4-2016

Anonymous proxy signature with hierarchical traceability

Jiannan WEI

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Yi MU

Kaitai LIANG

Advance Access publication on 28 September 2015
doi:10.1093/comjnl/bxv080

Anonymous Proxy Signature with Hierarchical Traceability

Jiannan Wei[1], Guomin Yang[1]*, Yi Mu[1] and Kaitai Liang[2]

[1]*Centre for Information and Computer Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia*
[2]*Department of Information and Computer Science, School of Science and Technology, Aalto University, Aalto FI-00076, Finland*
**Corresponding author: gyang@uow.edu.au*

Anonymous proxy signatures are very useful in the construction of anonymous credential systems such as anonymous voting and anonymous authentication protocols. As a basic requirement, we should ensure an honest proxy signer is anonymous. However, in order to prevent the proxy signer from abusing the signing right, we should also allow dishonest signers to be traced. In this paper, we present three novel anonymous proxy signature schemes with different levels of (namely, public, internal and original signer) traceability. We define the formal definitions and security models for these three different settings, and prove the security of our proposed schemes under some standard assumptions.

Keywords: anonymity; privacy; traceability; ring signature; proxy signature

Received 16 February 2015; revised 22 April 2015
Handling editor: Kui Ren

1. INTRODUCTION

Proxy signature, introduced by Mambo *et al.* [1], is a special type of digital signature that allows a signer (or original signer) to delegate his/her signing right to other users, namely the proxy signers. The delegation is usually done through a *warrant* in which the original signer can specify the identities of the proxy signers, the message types that can be signed by the proxy signers, as well as the delegation period.

A proxy signature scheme can be used to build a credential system. For example, in order to build an e-voting system, we can let the administrator serve as the original signer and delegate its signing right to the legitimate voters. By checking the validity of a proxy signature on the vote submitted by a user, we can determine whether the user is a legitimated voter or not.

In the above e-voting application, it is also desirable to ensure the anonymity of the voters can be preserved (i.e. the administrator cannot discover the real identity of a voter or proxy signer). However, this will bring another issue—a proxy signer may abuse the signing right and cast multiple votes. So a mechanism for tracing dishonest proxy signers is also necessary in the construction of an anonymous proxy signature scheme.

Informally, we can summarize the desirable security properties of an anonymous proxy signature scheme as follows:

(i) *Unforgeability.* Only legitimate proxy signers can generate valid proxy signatures.
(ii) *Anonymity.* As long as the proxy signer behaves honestly (e.g. cast only one vote for each topic in the e-voting scenario), his/her identity is well protected, which means even the original signer cannot determine the identity of the proxy signer or link two different signatures generated by the proxy signer.
(iii) *Traceability.* If a proxy signer behaves dishonestly, then someone can recover the identity of the proxy signer. Here we can define a **hierarchy** in terms of traceability:
    Level 1: *Public Traceability.* Anyone can trace a dishonest proxy signer.
    Level 2: *Internal Traceability.* Only internal users, including the original signer and other legitimate proxy signers, can trace a dishonest proxy signer.
    Level 3: *Original Signer Traceability.* Only the original signer can trace a dishonest proxy signer.

The different levels of traceability can be applied in different applications. For example, we can use an anonymous proxy

signature with Level 3 (i.e. original signer) traceability to build an anonymous paper review system for academic conferences. The program chair can serve as the original signer, and the program committee (PC) members can serve as proxy signers. If a PC member only gives one report for a paper, then no one including the program chair can recover the identity of the PC member. However, if a PC member submits two different reports for a paper, e.g. the paper is written by some friends and the PC member wants to increase the average score of the paper, then his/her identity can be traced by the program chair. It is worth noting that if a PC member submits a modified report for a paper at a later time, then the identity can be recovered by the program chair. But this is unavoidable since otherwise we cannot prevent the PC member from submitting two reports for one paper without being caught. Nevertheless, with Level 3 traceability, we can still ensure other PC members cannot trace the identity of a PC member under any circumstances.

1.1. Our contributions

Based on the motivation outlined above, we perform a formal study on anonymous proxy signature with hierarchical traceability in this paper.

We first define the formal definitions and security models for anonymous proxy signature with public, internal and original signer traceability, respectively. Our security models capture all the security requirements we have informally presented above. In terms of the Level 2 (i.e. internal) traceability, we also define the security model for untraceability against outsiders. Similarly, we also define untraceability against proxy signers for Level 3 (i.e. original signer) traceability.

We also present three concrete anonymous proxy signature schemes that can achieve different levels of traceability. Our first scheme with public traceability is based on the traceable ring signature scheme proposed by Fujisaki and Suzuki [2] by extending their scheme to the proxy setting. The main difficulty of this work is to construct anonymous proxy signature schemes with Level 2/3 traceability. To achieve Level 2 traceability, we modify our first scheme so that each proxy signature is randomized and only the original signer or a proxy signer, who has a secret de-randomizing key that is generated by the original signer and distributed to all the proxy signers, can perform the de-randomization operation. Also, we add two (efficient) non-interactive zero-knowledge (NIZK) proofs to enforce the de-randomizing key is used in the proxy signing operation. To deal with Level 3 traceability, we further extend our scheme to allow each proxy signer receive a distinct de-randomizing key from the original signer. We prove that our proposed schemes are secure under the Discrete Log (DL) and the Decisional Diffie–Hellman (DDH) assumptions.

1.2. Related work

The first proxy signature scheme was proposed by Mambo *et al.* [1]. In their seminal work, they classified proxy signatures into three main categories, namely full delegation, partial delegation and delegation by warrant. Shortly after that, Kim *et al.* [3] proposed a new type of proxy signature combining partial delegation and warrant. They further showed that such a combination can provide a higher level of security. Since then, many proxy signature schemes based on partial delegation and warrant have been proposed (e.g. [4–7]).

