_{\mathrm{II}}$ outputs a forgery $(w^*, pk^*, M^*, t^*, \{pk_i^*,\ \mathcal{R}_{t^*}\}_{i\in\mathsf{mediator}}, \sigma^*)$. It aborts if $pk^* \neq pk_k$. Parse the proxy signature $\sigma^* = (\Omega^*, \sigma_{M^*})$. Algorithm $\mathcal{B}$ sends $(M^*,\ \Omega^*)$ as the message and σ_{M^*} as the forged signature to algorithm $\mathcal{C}$. Therefore algorithm $\mathcal{B}$ can break the existential unforgeability of the underlying signature scheme.

Adversary $\mathcal{A}_{\mathrm{II}}$ guessing k successfully is $1/|\mathcal{N}|$. Therefore, algorithm $\mathcal{B}$ that can break the existential unforgeability of the underlying signature scheme with the advantage ϵ' is:

$$\epsilon' = \frac{1}{|\mathcal{N}|} \times \epsilon. \qquad \square$$

4.3 Security Analysis for Adversary $\mathcal{A}_{\mathrm{III}}$

Theorem 2. *Our revocable and re-delegable proxy signature is secure against the adversary $\mathcal{A}_{\mathrm{III}}$, assuming secure existentially unforgeable under an adaptive chosen-message attack signature scheme holds and the hierarchical revocation scheme is key robust.*

Adversary $\mathcal{A}_{\mathrm{III}}$ can be divided into two cases.

• $\mathcal{A}_{\mathrm{III}1}$ forges a valid proxy signature by using a revocation list that has not been queried before.

• $\mathcal{A}_{\mathrm{III}2}$ forges a valid proxy signature by using a revocation list that has been queried before.

Lemma 1. *Our revocable and re-delegable proxy signature scheme is $(t, q_C, q_D, q_R, q_S, \epsilon)$-secure against adversary $\mathcal{A}_{\mathrm{III}1}$, assuming the signature scheme σ is (t', q', ϵ')-secure existentially unforgeable under an adaptive chosen-message attack, where $\epsilon' = 1/|\mathcal{N}| \times \epsilon$.*

Proof. Given an algorithm $\mathcal{A}_{\mathrm{III}1}$ that can break our scheme, we can construct a PPT algorithm $\mathcal{B}$ which can break the existential unforgeability of the signature scheme used by the original signer or any mediate proxy signer in the revocation algorithm. Let $\mathcal{C}$ denote the challenger of $\mathcal{B}$.

• *Setup.* Algorithm $\mathcal{B}$ receives the challenge public key pk from $\mathcal{C}$ and sets the system parameters as follows.

1) Generate the system parameters pp.

2) Choose an uniformly random number k from the distribution $\{1, ..., |\mathcal{N}|\} \cup \{o\}$, and then set $pk_k = pk$.

3) Select $|\mathcal{N}|$ random numbers $\{x_i\}_{i=1}^{|\mathcal{N}|} \in \mathbb{Z}_p$, the set $sk_i = x_i$ and $pk_i = g^{x_i}$ for $i = 1, ..., |\mathcal{N}|$ and $i \neq k$. If $k = o$, set the public key of original signer $pk_o = pk$.

4) Return $(pp, pk_o, \{pk_i\}_{i=1}^{|\mathcal{N}|})$ to $\mathcal{A}_{\mathrm{III}1}$.

• $\mathcal{Q}_{\mathrm{III_C}1}$. Adversary $\mathcal{A}_{\mathrm{III}1}$ issues up to q_C corrupt queries. Algorithm $\mathcal{B}$ responds to a query on message pk_i as follows.

1) If $i \neq k$, it finds the secret key sk_i matching the public key pk_i.

2) If $i = k$, it aborts.

3) Return sk_i to $\mathcal{A}_{\mathrm{III}1}$.

• $\mathcal{Q}_{\mathrm{III_D}1}$. Adversary $\mathcal{A}_{\mathrm{III}1}$ issues up to q_D delegation queries. Algorithm $\mathcal{B}$ responds to a query on message $(pk_i, pk_{\mathsf{a}_i}, w_i)$ as follows.

1) Assign an available label $\langle v_i \rangle$ to pk_i.

2) Select a random number r.

3) Define the co-path identity $\{\{id_x\}_{x\in\mathsf{copath}_{x_j}}\}_{j=0}^{\ell}$. For each identity, algorithm $\mathcal{B}$ does the followings.

a) Algorithm $\mathcal{B}$ simulates a signature s_x for (id_x, w_i, pk_i) by programming the random oracle $\mathcal{H}_1$. If the proxy ancestor's public key pk_{a_i} is the target public key pk_k, the signature s_x is obtained from algorithm $\mathcal{C}$ by querying (id_x, w_i, pk_i).

b) Algorithm $\mathcal{B}$ then uses s_x to compute d_x as usual.

4) Set the delegated tag and send it to adversary $\mathcal{A}_{\mathrm{III}1}$.

$$T_i = (pk_{\mathsf{a}_i}, pk_i, w_i, \langle v_i \rangle, \{\{d_x\}_{x\in\mathsf{copath}_{x_j}}\}_{j=0}^{\ell}).$$

• $\mathcal{Q}_{\mathrm{III_R}1}$. $\mathcal{B}$ uses its signing oracle to answer the revocation queries. If the querying message includes the target public key pk_k, $\mathcal{B}$ uses its signing oracle to answer the revocation after querying (C_i, g^t) to the algorithm $\mathcal{C}$.

• $\mathcal{Q}_{\mathrm{III_S}1}$. Adversary $\mathcal{A}_{\mathrm{III}1}$ issues up to q_S signing queries. Algorithm $\mathcal{B}$ responds to a query $(w_i, pk_i, M_i, t_i, \{pk_j, \mathcal{R}_t\}_{j\in\mathsf{m}})$ as follows.

1) If $(pk_i, pk_{\mathsf{a}_i}, w_i)$ has not been queried, simulate a delegation query to gain delegation tag T_i. If the proxy ancestor's public key pk_{a_i} is the target public key pk_k, it queries the algorithm $\mathcal{C}$ to obtain the signature and then simulate the delegation tag T_i.