Many extensions on proxy signature have also been proposed according to different application needs, such as threshold proxy signature [8, 9], proxy blind signature [10], one-time proxy signature [11]. Threshold proxy signature, also known as multi-proxy signature, enables an original signer to delegate his signing right to multiple proxy signers. The proxy signers need to work together in order to produce a valid proxy signature on behalf of the original signer. One-time proxy signature puts strict restrictions on the signing capability of a proxy signer, who is only allowed to generate one valid proxy signature on behalf of the original signer. Proxy blind signature allows a user to obtain a valid signature on a message from a proxy signer in a way that the proxy signer learns neither the message nor the resulting signature.

Signer anonymity is an important security requirement of many digital signature schemes such as Group signatures [12], Ring signatures [13] and Anonymous signatures [14]. An anonymous signature [14] is an ordinary digital signature with the additional property that the signer's identity is protected if the message is unknown to the adversary. In a ring signature scheme [13], any ring member can sign messages on behalf of the whole ring without revealing his/her real identity. Moreover, no one can determine whether two signatures have been signed by the same ring member or not. A group signature [12] is similar except that it has a group manager who is able to reveal the identity of the signer for any valid group signature.

To construct an anonymous proxy signature, we can combine a normal proxy signature with one of the above signatures with signer anonymity. However, anonymous signature [14] is not suitable for our purpose since it requires the message to be hidden. A group signature [12] is not suitable either since the group manager can trace any valid group signature. Hence, several existing anonymous proxy signature schemes [5, 15–17] used the idea of combining ring signature with proxy signature to construct anonymous proxy signatures. However, there is a potential problem in these schemes: since there is no way to trace the identity of a proxy signer, he/she may abuse the signing right without being caught.

The notion of Traceable Signature, introduced by Kiayias *et al.* [18], is an extension of group signature, which allows a third party that is authorized by the group manager to trace group signatures generated by dishonest group signers. However, the identities of honest group signers are well protected against the third party. Nevertheless, the group manager can still trace all the signatures, and hence is not suitable for our purpose. To address the traceability problem such that no one can trace signatures generated by an honest signer, in this paper, we

make use a special type of ring signature named *traceable* ring signature, which was introduced by Fujisaki and Suzuki [2]. A traceable ring signature is a tag-based signature with the restriction that a ring member can only sign once per tag in order to remain anonymous. If a ring member signs two different messages under the same tag, then his/her identity can be *publicly* traced. As we will show later, by combining proxy signature with traceable ring signature, we can build an anonymous proxy signature scheme with public traceability. However, the construction of a scheme with type 2/3 traceability is not straightforward, and requires some novel design ideas.

In [19], Wei *et al.* partially solved the problem and proposed an anonymous proxy signature with restricted traceability, which is similar to the internal traceability defined in this paper. However, the scheme presented in [19] only achieves very weak security. To be more precise, it is only proved that a proxy signer cannot forge a delegation key without the help of the original signer. On the other hand, the standard definition of unforgeability requires that a proxy signer cannot forge a valid proxy signature. Note that from the adversary's view point, forging a proxy signature is strictly easier than forging a delegation key. Also, there is no anonymous proxy signature with Level 1/3 traceability presented in [19]. In this paper, we address all these problems by presenting anonymous proxy signature schemes, which can achieve the standard notion of unforgeability and provide different levels of traceability.

Paper organization. In the next section, we give the definition and security model for anonymous proxy signature with public traceability, and present a concrete scheme with security proofs. Then in Section 3, we present a new scheme that achieves internal traceability. We also proved the untraceability against outsiders for the second scheme. Lastly, we present our construction of anonymous proxy signature with original signer traceability in Section 4. We also give an efficiency analysis for the proposed schemes in Section 5 and conclude the paper in Section 6.

2. ANONYMOUS PROXY SIGNATURE WITH PUBLIC TRACEABILITY

2.1. Definition

An anonymous proxy signature with public traceability consists of the following algorithms:

Parameter generation ($\mathcal{PG}$): This is a probabilistic polynomial-time (PPT) algorithm that on input a security parameter κ outputs the system parameters *Param*.

Key generation ($\mathcal{KG}$): Given the system parameters *Param*, it outputs a user public and private key pair (Y, x).

Delegation sign ($\mathcal{DS}$): Given a warrant m_ω[1] and an original signer's private key x_0, it outputs a delegation signing key σ_0 for m_ω.

Delegation verification ($\mathcal{DV}$): Given an original signer's public key Y_0, a warrant m_ω, and a delegation signing key σ_0, it outputs 'accept' if σ_0 is valid with regard to Y_0 and m_ω; otherwise, it outputs 'reject'. A proxy signing key psk_i is generated based on $(Y_0, m_\omega, \sigma_0)$, and a proxy signer's private key x_i, if the delegation signing key σ_0 is valid.

Proxy Sign ($\mathcal{PS}$): On input a message $m \in \{0,1\}^*$, a tag $L = (issue, Y_N)$, where $Y_N = \{Y_1, \ldots, Y_n\}$ are the public keys of the proxy signers in the warrant m_ω, and a proxy signing key psk, it outputs a proxy signature σ.

Proxy Verification ($\mathcal{PV}$): On input a message m, the original signer's public key Y_0, a proxy signature σ, a tag $L = (issue, Y_N)$ as defined above and a warrant m_ω, it outputs 'accept' if the signature is valid; otherwise, it outputs 'reject'.

Trace ($\mathcal{TR}$): On input two message-signature pairs (m, σ) and (m', σ') with respect to the same tag L and warrant m_ω, it outputs either 'indep', 'linked', or $Y_i \in Y_N$.