2) If $(t_i, \{pk_j, \mathcal{R}_t\}_{j\in\mathsf{m}})$ has not been queried, simulate a revocation query to get RL_{t_i}. If the target public key is one of the members in delegation chain $(k \in \{\mathsf{m}\})$, the algorithm $\mathcal{B}$ simulates the revocation list RL_{t_i} by querying the signature to the algorithm $\mathcal{C}$.

3) Run the $\Delta.\mathsf{Sign}$ algorithm using the secret key of the proxy signer to generate the proxy signature. If the

querying public key pk_i is the target public key pk_k, the algorithm $\mathcal{B}$ will query algorithm $\mathcal{C}$ to answer the signing query.

• *Output.* Finally adversary $\mathcal{A}_{\mathrm{III}1}$ outputs a forgery $(w^*, pk^*, M^*, t^*, \{pk_i^*,\ \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*)$. If $pk^* \neq pk_k$, it aborts. Parse $\sigma^* = (\Omega^*,\ \sigma_{M^*})$ and $\Omega_i^* = (id^*, id^{*\prime}, w^*, D_1^*, D_2^*, C^*, \Theta^*)$, where $\Omega_i^* \in \Omega^*$ and the signing key of Ω_i^* has not been corrupted. Algorithm $\mathcal{B}$ sends (C^*, g^{t^*}) as the message and Θ^* as the forged signature to $\mathcal{C}$. Thus algorithm $\mathcal{B}$ can break the existential unforgeability of the signature scheme.

The probability analysis is the same as the one in type II adversary. Adversary $\mathcal{A}_{\mathrm{III}1}$ guessing k successfully is $1/|\mathcal{N}|$. Hence, algorithm $\mathcal{B}$ which can break the signature scheme with the advantage ϵ' is:

$$\epsilon' = \frac{1}{|\mathcal{N}|} \times \epsilon. \qquad \square$$

Lemma 2. *Our revocable and re-delegable proxy signature is $(t, q_C, q_D, q_R, q_S, \epsilon)$-secure against the adversary $\mathcal{A}_{\mathrm{III}}$, assuming the hierarchical revocation scheme Δ is $(t', q'_H, q'_G, \epsilon')$ key robust secure, where $\epsilon' = 1/|\mathcal{N}| \times \epsilon$.*

Proof. Suppose $\mathcal{A}_{\mathrm{III}2}$ is a forger that can break the scheme. There exists a PPT algorithm $\mathcal{B}$ which can break the key robust property of the hierarchical revocation scheme. Let $\mathcal{C}$ denote the challenger of $\mathcal{B}$.

• *Setup.* Algorithm $\mathcal{B}$ receives (pp, pk) from algorithm $\mathcal{C}$ and sets the system parameters as follows.

1) Set the system parameters pp.

2) Choose a uniformly random number k from the distribution $\{1, ..., |\mathcal{N}|\} \cup \{o\}$, and then set $pk_k = pk$.

3) Select $|\mathcal{N}|$ random numbers $\{x_i\}_{i=1}^{|\mathcal{N}|} \in \mathbb{Z}_p$, the set $sk_i = x_i$ and $pk_i = g^{x_i}$ for $i = 1, ..., |\mathcal{N}|$ and $i \neq k$. If $k = o$, set the public key of original signer $pk_o = pk$.

4) Return $(pp, pk_o, \{pk_i\}_{i=1}^{|\mathcal{N}|})$ to $\mathcal{A}_{\mathrm{III}2}$.

• $\mathcal{Q}_{\mathrm{III_C}2}$. Adversary $\mathcal{A}_{\mathrm{III}2}$ issues up to q_C corrupt queries. Algorithm $\mathcal{B}$ responds to a query on message pk_i as follows.

1) If $i \neq k$, it finds the secret key sk_i under the public key pk_i.

2) If $i = k$, it aborts.

3) Return sk_i to $\mathcal{A}_{\mathrm{III}2}$.

• $\mathcal{Q}_{\mathrm{III_D}2}$. Adversary $\mathcal{A}_{\mathrm{III}2}$ issues up to q_D delegation queries. The processes of the algorithm $\mathcal{B}$ responding to a query $(pk_i, pk_{\mathsf{a}_i}, w_i)$ is the same as $\mathcal{Q}_{\mathrm{III_D}1}$ except the algorithm $\mathcal{C}$ will return the delegation tag T_i instead of the signature s_x.

• $\mathcal{Q}_{\mathrm{III_R}2}$. Adversary $\mathcal{A}_{\mathrm{III}2}$ issues up to q_R delegation queries. The processes of algorithm $\mathcal{B}$ responding to a query $(t_i, (pk_j, \mathcal{R}_{t_i}))$ are the same as $\mathcal{Q}_{\mathrm{III_R}1}$ except that the algorithm $\mathcal{C}$ will return the revocation list RL_{t_i} under the time epoch t_i instead of the signature s_x.

• $\mathcal{Q}_{\mathrm{III_S}2}$. Adversary $\mathcal{A}_{\mathrm{III}2}$ issues up to q_S signing queries. Algorithm $\mathcal{B}$ responds to a query $(w_i, pk_i, M_i, t_i, \{pk_j, \mathcal{R}_t\}_{j\in\mathsf{mediator}})$ as $\mathcal{Q}_{\mathrm{III_S}1}$ except that algorithm $\mathcal{C}$ will return the delegation tag in delegation phase and the revocation list in revocation phase when one of proxy mediators is the target signer ($k \in \mathsf{m}$).

• *Output.* Finally adversary $\mathcal{A}_{\mathrm{III}2}$ outputs a forgery $(w^*, pk^*, M^*, t^*, \{pk_i^*,\ \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*)$. If $pk_{\mathsf{a}}^* \neq pk_k$, it aborts. Parse $\sigma^* = (\Omega^*,\ \sigma_M^*)$ and $\Omega_i^* = (id^*, id^{*\prime}, w^*, D_1^*, D_2^*, C^*, \Theta^*)$, where $\Omega_i^* \in \Omega^*$ and the signing key of Ω_i^* has not been queried. Since (w^*, pk^*) has not appeared in any delegation query, algorithm $\mathcal{B}$ returns $(id^*, id^{*\prime}, w^*,\ C^*, pk^*, D_1^*, D_2^*)$ and breaks the key robustness of the hierarchical revocation scheme.