Remark. Similar as a traceable ring signature [2], the trace algorithm has three possible outputs:

(1) if σ and σ' are generated by different signers, then **Trace** returns 'indep';
(2) else, if σ and σ' are generated by one signer, then
   (a) if $m = m'$ (i.e. a signer signs the same message m twice under the same tag and warrant), then **Trace** returns 'linked';
   (b) else, if $m \neq m'$ (i.e. a signer signs two different messages under the same tag and warrant), then **Trace** returns the public key Y_i of the signer.

2.2. Security model

In this section, we present the security models for anonymous proxy signature with public traceability.

2.2.1. *Unforgeability*

There are three types of adversaries:

Type I Adversary (or an outsider). This type of adversary only has the public keys of the original signer and the proxy signers. His aim is to forge a valid proxy signature.

Type II Adversary is a proxy signer. This type of adversary has all the public keys and the private key of a proxy signer. His aim is to forge a proxy signature for a warrant m_ω that has not been delegated by the original signer.

Type III Adversary is the original signer. This type of adversary has all the public keys and the private key of the original signer. His aim is to forge a valid proxy signature on behalf of a proxy signer.

It is obvious that if an anonymous proxy signature scheme is unforgeable against Type II and Type III adversaries, it is also unforgeable against Type I adversary. So we will only focus on the adversarial models with regard to Type II and Type III adversaries in the rest of this paper.

[1] In this paper, we assume that the proxy signers' public keys are explicitly included in a warrant.

Unforgeability against type II adversary

$\mathcal{A}_{II}$ aims to forge a valid proxy signature for a warrant, which has not been delegated by the original signer. The model is defined via the following game.

Setup: The challenger $\mathcal{C}$ runs the key generation algorithm to generate the secret key and public key pairs (x_0, Y_0), $(x_1, Y_1), \ldots, (x_n, Y_n)$ representing the keys of the original signer and n proxy signers, respectively. $\mathcal{C}$ then sends $(Y_0, Y_1, \ldots, Y_n, x_1, \ldots, x_n)$ to the adversary $\mathcal{A}_{II}$.

Delegation signing queries: $\mathcal{A}_{II}$ can request a delegation signing key on any warrant m_ω he chooses. In response, $\mathcal{C}$ returns a signature σ_0 for m_ω.

Output: Finally, $\mathcal{A}_{II}$ outputs a warrant m^*_ω, a tag $L = (issue, Y_N)$ and a message-signature pair (m, σ^*) and we say $\mathcal{A}_{II}$ wins the game if

(i) σ^* is a valid proxy signature with respect to m and m^*_ω; and
(ii) m^*_ω has never appeared in the delegation signing queries.

We define the advantage of the adversary as

$$\mathbf{Adv}^{UF}_{\mathcal{A}_{II}}(k) = \Pr[\mathcal{A}_{II} \text{ wins the game}].$$

Unforgeability against Type III Adversary

$\mathcal{A}_{III}$ is an original signer who aims to forge an anonymous proxy signature. It is defined via the following security game.

Setup: The challenger $\mathcal{C}$ runs the key generation algorithm to obtain the secret key and public key pairs (x_0, Y_0), $(x_1, Y_1), \ldots, (x_n, Y_n)$ representing the keys of the original signer and n proxy signers, respectively. $\mathcal{C}$ then sends $(Y_0, Y_1, \ldots, Y_n, x_0)$ to the adversary $\mathcal{A}_{III}$.

Proxy signing queries: $\mathcal{A}_{III}$ can access the proxy signing oracles: $\mathbf{Sig}_{psk_i}$ for any $1 \le i \le n$ by providing a valid delegation signing key σ_0 for any warrant m_ω, and a tag L, which includes Y_i. $\mathcal{C}$ then generates the proxy signature using psk_i and returns it to $\mathcal{A}_{III}$.

Output: Finally, $\mathcal{A}_{III}$ outputs a warrant m_ω, a tag $L = (issue, Y_N)$, and a message-signature pair (m^*, σ^*) and we say $\mathcal{A}_{III}$ wins the game if

(i) σ^* is a valid proxy signature; and
(ii) (m_ω, L, m^*) has never appeared in the proxy signing queries.

We define the advantage of the adversary as

$$\mathbf{Adv}^{UF}_{\mathcal{A}_{III}}(k) = \Pr[\mathcal{A}_{III} \text{ wins the game}].$$

2.2.2. *Anonymity against original signer*

If a proxy signer is honest, then a proxy signature should remain anonymous even against the original signer. We define the anonymity against the original signer via the following game.

Setup: The challenger $\mathcal{C}$ runs the key generation algorithm to obtain the secret key and public key pairs (x_0, Y_0), $(x_1, Y_1), \ldots, (x_n, Y_n)$ representing the keys of the original signer and n proxy signers, respectively. $\mathcal{C}$ then sends $(Y_0, Y_1, \ldots, Y_n, x_0)$ to the adversary $\mathcal{A}$.

Key selection: The adversary $\mathcal{A}$ outputs (Y_i, Y_j) as the two target proxy signer's public keys. Let $b \in \{i, j\}$ be a random bit chosen by the challenger.

Proxy signing query: $\mathcal{A}$ may access three signing oracles: $\mathbf{Sig}_{psk_b}$, $\mathbf{Sig}_{psk_i}$ and $\mathbf{Sig}_{psk_j}$ by providing a valid delegation signing key σ_0 for any warrant m_ω, and a tag L which includes both Y_i and Y_j.

(i) $\mathbf{Sig}_{psk_b}$ is the signing oracle with respect to proxy signer b (note that $b \in \{i, j\}$) who has a valid proxy signing key psk_b derived based on x_b and σ_0;
(ii) $\mathbf{Sig}_{psk_i}$ (respectively, $\mathbf{Sig}_{psk_j}$) is the signing oracle with respect to proxy signer i (respectively, proxy signer j) who has a valid proxy signing key psk_i (respectively, psk_j) derived based on x_i (respectively, x_j) and σ_0.

The following conditions must hold for all the signing queries made by $\mathcal{A}$:

(i) If (L, m) and (L, m') are two queries of $\mathcal{A}$ to the challenge signing oracle $\mathbf{Sig}_{psk_b}$, then $m = m'$.
(ii) If (L, m) is a query of $\mathcal{A}$ to $\mathbf{Sig}_{psk_b}$ and $(\hat{L}, \hat{m})$ is a query of $\mathcal{A}$ to $\mathbf{Sig}_{psk_i}$ or $\mathbf{Sig}_{psk_j}$, then $L \neq \hat{L}$.

Output: Finally, $\mathcal{A}$ outputs a bit b'. $\mathcal{A}$ wins the game if $b' = b$. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{AN}_{\mathcal{A}}(k) = \Pr[b = b'] - \tfrac{1}{2}.$$

2.2.3. *Tag-linkability*

Tag-linkability is defined by following the definition given in Fujisaki and Suzuki's paper [2]. The adversary $\mathcal{A}$, which is a PPT algorithm, takes as input the system parameters, and outputs the original signer's public key Y_0, a warrant m_ω and a delegation signing key σ_0, a tag $L = (issue, Y_N)$, where $Y_N = (Y_1, \ldots, Y_n)$ are the proxy signers' public keys, and $n + 1$ message-signature pairs $\{(m^{(1)}, \sigma^{(1)}), \ldots, (m^{(n+1)}, \sigma^{(n+1)})\}$. The adversary's advantage is defined as

$$\mathbf{Adv}^{TL}_{\mathcal{A}}(k) = \Pr[\mathbf{Expt}^{(\mathcal{A})}(k) = 1].$$

The experiment $\mathbf{Expt}^{\mathcal{A}}(k)$ is defined as follows:

(1) $(Y_0, m_\omega, \sigma_0, L, \{(m^{(1)}, \sigma^{(1)}), \ldots, (m^{(n+1)}, \sigma^{(n+1)})\}) \leftarrow \mathcal{A}(1^k)$;
(2) Return 1 iff
  (a) $\mathcal{DV}(Y_0, m_\omega, \sigma_0) = 1$, and
  (b) $\mathcal{PV}(Y_0, m_\omega, L, m^{(i)}, \sigma^{(i)}) = 1$, for all $i \in \{1, \ldots, n+1\}$, and
  (c) $\mathcal{TR}(Y_0, m_\omega, \sigma_0, L, m^{(i)}, \sigma^{(i)}, m^{(j)}, \sigma^{(j)}) =$ 'indep' for all $i, j \in \{1, \ldots, n+1\}$ where $i \neq j$,

where $\mathcal{DV}$, $\mathcal{PV}$ and $\mathcal{TR}$ are the Delegation Verification, Proxy Verification and Trace algorithms defined in Section 2.1.

2.3. Scheme 1 with public traceability

The idea behind our anonymous proxy signature with public traceability is to extend Fujisaki and Suzuki's publicly traceable ring signature scheme [2] to the proxy environment. To deal with delegation signing, we apply the Schnorr signature scheme [20] to generate the delegation signing keys. The details of the scheme are presented below.

Parameter generation. Taking as input the security parameter κ, this algorithm outputs (G, q, P), where G is a cyclic group of prime order q and P is a generator of G. Let $H_0 : \{0,1\} * G \rightarrow \mathbb{Z}_q^*$, $H : \{0,1\}^* \rightarrow G$, $H' : \{0,1\}^* \rightarrow G$ and $H'' : \{0,1\}^* \rightarrow \mathbb{Z}_q$ denote independent cryptographic hash functions. The system parameters are $Param = (G, q, P, H_0, H, H', H'')$.

Key generation. User i randomly selects $x_i \in \mathbb{Z}_q$ and computes $Y_i = x_i P$. The public key of user i is Y_i and the corresponding secret key is x_i.

Delegation sign. The original signer first generates a warrant m_ω. There is an explicit description of the delegation relation such as the identities of the original signer u_0 and the proxy signers $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ and their public keys, the expiration time of the delegation, etc. The original signer picks a random number $r \in \mathbb{Z}_q$ and computes $R = rP$, $s = r + x_0 H_0(m_\omega, R) \bmod q$. Finally, the original signer sends (m_ω, R, s) to all the proxy signers in $\mathcal{U}$ via a secure channel.

Delegation verification. Upon receiving (m_ω, R, s), the proxy signer u_i checks if $sP = R + H_0(m_\omega, R)Y_0$. If it does not hold, the delegation is rejected. Otherwise, the proxy signer u_i computes his/her proxy signing secret key $psk_i = s + x_i H_0(m_\omega, R) = r + H_0(m_\omega, R)(x_0 + x_i) \bmod q$. For simplicity, we use $pk_i = psk_i P = R + H_0(m_\omega, R)(Y_0 + Y_i)$ to denote the corresponding proxy signing public key in the rest of the scheme.

Proxy sign. To sign a message $m \in \{0,1\}^*$ with respect to a tag $L = (issue, Y_N)$, where Y_N are public keys of the proxy signers described in the warrant m_ω, the real proxy signer u_i proceeds as follows:

(1) Compute $F = H(L)$ and $\sigma_i = psk_i F$.
(2) Set $A_0 = H'(L, m)$ and $A_1 = 1/i(\sigma_i - A_0)$.
(3) For all $j \neq i$, compute $\sigma_j = A_0 + jA_1 \in G$. Note that every $(j, \log_F(\sigma_j))$ is on the line defined by $(0, \log_F(A_0))$ and (i, psk_i).
(4) Generate (c_N, z_N) based on a (non-interactive) zero-knowledge proof of knowledge for the language

$$\mathcal{L} = \{(L, F, \sigma_N) \mid \exists i \in N \; s.t. \log_P(pk_i) = \log_F(\sigma_i)\},$$

where $\sigma_N = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ as follows:

(a) Pick random $\omega_i \leftarrow \mathbb{Z}_q$ and set $a_i = \omega_i P, b_i = \omega_i F \in G$.
(b) Pick random $z_j, c_j \leftarrow \mathbb{Z}_q$, and set $a_j = z_j P + c_j pk_j$, $b_j = z_j F + c_j \sigma_j \in G$ for every $j \neq i$.
(c) Set $c = H''(L, m, A_0, A_1, a_N, b_N)$, where $a_N = (a_1, \ldots, a_n)$ and $b_N = (b_1, \ldots, b_n)$.
(d) Set $c_i = c - \Sigma_{j \neq i} c_j \bmod q$ and $z_i = \omega_i - c_i \cdot psk_i \bmod q$.
(e) Return (c_N, z_N), where $c_N = (c_1, \ldots, c_n)$ and $z_N = (z_1, \ldots, z_n)$, as a proof for $\mathcal{L}$.