Adversary $\mathcal{A}_{\mathrm{III}2}$ guessing k successfully is $1/|\mathcal{N}|$. Hence, algorithm $\mathcal{B}$ can break key robustness of the hierarchical revocation scheme with the advantage ϵ' is:

$$\epsilon' = \frac{1}{|\mathcal{N}|} \times \epsilon. \qquad \square$$

4.4 Security Analysis for Adversary $\mathcal{A}_{\mathrm{IV}}$

Theorem 3. *Our revocable and re-delegable proxy signature scheme is secure against adversary $\mathcal{A}_{\mathrm{IV}}$ assuming the hierarchical revocation scheme is key robust and the signature scheme used in the revocation algorithm is existentially unforgeable under adaptive chosen-message attacks.*

Adversary $\mathcal{A}_{\mathrm{IV}}$ can be divided into two cases:

• $\mathcal{A}_{\mathrm{IV}1}$ forges a valid proxy signature by changing the revocation list;

• $\mathcal{A}_{\mathrm{IV}2}$ forges a valid proxy signature by changing its position in the tree.

Lemma 3. *Our revocable and re-delegable proxy signature is $(t, q_C, q_D, q_R, q_S, \epsilon)$-secure against the adversary $\mathcal{A}_{\mathrm{IV}}$, assuming the signature scheme σ is (t', q', ϵ')-secure existentially unforgeable under an adaptive chosen-message attack, where $\epsilon' = 1/|\mathcal{N}| \times \epsilon$.*

Proof. Given an algorithm $\mathcal{A}_{\mathrm{IV}1}$ that can break our scheme, we can construct a PPT algorithm $\mathcal{B}$ which can break the existential unforgeability of the signature scheme used by the original signer or any mediate proxy signer in the revocation algorithm. Let $\mathcal{C}$ denote the challenger of $\mathcal{B}$.

• *Setup.* Algorithm $\mathcal{B}$ receives the challenge public key pk from $\mathcal{C}$ and sets the system parameters as follows.

1) Generate the system parameters pp.

2) Choose a uniformly random number k from the distribution $\{1, ..., |\mathcal{N}|\} \cup \{o\}$, and then set $pk_k = pk$.

3) Select $|\mathcal{N}|$ random numbers $\{x_i\}_{i=1}^{|\mathcal{N}|} \in \mathbb{Z}_p$, the set $sk_i = x_i$ and $pk_i = g^{x_i}$ for $i = 1, ..., |\mathcal{N}|$ and $i \neq k$. If $k = o$, set the public key of original signer $pk_o = pk$.

4) Return $(pp, pk_o, \{pk_i\}_{i=1}^{|\mathcal{N}|})$ to $\mathcal{A}_{\mathrm{IV}1}$.

• $\mathcal{Q}_{\mathrm{IV_C}1}$. Adversary $\mathcal{A}_{\mathrm{VI}1}$ issues up to q_C corrupt queries. Algorithm $\mathcal{B}$ responds to a query on message pk_i as follows.

1) If $i \neq k$, it finds the secret key sk_i corresponding to the public key pk_i.

2) If $i = k$, it aborts.

3) Return sk_i to $\mathcal{A}_{\mathrm{IV}1}$.

• $\mathcal{Q}_{\mathrm{IV_D}1}$. Adversary $\mathcal{A}_{\mathrm{IV}1}$ issues up to q_D delegation queries. Algorithm $\mathcal{B}$ responds to a query on message $(pk_i, pk_{\mathsf{a}_i}, w_i)$ as follows.

1) Assign an available label $\langle v_i \rangle$ to pk_i.

2) Select a random number r.

3) Define the co-path identity $\{\{id_x\}_{x \in \mathsf{copath}_{x_j}}\}_{j=0}^{\ell}$. For each identity, algorithm $\mathcal{B}$ does the followings.

a) Algorithm $\mathcal{B}$ simulates a signature s_x for (id_x, w_i, pk_i) by programming the random oracle $\mathcal{H}_1$. If the proxy ancestor's public key pk_{a_i} is the target public key pk_k, the signature s_x is obtained from algorithm $\mathcal{C}$ by querying (id_x, w_i, pk_i).

b) Algorithm $\mathcal{B}$ then uses s_x to compute d_x as usual.

4) Set the delegated tag and send it to adversary $\mathcal{A}_{\mathrm{IV}1}$.

$$T_i = (pk_{\mathsf{a}_i}, pk_i, w_i, \langle v_i \rangle, \{\{d_x\}_{x \in \mathsf{copath}_{x_j}}\}_{j=0}^{\ell}).$$

• $\mathcal{Q}_{\mathrm{IV_R}1}$. $\mathcal{B}$ uses its signing oracle to answer the revocation queries. If the querying message includes the target public key pk_k, $\mathcal{B}$ uses its signing oracle to answer the revocation after querying (C_i, g^t) to the algorithm $\mathcal{C}$.

• $\mathcal{Q}_{\mathrm{IV_S}1}$. Adversary $\mathcal{A}_{\mathrm{IV}1}$ issues at most q_S signing queries. Algorithm $\mathcal{B}$ responds to a query $(w_i, pk_i, M_i, t_i, \{pk_j, \mathcal{R}_t\}_{j \in \mathsf{m}})$ as follows.

1) If $(pk_i, pk_{\mathsf{a}_i}, w_i)$ has not been queried, simulate a delegation query to gain delegation tag T_i. If the proxy ancestor's public key pk_{a_i} is the target public key pk_k, it queries the algorithm $\mathcal{C}$ to obtain the signature and then simulate the delegation tag T_i.

2) If $(t_i, \{pk_j, \mathcal{R}_t\}_{j \in \mathsf{m}})$ has not been queried, simulate a revocation query to get RL_{t_i}. If the target public key is one of the members in delegation chain $(k \in \{\mathsf{m}\})$, the algorithm $\mathcal{B}$ simulates the revocation list RL_{t_i} by querying the signature to the algorithm $\mathcal{C}$.

3) Run the $\Delta.\mathsf{Sign}$ algorithm using the secret key of the proxy signer to generate the proxy signature. If the querying public key pk_i is the target public key pk_k, the algorithm $\mathcal{B}$ will query the algorithm $\mathcal{C}$ to answer the signing query.

• *Output.* Finally adversary $\mathcal{A}_{\mathrm{IV}1}$ outputs a forgery $(w^*, pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i \in \mathsf{m}}, \sigma^*)$. If $pk^* \neq pk_k$, it aborts. Parse $\sigma^* = (\Omega^*, \sigma_{M^*})$ and $\Omega_i^* = (id^*, id^{*\prime}, w^*, D_1^*, D_2^*, C^*, \Theta^*)$, where $\Omega_i^* \in \Omega^*$ and the signing key of Ω_i^* has not been corrupted. Algorithm $\mathcal{B}$ sends (C^*, g^{t^*}) as the message and Θ^* as the forged signature to $\mathcal{C}$. Thus the algorithm $\mathcal{B}$ can break the existential unforgeability of the signature scheme.

The probability analysis is the same as the one in type II adversary. Adversary $\mathcal{A}_{\mathrm{IV}1}$ guessing k successfully is $1/|\mathcal{N}|$. Hence, algorithm $\mathcal{B}$ that can break the signature scheme with the advantage ϵ' is:

$$\epsilon' = \frac{1}{|\mathcal{N}|} \times \epsilon. \qquad \square$$

Lemma 4. *Our revocable and re-delegable proxy signature is $(t, q_C, q_D, q_R, q_S, \epsilon)$-secure against the adversary $\mathcal{A}_{\mathrm{IV}}$, assuming the hierarchical revocation scheme Δ is $(t', q'_H, q'_G, \epsilon')$ key robust secure, where $\epsilon' = 1/|\mathcal{N}| \times \epsilon$.*

Proof. Suppose $\mathcal{A}_{\mathrm{IV}2}$ is a forger that can break the scheme, there exists a PPT algorithm $\mathcal{B}$ which can break the key robust property of the hierarchical revocation scheme. Let $\mathcal{C}$ denote the challenger of $\mathcal{B}$.

• *Setup.* Algorithm $\mathcal{B}$ receives (pp, pk) from algorithm $\mathcal{C}$ and sets the system parameters as follows.

1) Set the system parameter pp.

2) Choose a uniformly random number k from the distribution $\{1, ..., |\mathcal{N}|\} \cup \{o\}$, and then set $pk_k = pk$.

3) Select $|\mathcal{N}|$ random numbers $\{x_i\}_{i=1}^{|\mathcal{N}|} \in \mathbb{Z}_p$, the set $sk_i = x_i$ and $pk_i = g^{x_i}$ for $i = 1, ..., |\mathcal{N}|$ and $i \neq k$. If $k = o$, set the public key of original signer $pk_o = pk$.

4) Return $(pp, pk_o, \{pk_i\}_{i=1}^{|\mathcal{N}|})$ to $\mathcal{A}_{\mathrm{IV}2}$.

• $\mathcal{Q}_{\mathrm{IV_C}2}$. Adversary $\mathcal{A}_{\mathrm{IV}2}$ issues up to q_C corrupt queries. Algorithm $\mathcal{B}$ responds to a query on message pk_i as follows.

1) If $i \neq k$, it finds the secret key sk_i under the public key pk_i.

2) If $i = k$, it aborts.

3) Return sk_i to $\mathcal{A}_{\mathrm{IV}2}$.

• $\mathcal{Q}_{\mathrm{IV_D}2}$. Adversary $\mathcal{A}_{\mathrm{IV}2}$ issues up to q_D delegation queries. The process of the algorithm $\mathcal{B}$ responding to a query $(pk_i, pk_{\mathsf{a}_i}, w_i)$ is the same as $\mathcal{Q}_{\mathrm{IV_D}1}$ except that

algorithm $\mathcal{C}$ will return the delegation tag T_i instead of the signature s_x.

• $\mathcal{Q}_{\mathrm{IV_R}2}$. Adversary $\mathcal{A}_{\mathrm{IV}2}$ issues up to q_R delegation queries. The process of algorithm $\mathcal{B}$ responding to a query $(t_i, (pk_j, \mathcal{R}_{t_i}))$ is the same as $\mathcal{Q}_{\mathrm{IV_R}1}$ except that algorithm $\mathcal{C}$ will return the revocation list RL_{t_i} under the time epoch t_i instead of the signature s_x.

• $\mathcal{Q}_{\mathrm{IV_S}2}$. Adversary $\mathcal{A}_{\mathrm{IV}2}$ issues at most q_S signing queries. Algorithm $\mathcal{B}$ responds to a query $(w_i, pk_i, M_i, t_i, \{pk_j, \mathcal{R}_t\}_{j\in\mathsf{mediator}})$ as $\mathcal{Q}_{\mathrm{IV_S}1}$ except that algorithm $\mathcal{C}$ will return the delegation tag in delegation phase and the revocation list in revocation phase when one of proxy mediators is the target signer ($k \in \mathsf{m}$).

• *Output*. Finally adversary $\mathcal{A}_{\mathrm{IV}2}$ outputs a forgery $(w^*, pk^*, M^*, t^*, \{pk_i^*, \mathcal{R}_{t^*}\}_{i\in\mathsf{m}}, \sigma^*)$. If $pk_{\mathsf{a}}^* \neq pk_k$, it aborts. Parse $\sigma^* = (\Omega^*, \sigma_M^*)$ and $\Omega_i^* = (id^*, id^{*\prime}, w^*, D_1^*, D_2^*, C^*, \Theta^*)$, where $\Omega_i^* \in \Omega^*$ and the signing key of Ω_i^* has not been queried. Since (w^*, pk^*) has not appeared in any delegation query, algorithm $\mathcal{B}$ returns $(id^*, id^{*\prime}, w^*, C^*, pk^*, D_1^*, D_2^*,)$ and breaks the key robustness of the hierarchical revocation scheme.