(5) Return $\sigma = (A_1, R, c_N, z_N)$ as the signature on (L, m).

Verification. To verify a proxy signature $\sigma = (A_1, R, c_N, z_N)$ on message m and tag L, check the following:

(1) Parse L as $(issue, Y_N)$, and compute $pk_i = R + H_0(m_\omega, R)(Y_0 + Y_i)$ for all $i \in N$.
(2) Set $F = H(L)$ and $A_0 = H'(L, m)$, and compute $\sigma_i = A_0 + iA_1 \in G$ for all $i \in N$.
(3) Compute $a_i = z_i P + c_i pk_i$, $b_i = z_i F + c_i \sigma_i$, for all $i \in N$.
(4) Check that $H''(L, m, A_0, A_1, a_N, b_N) = \Sigma_{i \in N} c_i \bmod q$, where $a_N = (a_1, \ldots, a_n)$ and $b_N = (b_1, \ldots, b_n)$.
(5) If all the above checks are successful, outputs accept; otherwise, outputs reject.

Trace. To check the relation between (m, σ) and (m', σ') under the same warrant m_ω and the same tag L where $\sigma = (A_1, R, c_N, z_N)$ and $\sigma' = (A_1', R', c_N', z_N')$, check the following:

(1) Parse L as $(issue, Y_N)$. Set $F = H(L)$ and $A_0 = H'(L, m)$ and compute $\sigma_i = A_0 + iA_1 \in G$ for all $i \in N$. Do the same operation for σ' to get σ_i' for all $i \in N$.
(2) For all $i \in N$, if $\sigma_i = \sigma_i'$, store Y_i in **TList**, where **TList** is initially empty.
(3) Output Y if Y is the only entry in **TList**; 'linked' if **TList** $= Y_N$; 'indep' otherwise.

Correctness. The correctness of our scheme can be verified as follows:

$$\begin{aligned}
&z_i P + c_i pk_i \\
&\quad = g[\omega_i - c_i[r + H_0(m_\omega, R)(x_0 + x_i)]]P + c_i pk_i \\
&\quad = \omega_i P - c_i[r + H_0(m_\omega, R)(x_0 + x_i)]P \\
&\qquad + c_i[R + H_0(m_\omega, R)(Y_0 + Y_i)] \\
&\quad = \omega_i P \\
&\quad = a_i \\
&z_i F + c_i \sigma_i \\
&\quad = \Big[\omega_i - c_i\big[r + H_0(m_\omega, R)(x_0 + x_i)\big]\Big]F \\
&\qquad + c_i psk_i F \\
&\quad = \omega_i F - c_i\big(r + H_0(m_\omega, R)(x_0 + x_i)\big)F \\
&\qquad + c_i F(r + H_0(m_\omega, R)(x_0 + x_i)) \\
&\quad = \omega_i F \\
&\quad = b_i.
\end{aligned}$$

2.4. Security analysis

In this section, we analyse the security of our proposed anonymous proxy signature with public traceability. The security

proofs for Tag-linkability, Anonymity and Unforgeability against the Type III adversary can be obtained directly by following the proofs in [2] since the proxy signers in our scheme have the same role as the ring signers in Fujisaki and Suzuki's paper [2]. Below we focus on the security proof for *Unforgeability against the Type II adversary*.

DEFINITION 2.1 ((DL Problem)). *Let G denote a cyclic group of order q and P a generator of G. Given* $(P, xP) \in G^2$ *for a randomly selected* $x \in \mathbb{Z}_q$, *compute x. The advantage of an algorithm* $\mathcal{A}$ *to solve the DLP is defined as*

$$Adv_{\mathcal{A}}^{DLP}(k) = \Pr[\mathcal{A}(P, xP) = x].$$

THEOREM 2.1. *If there exists a type II adversary* $\mathcal{A}_{\mathcal{II}}$, *which can break the Type II unforgeability of the proposed anonymous proxy signature scheme*, *then we can construct another adversary* $\mathcal{B}$ *who can use* $\mathcal{A}_{\mathcal{II}}$ *to solve the DL problem.*

Proof. We prove the theorem via the following two lemmas.

LEMMA 2.1. *If the adversary* $\mathcal{A}_{II}$ *can forge a valid signature in our scheme*, *then we can construct another adversary* $\mathcal{A}'_{II}$ *who can forge a valid delegation signing key.*

After $\mathcal{A}'_{II}$ obtains $(Y_0, Y_1, \ldots, Y_n, x_1, \ldots, x_n)$, it simulates the game for $\mathcal{A}_{II}$ as follows.

$\mathcal{A}'_{II}$ answers H_0, H, H' hash queries and the delegation signing queries by referring those queries to its own oracles. $\mathcal{A}'_{II}$ simulates H'' queries by maintaining a hash table for the hash function H'' and answers $\mathcal{A}_{II}$'s H'' queries as follows: whenever a message is queried, $\mathcal{A}'_{II}$ first checks if a record for this input exists in the table. If so, the same output is returned; otherwise, $\mathcal{A}'_{II}$ chooses a random number in $\mathbb{Z}_q$ as the hash output, and adds the hash input and output into the table, and finally returns the output to $\mathcal{A}_{II}$.

Finally, $\mathcal{A}_{II}$ outputs a proxy signature $\sigma^* = (A_1, R, c_N, z_N)$, which is a valid signature for the message m and the warrant m^*_ω. Note that m^*_ω should not be queried in any delegation signing queries.

Due to the use of the Proof of Knowledge protocol for the language $\mathcal{L}$ in the scheme, if the adversary $\mathcal{A}_{II}$ can output a valid signature (A_1, R, c_N, z_N), then by rewinding $\mathcal{A}_{II}$ and providing another hash value c'_N, $\mathcal{A}'_{II}$ can obtain another valid signature (A_1, R, c'_N, z'_N) where $c_N \neq c'_N$, which means there exists at least one index i such that $c_i \neq c'_i$. Then $\mathcal{A}'_{II}$ can compute

$$psk_i = \frac{z'_i - z_i}{c'_i - c_i} \bmod q$$

which is a valid proxy signing key of proxy signer i for warrant m^*_ω. Then $\mathcal{A}'_{II}$ can return $s = psk_i - x_i H_0(m^*_\omega, R)$, which is a valid delegation signing key. Notice that $\mathcal{B}$ knows the secret keys of all the proxy signers.

LEMMA 2.2. *If there exists an adversary* $\mathcal{A}'_{II}$ *who can forge a valid delegation signing key*, *then we can construct another algorithm* $\mathcal{B}$ *who can solve the DL problem.*

Given $(P, Y^* = x^*P)$ for some unknown $x^* \in \mathbb{Z}_q$ as an instance of DL problem, we will show how $\mathcal{B}$ can use $\mathcal{A}'_{\mathcal{II}}$ to find x^*. $\mathcal{B}$ sets the original signer's public key $Y_0 = Y^* = x^*P$, and generates the keys for the proxy signers honestly. After that, $\mathcal{B}$ sends Y_0 and all the public/private key pairs of the proxy signers to adversary $\mathcal{A}'_{II}$.

$\mathcal{B}$ maintains a table for the hash function H_0 to record all the hash queries/answers as follows:

H_0 **hash queries:** $\mathcal{A}'_{II}$ sends a query (m_ω, R), $\mathcal{B}$ will check the hash table.

(1) If (m_ω, R) has already been queried to H_0 oracle, which means there is a record of $((m_\omega, R), h_i)$ in the hash table, $\mathcal{B}$ simply returns h_i to $\mathcal{A}'_{\mathcal{II}}$.
(2) Otherwise, $\mathcal{B}$ chooses a random number $h_i \in \mathbb{Z}_q$, adds $((m_\omega, R), h_i)$ into the hash table, and returns h_i to $\mathcal{A}'_{\mathcal{II}}$.

The H and H' hash queries are handled similarly by maintaining the corresponding hash tables.

Delegation signing queries: For each query m_ω chosen by $\mathcal{A}'_{\mathcal{II}}$, $\mathcal{B}$ performs the following steps:

(1) Randomly choose $c, s \in \mathbb{Z}^*_q$ and compute $R = sP - cY^*$.
(2) Set the hash value as $H_0(m_\omega, R) = c$ and store $((m_\omega, R), c)$ into the H_0 list.
(3) Return $\sigma_0 = (R, s)$ as the delegation signing key for m_ω.

When $\mathcal{A}'_{II}$ outputs a valid delegation signing key $s = r + x^*h$ where $h = H_0(m^*_\omega, R)$, based on the forking lemma, by rewinding $\mathcal{A}'_{II}$ and returning a new hash output h' for $H_0(m^*_\omega, R)$, $\mathcal{B}$ can obtain another valid delegation signing key $s' = r + x^*h'$. Therefore, $\mathcal{B}$ can output

$$x^* = \frac{s - s'}{h - h'} \bmod q$$

as the answer of the DL problem. □

3. ANONYMOUS PROXY SIGNATURE WITH INTERNAL TRACEABILITY

In this section, we consider a different scenario where only internal users, including the original signer and all the legitimate proxy signers, can perform the trace function. So any dishonest behaviours of an internal proxy signer will not be made public.

3.1. Definition

An anonymous proxy signature with internal traceability consists of the same set of algorithms as defined in the previous section with the following difference:

Trace ($\mathcal{TR}$): On input two message-signature pairs (m, σ) and (m', σ') with respect to the same tag L and warrant m_ω, **and a valid delegation signing key** σ_0 **for** m_ω, it outputs either 'indep', 'linked', or $Y_i \in Y_N$.

3.2. Security model

The security models for Unforgeability, Anonymity and Tag-linkability are the same as in the previous section. One distinguishing feature of internal traceability compared with public traceability is that even if a proxy signer signs twice with respect to the same tag, only the original signer and other legitimate proxy signers can perform the trace function, i.e. outsiders cannot trace a dishonest proxy signer.

3.2.1. Untraceability against outsider

It is defined via the following game.

Setup: The challenger $\mathcal{C}$ runs the key generation algorithm to obtain the secret key and public key pairs (x_0, Y_0), $(x_1, Y_1), \ldots, (x_n, Y_n)$ of the original signer and n proxy signers, respectively. $\mathcal{C}$ then sends $(Y_0, Y_1, \ldots, Y_n)$ to the adversary $\mathcal{A}$.

Key selection: The adversary $\mathcal{A}$ outputs a warrant m_ω, a tag $L = (issue, Y_N)$, and (Y_i, Y_j) where $i \neq j$ as the two target proxy signer's public keys. The challenger then randomly selects $b \in \{i, j\}$.