Adversary $\mathcal{A}_{\mathrm{IV}2}$ guessing k successfully is $1/|\mathcal{N}|$. Hence, the algorithm $\mathcal{B}$ that can break the key robustness of the hierarchical revocation scheme with the advantage ϵ' is:

$$\epsilon' = \frac{1}{|\mathcal{N}|} \times \epsilon. \qquad \square$$

5 Revocable and Re-Delegable Vehicle Ordering System (RRVOS)

Our proxy signature scheme is a hierarchical delegation and revocation system which can be used to secure different applications. In this section, we propose a revocable and re-delegable vehicle ordering system (RRVOS) as one application of the proposed proxy signature scheme. An RRVOS system consists of the following parties: a service provider (e.g., an online transportation network company), agents (e.g., taxi companies), vehicles, and customers. The system will enhance the safety of the customers by allowing only registered and authorized vehicles to take orders. On the other hand, there should be a way to efficiently revoke vehicles that have received a lot of customer complaints. It is easy to see that our proposed proxy signature can achieve these goals and enable a scalable hierarchical management structure. The details of the system (depicted in Fig.7) are described as follows.

1) A transportation network company owns the whole system and acts as the original signer.

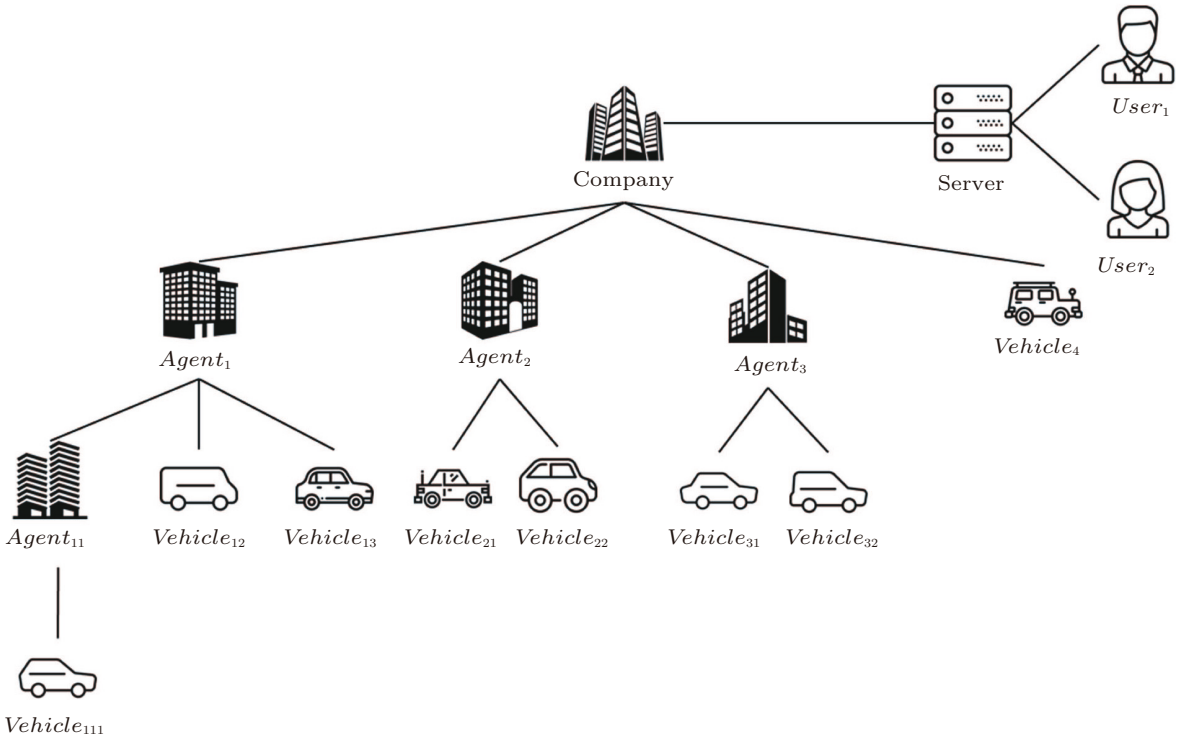

Fig.7. Vehicle ordering system.

2) The transportation network agent, which has received the delegation from the transportation network company or another upper-level agent, provides the delegation to other agents and vehicles. The agent can be further classified into two types.

a) Type-A agent has the delegation right to other agents and vehicles.
b) Type-B agent only can delegate vehicles.

3) The vehicle obtains the delegation from the transportation network company or from an agent. The responsibility of vehicles is offering the service to customers. Since the vehicles are registered and authorized by the company or its agent, the safety of the customer will be ensured.

4) The transportation network company maintains a server which offers a communication platform between customers and vehicles.

5) A customer can order the vehicles through the system: the customer sends a request with the requirements on the vehicle to the server and the server broadcasts the information to all the vehicles. In order to take the order, a vehicle generates a response that will be sent to the user.

5.1 Security Requirements of RRVOS

The following five properties should be achieved simultaneously by a secure RRVOS.

Authorization. Only authorized vehicles that have obtained the delegation from the company or an authorized agent can successfully take an order.

Revocation. The company and the agent are able to revoke the signing ability they have delegated.

Mutual Authentication. The customer and the vehicle are sure about the identity of each other.

Message Integrity. The integrity and authenticity of the messages communicated between the vehicle and the customer should be preserved.

Non-Repudiation. Anyone cannot deny the message (i.e., a request from the customer or a response from the vehicle) it has sent.

5.2 RRVOS Protocol

In this subsection, we present a secure RRVOS based on our new proxy signature scheme. In Table 1, we summarize the notations used in the protocol description.

• Setup. *Company* runs the algorithm Γ.Setup to generate the system parameters including public parameter pp, the public key pk_o, and the secret key sk_o which is used for delegation.

Table 1. Notations

Symbol	Meaning
$Company$	A transportation network company
$Agent_i$	A transportation network agent
$Vehicle_i$	A vehicle
$Server$	A server maintained by $Company$
$User_i$	A user/customer
pk_i, sk_i	A public and secret key pair of entity i
T_i	A delegation tag of entity i
$\mathcal{R}_t$	A list of revocable public key in the time epoch t
RL_t	A revocation list in the time epoch t
TS_A, TS_B	Timestamps
A	A warrant for re-delegation of agent
V	A warrant for re-delegation of vehicle

1) $Company$:
$(pp, pk_o, sk_o) \leftarrow \Gamma.\mathsf{Setup}(1^\lambda, 1^\ell)$.

• Type-A Agent Registration. Suppose $Agent_1$ has the capability of agent delegation and vehicle delegation. The registration of $Agent_1$ has three phases. $Agent_1$ first generates its public key pk_{A_1} and secret key sk_{A_2}. After that, $Agent_1$ sends the public key pk_{A_1} to $Company$. Finally, $Company$ grants the delegation tag T_{A_1} by running the algorithm Γ.Delegation with a warrant for agent and vehicle to $Agent_1$.

1) $Agent_1$:
$(pk_{A_1}, sk_{A_1}) \leftarrow \Gamma.\mathsf{KeyGen}(1^\lambda)$.
2) $Agent_1 \rightarrow Company$: pk_{A_1}.
3) $Agent_1 \leftarrow Company$:
$T_{A_1} \leftarrow \Gamma.\mathsf{Delegation}(pk_{A_1}, pk_o, sk_o, \{\mathsf{A}, \mathsf{V}\}, \emptyset)$.

• Type-B Agent Registration. Suppose $Agent_{11}$ has the capability of vehicle delegation. The registration of $Agent_{11}$ has three phases. $Agent_{11}$ first generates the public key $pk_{A_{11}}$ and the secret key $sk_{A_{11}}$ and sends the public key $pk_{A_{11}}$ to $Agent_1$. Finally, $Agent_1$ grants the delegation tag $T_{A_{11}}$ by running the algorithm Γ.Delegation with a warrant for vehicle to $Agent_{11}$.

1) $Agent_{11}$:
$(pk_{A_{11}}, sk_{A_{11}}) \leftarrow \Gamma.\mathsf{KeyGen}(1^\lambda)$.
2) $Agent_{11} \rightarrow Agent_1$: $pk_{A_{11}}$.
3) $Agent_{11} \leftarrow Agent_1$:
$T_{A_{11}} \leftarrow \Gamma.\mathsf{Delegation}(pk_{A_{11}}, pk_{A_1}, sk_{A_1}, \{\mathsf{V}\}, T_{A_1})$.

• Vehicle Registration. $Vehicle_{111}$ is a vehicle which can offer the service to the user. The registration of $Vehicle_{111}$ has three phases. $Vehicle_{111}$ first generates the public key $pk_{V_{111}}$ and the secret key $sk_{V_{111}}$ and sends the public key $pk_{A_{111}}$ to $Agent_{11}$. Finally, $Agent_{11}$ grants the delegation tag $T_{V_{111}}$ by running the algorithm Γ.Delegation to $Vehicle_{111}$.

1) $Vehicle_{111}$:
$(pk_{V_{111}}, sk_{V_{111}}) \leftarrow \Gamma.\mathsf{KeyGen}(1^\lambda)$.
2) $Vehicle_{111} \rightarrow Agent_{11}$: $pk_{V_{111}}$.
3) $Vehicle_{111} \leftarrow Agent_{11}$:
$T_{V_{111}} \leftarrow \Gamma.\mathsf{Delegation}(pk_{V_{111}}, pk_{A_{11}}, sk_{A_{11}}, \emptyset, T_{A_{11}})$.

• User Registration. The registration of $User_1$ has the following three phases. $User_1$ generates the public key pk_{U_1} and the secret key sk_{U_2} and sends the public key pk_{U_1} to *Company* who runs $\Gamma.\mathsf{Delegation}$ and sends the delegation tag T_{U_1} to $User_1$.

1) $User_1$:
$(pk_{U_1}, sk_{U_1}) \leftarrow \Gamma.\mathsf{KeyGen}(1^\lambda)$.
2) $User_1 \rightarrow Server$: pk_{U_1}.
3) $User_1 \leftarrow Server$:
$T_{U_1} \leftarrow \Gamma.\mathsf{Delegation}(pk_U, pk_o, sk_o, \emptyset, \emptyset)$.

• Type-A Agent Revocation. $Agent_1$ is a type-A agent. The revocation of $Agent_1$ has two phases. *Company* adds the public key pk_{A_1} into the revocation list $\mathcal{R}_t$ in the epoch t. Then, *Company* runs $\Gamma.\mathsf{Revocation}$ and publishes the new revocation list RL for revoking $Agent_1$.

1) *Company*: $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup pk_{A_1}$.
2) *Company*:
$RL \leftarrow \Gamma.\mathsf{Revocation}(pk_o, sk_o, t, \mathcal{R}_t)$.

• Type-B Agent Revocation. $Agent_{11}$ is a type-B agent. The revocation of $Agent_{11}$ has two phases. $Agent_1$ adds the public key $pk_{A_{11}}$ into the revocation list $\mathcal{R}_t$ in the epoch t. Then, $Agent_1$ runs $\Gamma.\mathsf{Revocation}$ and publishes the new revocation list RL for revoking $Agent_{11}$.

1) $Agent_1$: $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup pk_{A_{11}}$.
2) $Agent_1$:
$RL \leftarrow \Gamma.\mathsf{Revocation}(pk_{A_1}, sk_{A_1}, t, \mathcal{R}_t)$.

• Vehicle Revocation. $Vehicle_{12}$ is a vehicle. The revocation of $Vehicle_{12}$ has two phases. $Agent_1$ adds the public key $pk_{V_{12}}$ into the revocation list $\mathcal{R}_t$ in the epoch t. Then, $Agent_1$ runs $\Gamma.\mathsf{Revocation}$ and publishes the new revocation list RL for revoking $Vehicle_{12}$.

1) $Agent_1$: $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup pk_{V_{12}}$.
2) $Agent_1$:
$RL \leftarrow \Gamma.\mathsf{Revocation}(pk_{A_1}, sk_{A_1}, t, \mathcal{R}_t)$.