Proxy signing query: $\mathcal{A}$ may access three signing oracles: $\mathbf{Sig}_{psk_b}$, $\mathbf{Sig}_{psk_i}$ and $\mathbf{Sig}_{psk_j}$ for the warrant m_ω and the tag L where

(i) $\mathbf{Sig}_{psk_b}$ is the signing oracle with respect to proxy signer b (notice that $b \in \{i, j\}$) who has a valid proxy signing key psk_b;
(ii) $\mathbf{Sig}_{psk_i}$ (respectively, $\mathbf{Sig}_{psk_j}$) is the signing oracle with respect to proxy signer i (respectively, proxy signer j) who has a valid proxy signing key psk_i (respectively, psk_j).

Output: Finally, $\mathcal{A}$ outputs a guess b' for b. $\mathcal{A}$ wins the game if $b' = b$. Define the adversary's advantage as

$$\mathbf{Adv}_{\mathcal{A}}^{UT}(k) = \Pr[b' = b] - \tfrac{1}{2}.$$

3.3. Scheme 2 with internal traceability

We extend our Scheme 1 to achieve internal traceability. The challenge problem is to develop new techniques that could disallow outsiders to perform the trace operation. Our idea to solve the problem is to randomize each proxy signature so that only the original signer or a valid proxy signer who has obtained the same delegation signing key has the ability to de-randomize the proxy signature. Below are the details of our proposed scheme.

Parameter generation. Same as in Scheme 1 except that now the system parameters include two additional hash functions $\tilde{H}$ and $\hat{H}$.

Key generation. Same as in Scheme 1.

Delegation sign. The original signer first generates a warrant m_ω. Then it randomly chooses $\alpha \in \mathbb{Z}_q$ and computes $W_o = \alpha P$. After that, the proxy signer picks another random number $r \in \mathbb{Z}_q$ and computes $R = rP$, $s = r + x_0 H_0(m_\omega, R, W_o) \bmod q$. Finally, the proxy signer sends (m_ω, α, R, s) to all the proxy signers via a secure channel.

Delegation verification. Upon receiving (m_ω, α, R, s), the proxy signer u_i checks if $sP = R + H_0(m_\omega, R, W_o = \alpha P)Y_0$. If it does not hold, the delegation is rejected. Otherwise, the proxy signer u_i computes his/her proxy signing secret key $psk_i = s + x_i H_0(m_\omega, R, W_o) = r + H_0(m_\omega, R, W_o)(x_0 + x_i) \mod q$. For simplicity, let $pk_i = psk_i P = R + H_0(m_\omega, R, W_o)(Y_0 + Y_i)$ denote the corresponding proxy signing public key.

Proxy sign. To sign a message $m \in \{0, 1\}^*$ with respect to a tag $L = (issue, Y_N)$, where Y_N are public keys of the proxy signers described in the warrant m_ω, the real proxy signer u_i proceeds as follows:

(1) Randomly choose $\beta \in \mathbb{Z}_q$, compute $F = H(L)$, $W_p = (W_{p1}, W_{p2}) = (\alpha\beta P, \beta F)$ and $\sigma_i = \alpha\beta F + psk_i F = (\alpha\beta + psk_i)F$.
(2) Set $A_0 = H'(L, m)$ and $A_1 = (1/i)(\sigma_i - A_0)$.
(3) For all $j \neq i$, compute $\sigma_j = A_0 + jA_1 \in G$. Note that every $(j, \log_F(\sigma_j))$ is on the line defined by $(0, \log_F(A_0))$ and $(i, psk_i + \alpha\beta)$.
(4) Generate (c_N, z_N) based on a (non-interactive) zero-knowledge proof of knowledge for the language

$$\mathcal{L} = \{(L, F, \sigma_N) \mid \exists i \in N \; s.t. \log_P(pk_i') = \log_F(\sigma_i)\},$$

where $\sigma_N = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ and $pk_i' = W_{p1} + pk_i = (\alpha\beta + psk_i)P$ as follows:

(a) Pick random $\omega_i \leftarrow \mathbb{Z}_q$ and set $a_i = \omega_i P, b_i = \omega_i F \in G$.
(b) Pick random $z_j, c_j \leftarrow \mathbb{Z}_q$, and set $a_j = z_j P + c_j pk_j'$, $b_j = z_j F + c_j \sigma_j \in G$ for every $j \neq i$.
(c) Set $c = H''(L, m, A_0, A_1, W_o, W_p, a_N, b_N)$ where $a_N = (a_1, \ldots, a_n)$ and $b_N = (b_1, \ldots, b_n)$.
(d) Set $c_i = c - \Sigma_{j \neq i} c_j \bmod q$ and $z_i = \omega_i - c_i(\alpha\beta + psk_i) \bmod q$.
(e) Return (c_N, z_N), where $c_N = (c_1, \ldots, c_n)$ and $z_N = (z_1, \ldots, z_n)$, as a proof for $\mathcal{L}$.

(5) Perform another (non-interactive) zero-knowledge proof of knowledge for

$$\mathcal{L}' = \{(F, W_{p2}, W_o, W_{p1}) \mid \log_{W_o} W_{p1} = \log_F W_{p2}\}$$

as follows:

(a) Pick random $\omega \leftarrow \mathbb{Z}_p$ and set $\tilde{a} = \omega W_o$, $\tilde{b} = \omega F \in G$.
(b) Set $\tilde{c} = \tilde{H}(L, m, A_0, A_1, W_o, W_p, \tilde{a}, \tilde{b})$.
(c) Set $\tilde{z} = \omega - \tilde{c}\beta$.
(e) Return $(\tilde{c}, \tilde{z})$ as a proof for $\mathcal{L}'$.

(6) Perform a third (non-interactive) zero-knowledge proof of knowledge for

$$\mathcal{L}'' = \{(P, W_o) \mid W_o = \alpha P\}$$

as follows:

(a) Pick random $\omega \leftarrow \mathbb{Z}_p$ and set $\hat{a} = \omega P \in G$.
(b) Set $\hat{c} = \hat{H}(L, m, A_0, A_1, W_o, W_p, \hat{a})$.
(c) Set $\hat{z} = \omega - \hat{c}\alpha$.
(e) Return $(\hat{c}, \hat{z})$ as a proof for $\mathcal{L}''$.