• User Revocation. $User_1$ can be revoked by *Company*. The revocation of $User_1$ has two phases. *Company* adds the public key pk_{U_1} into the revocation list $\mathcal{R}_t$ in the epoch t. Then, *Company* runs $\Gamma.\mathsf{Revocation}$ and publishes the new revocation list RL for revoking $User_1$.

1) *Server*: $\mathcal{R}_t \leftarrow \mathcal{R}_t \cup pk_{U_1}$.
2) *Server*:
$RL \leftarrow \Gamma.\mathsf{Revocation}(pk_o, sk_o, t, \mathcal{R}_t)$.

• Vehicle Ordering. $User_2$ can order vehicles by the following eight steps. $User_2$ generates the message M_{U_2} including the a time-stamp TS_A and the order including the time, location, class and type of the vehicle, etc. Then, $User_2$ sends the message M_{U_2} and the signature of the message M_{U_2} to *Server*. After that, *Server* broadcasts message M_{U_2} and the signature to all vehicles. All valid vehicles prepare message (TS, M_V) and the corresponding signature where M_V contains information such as vehicle plate number. Suppose $Vehicle_{12}$ is the first one to send the message $(TS_B, M_{V_{12}})$ and the corresponding signature to *Server* who will then forward this message to $User_2$. The detail of this protocol is as follows.

1) $User_2$:
$M_{U_2} \leftarrow (TS_A, \mathsf{Time}, \mathsf{Location}, \mathsf{Class}, \mathsf{Type})$.
2) $User_2 \rightarrow Server$:
$M_{U_2}, \sigma_{U_2} \leftarrow \Gamma.\mathsf{Sign}(sk_{U_2}, T_{U_2}, RL_t, M_{U_2})$.
3) $Server \rightarrow Vehicles$: M_{U_2}, σ_{U_2}.
4) $Vehicles$: reject the connection if

$$\Gamma.\mathsf{Verify}(pk_{U_2}, pk_o, TS_A, M_{U_2}, \sigma_{U_2}) = 0$$

or TS_A is an invalid timestamp.
5) $Vehicle_{12}$: $M_{V_{12}} \leftarrow (TS_B, \mathsf{Plate\ No.})$.
6) $Vehicle_{12} \rightarrow Server$:

$$M_{V_{12}}, \sigma_{V_{12}} \leftarrow \Gamma.\mathsf{Sign}(sk_{V_{12}}, T_{V_{12}}, RL_t, M_{V_{12}}).$$

7) $Server \rightarrow User_2$: $M_{V_{12}}, \sigma_{V_{12}}$.
8) $User_2$: reject the connection if

$$\Gamma.\mathsf{Verify}(pk_{V_{12}}, pk_o, TS_B, M_{V_{12}}, \sigma_{V_{12}}) = 0$$

or TS_B is an invalid timestamp.

5.3 Security Analysis

In this subsection, we give the security analysis based on the security requirements described in Subsection 5.1.

• The authorization requirement is satisfied by the proposed protocol since only authorized users can generate a valid proxy signature to order a vehicle and only authorized vehicles can take the order by generating a valid response message.

• The revocation of a user or vehicle is done by using the corresponding revocation mechanism in the proposed proxy signature. Based on Theorem 3, we can guarantee that a revoked user/vehicle is not able to generate a signature that can pass the verification.

• Mutual authentication is achieved by the security of the proxy signature. In particular, based on Theorem 1, we can ensure that even *Company* or any *Agent* cannot forge a valid proxy signature of a user or vehicle.

• Message integrity is achieved due to the extensional unforgeability under choose message attack (EU-CMA) of the proxy signature scheme. Therefore, no one is able to modify a message sent by the user or vehicle.

• Non-repudiation is also achieved due to the extensional unforgeability under choose message attack (EU-CMA) of the proxy signature scheme and the fact that a proxy signature is publicly verifiable.

6 Conclusions

In this paper, we proposed a novel solution for proxy signature with revocation and re-delegation. Compared with the previous approaches, our solution does not require any third party. Also, the verifier does not need to access the latest revocation list to verify a proxy signature, and our scheme provides the property of re-delegation and it has been proved secure against various types of adversaries. Moreover, we presented a secure vehicle ordering system as one of the applications of the proposed revocable and re-delegable proxy signature scheme.

Acknowledgement We thank the anonymous reviewers for their valuable comments and suggestions that have greatly helped us improve the clarity and quality of this paper.

References

[1] Yu H L, Zheng D D, Zhao B Y, Zheng W M. Understanding user behavior in large-scale video-on-demand systems. In *Proc. the 1st ACM SIGOPS/EuroSys European Conf. Computer Systems*, April 2006, pp.333-344.

[2] Goldmann M, Kreitz G. Measurements on the spotify peer-assisted music-on-demand streaming system. In *Proc. IEEE Int. Conf. Peer-to-Peer Computing*, September 2011, pp.206-211.

[3] Mambo M, Usuda K, Okamoto E. Proxy signatures: Delegation of the power to sign messages. *IEICE Trans. Fundamentals of Electronics Communications and Computer Sciences*, 1996, E79-A(9): 1338-1354.

[4] Mambo M, Usuda K, Okamoto E. Proxy signatures for delegating signing operation. In *Proc. the 3rd ACM Conf. Computer and Communications Security*, March 1996, pp.48-57.

[5] Lee B, Kim H, Kim K. Secure mobile agent using strong non-designated proxy signature. In *Proc. the 6th Australasian Conf. Information Security and Privacy*, July 2001, pp.474-486.

[6] Kim S, Park S, Won D. Proxy signatures, revisited. In *Proc. the 1st Int. Conf. Information and Communications Security*, November 1997, pp.223-232.

[7] Hwang M S, Tzeng S F, Chiou S F. An improvement of strong proxy signature and its applications. In *Proc. the Int. Conf. Security and Cryptography*, July 2008, pp.95-98.

[8] Chen X F, Zhang F G, Kim K. ID-based multi-proxy signature and blind multisignature from bilinear pairings. In *Proc. KIISC*, Nov. 2003, pp.11-19.

[9] Zhang F G, Safavi-Naini R, Lin C Y. New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairing. *IACR Cryptology ePrint Archive*, 2003. https://eprint.iacr.org/2003/104.pdf, Jan. 2018.

[10] Zhang F G, Safavi-Naini R, Susilo W. An efficient signature scheme from bilinear pairings and its applications. In *Proc. the 7th Int. Workshop on Theory and Practice in Public Key Cryptography*, March 2004, pp.277-290.

[11] Li X X, Chen K F, Li S Q. Multi-proxy signature and proxy multi-signature schemes from bilinear pairings. In *Proc. the 5th Int. Conf. Parallel and Distributed Computing: Applications and Technologies*, December 2004, pp.591-595.

[12] Yi L J, Bai G Q, Xiao G Z. Proxy multi-signature scheme: A new type of proxy signature scheme. *Electronics Letters*, 2000, 36(6): 527-528.

[13] Sun H M. Design of time-stamped proxy signatures with traceable receivers. *IEE Proceedings-Computers and Digital Techniques*, 2000, 147(6): 462-466.

[14] Schuldt J C N, Matsuura K, Paterson K G. Proxy signatures secure against proxy key exposure. In *Proc. the 11th Int. Workshop on Practice and Theory in Public-Key Cryptography*, March 2008, pp.141-161.

[15] Das M L, Saxena A, Gulati V P. An efficient proxy signature scheme with revocation. *Informatica*, 2004, 15(4): 455-464.

[16] Seo S H, Shim K A, Lee S H. A mediated proxy signature scheme with fast revocation for electronic transactions. In *Proc. the 2nd Int. Conf. Trust Privacy and Security in Digital Business*, August 2005, pp.216-225.

[17] Liu Z H, Hu Y P, Zhang X S, Ma H. Provably secure multi-proxy signature scheme with revocation in the standard model. *Computer Communications*, 2011, 34(3): 494-501.

[18] Lu E J L, Hwang M S, Huang C J. A new proxy signature scheme with revocation. *Applied Mathematics and Computation*, 2005, 161(3): 799-806.

[19] Fuchsbauer G, Pointcheval D. Anonymous proxy signatures. In *Proc. the 6th Int. Conf. Security and Cryptography for Networks*, September 2008, pp.201-217.

[20] Laberteaux K P, Haas J J, Hu Y C. Security certificate revocation list distribution for VANET. In *Proc. the 5th Int. Workshop on Vehicular Inter-NET Working*, September 2008, pp.88-89.

[21] Chaib N, Lagraa N, Yagoubi M B. EPRV: Efficient pseudonym revocation in VANETs. *Ad Hoc & Sensor Wireless Networks*, 2017, 38(1/2/3/4): 199-225.

[22] Caballero-Gil C, Molina-Gil J, Hernández-Serrano J, León O, Soriano-Ibañez M. Providing k-anonymity and revocation in ubiquitous VANETs. *Ad Hoc Networks*, 2016, 36: 482-494.

[23] Studer A, Shi E, Bai F, Perrig A. Tacking together efficient authentication, revocation, and privacy in VANETs. In *Proc. the 6th Annual IEEE Communications Society Conf. Sensor Mesh and Ad Hoc Communications and Networks*, June 2009.

[24] Xu S M, Yang G M, Mu Y, Ma S. Proxy signature with revocation. In *Proc. the 21st Australasian Conf. Information Security and Privacy*, July 2016, pp.21-36.

[25] Naor D, Naor M, Lotspiech J. Revocation and tracing schemes for stateless receivers. In *Proc. the 21st Annual Int. Cryptology Conf.*, August 2001, pp.41-62.

[26] Halevy D, Shamir A. The LSD broadcast encryption scheme. In *Proc. the 22nd Annual Int. Cryptology Conf.*, August 2002, pp.47-60.

[27] Dodis Y, Fazio N. Public key broadcast encryption for stateless receivers. In *Proc. ACM CCS-9 Workshop Digital Rights Management*, November 2002, pp.61-80.

[28] Goldwasser S, Micali S, Rivest R L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 1988, 17(2): 281-308.

[29] Boneh D, Shen E, Waters B. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Proc. the 9th Int. Conf. Theory and Practice in Public-Key Cryptography*, April 2006, pp.229-240.

[30] Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. In *Proc. the 7th Int. Conf. the Theory and Application of Cryptology and Information Security*, December 2001, pp.514-532.

[31] Boneh D, Boyen X, Goh E J. Hierarchical identity based encryption with constant size ciphertext. In *Proc. the 24th Annual Int. Conf. the Theory and Applications of Cryptographic Techniques*, May 2005, pp.440-456.

[32] Huang X Y, Mu Y, Susilo W, Zhang F G, Chen X F. A short proxy signature scheme: Efficient authentication in the ubiquitous world. In *Proc. Int. Conf. Embedded and Ubiquitous Computing*, December 2005, pp.480-489.

[33] Huang X Y, Susilo W, Mu Y, Wu W. Proxy signature without random oracles. In *Proc. the 2nd Int. Conf. Mobile Adhoc and Sensor Networks*, December 2006, pp.473-484.

Shengmin Xu received his Bachelor's and Honours degrees from the School of Computing and Information Technology, University of Wollongong, Wollongong, in 2012 and 2014, respectively. Currently, he is a Ph.D candidate at the same university, under the supervision of Dr. Guomin Yang and Prof. Yi Mu. His research interests include cryptography and information security.

Guomin Yang obtained his Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2009. He worked as a research scientist at the Temasek Laboratories of the National University of Singapore (NUS) from Sept. 2009 to May 2012. He is currently a senior lecturer at the School of Computing and Information Technology, University of Wollongong, Wollongong. His research mainly focuses on applied cryptography and network security. He received the Australian Research Council Discovery Early Career Researcher Award in 2015.

Yi Mu received his Ph.D. degree in quantum communication from the Australian National University in 1994. He currently is a professor in School of Computing and Information Technology, University of Wollongong, Wollongong. His current research interests include information security and cryptography. Prof. Mu is the editor-in-chief of International Journal of Applied Cryptography and serves as associate editor for many other international journals. He is a senior member of IEEE.