(7) Return $\sigma = (A_1, R, W_o, W_p, c_N, z_N, \tilde{c}, \tilde{z}, \hat{c}, \hat{z})$ as the signature on (L, m).

Verification. To verify a proxy signature $\sigma = (A_1, R, W_o, W_p, c_N, z_N, \tilde{c}, \tilde{z}, \hat{c}, \hat{z})$ on message m and tag L, check the following:

(1) Parse L as $(issue, Y_N)$, and compute $pk_i' = W_{p1} + pk_i = W_{p1} + R + H_0(m_\omega, R, W_o)(Y_0 + Y_i)$ for all $i \in N$.
(2) Set $F = H(L)$ and $A_0 = H'(L, m)$, and compute $\sigma_i = A_0 + iA_1 \in G$ for all $i \in N$.
(3) Compute $a_i = z_i P + c_i pk_i'$, $b_i = z_i F + c_i \sigma_i$, for all $i \in N$.
(4) Check that $H''(L, m, A_0, A_1, W_o, W_p, a_N, b_N) = \Sigma_{i \in N} c_i \bmod q$, where $a_N = (a_1, \ldots, a_n)$ and $b_N = (b_1, \ldots, b_n)$.
(5) Compute $\tilde{a} = \tilde{z} W_o + \tilde{c} W_{p1}$, $\tilde{b} = \tilde{z} F + \tilde{c} W_{p2}$.
(6) Check if $\tilde{H}(L, m, A_0, A_1, W_o, W_p, \tilde{a}, \tilde{b}) = \tilde{c}$.
(7) Compute $\hat{a} = \hat{z} P + \hat{c} W_o$.
(8) Check if $\hat{H}(L, m, A_0, A_1, W_o, W_p, \hat{a}) = \hat{c}$.
(9) If all the above checks are successful, outputs accept; otherwise, outputs reject.

Trace. To check the relation between (m, σ) and (m', σ') under the same warrant m_ω and the same tag L, where $\sigma = (A_1, R, W_o, W_p, c_N, z_N, \tilde{c}, \tilde{z}, \hat{c}, \hat{z})$ and $\sigma' = (A_1', R', W_o', W_p', c_N', z_N', \tilde{c}', \tilde{z}', \hat{c}', \hat{z}')$, check the following:

(1) Parse L as $(issue, Y_N)$. Set $F = H(L)$ and $A_0 = H'(L, m)$. Compute $\sigma_i = A_0 + iA_1 \in G$ for all $i \in N$. With the secret α, the original signer or any proxy signer specified in the warrant m_ω can compute $\widehat{\sigma_i} = \sigma_i - \alpha W_{p2} = \sigma_i - \alpha(\beta F) = psk_i F \in G$ for all $i \in N$. Do the same operation for σ' to get $\widehat{\sigma_i'}$ for all $i \in N$.
(2) For all $i \in N$, if $\widehat{\sigma_i} = \widehat{\sigma_i'}$, store Y_i in **TList**, where **TList** is initially empty.
(3) Output Y if Y is the only entry in **TList**; 'linked' if **TList** $= Y_N$; 'indep' otherwise.

3.4. Security analysis

Below we prove the unforgeability and untraceability against outsiders of Scheme 2.

THEOREM 3.1. *If there exists a type II adversary $\mathcal{A}_{II}$ who can forge a valid proxy signature, then we can construct another adversary $\mathcal{B}$ who can use $\mathcal{A}_{II}$ to solve the DL problem.*

Proof. The proof follows the same approach we have used in the proof of Theorem 2.1. However, the proof of Lemma 2.1 needs to be modified since the proxy signing key in Scheme 2 has a slightly different structure.

LEMMA 3.1. *If the adversary $\mathcal{A}_{II}$ can forge a valid signature in Scheme 2, then we can construct another adversary $\mathcal{A}_{II}'$ who can forge a valid delegation signing key.*

After $\mathcal{A}_{II}'$ obtains $(Y_0, Y_1, \ldots, Y_n, x_1, \ldots, x_n)$, it simulates the game for $\mathcal{A}_{II}$ as follows.

$\mathcal{A}_{II}'$ answers H_0, H, H' hash queries and the delegation signing queries by referring those queries to its own oracles. $\mathcal{A}_{II}'$ simulates H'' queries by maintaining a hash table for the hash function H'' and answers $\mathcal{A}_{II}$'s H'' queries as follows: whenever a message is queried, $\mathcal{A}_{II}'$ first checks if a record for this input exists in the table. If so, the same output is returned; otherwise, $\mathcal{A}_{II}'$ chooses a random number in $\mathbb{Z}_q$ as the hash output, and adds the hash input and output into the table, and finally returns the output to $\mathcal{A}_{II}$. $\mathcal{A}_{II}'$ uses the same way to simulate $\tilde{H}$ and $\hat{H}$ queries.

Finally, $\mathcal{A}_{II}$ outputs a valid proxy signature $\sigma^* = (A_1, R, W_o, W_p, c_N, z_N, \tilde{c}, \tilde{z}, \hat{c}, \hat{z})$, which is a valid signature for the message m and the warrant m_ω^*. Note that m_ω^* should not be queried in any delegation signing queries.

As in the proof of Lemma 1, due to the Proof of Knowledge protocol for the language $\mathcal{L}$, $\mathcal{A}_{II}'$ can obtain

$$psk_i' = psk_i + \alpha\beta \bmod q.$$

Similarly, due to the Proof of Knowledge protocols for the language $\mathcal{L}'$ and $\mathcal{L}''$, $\mathcal{A}_{II}'$ can obtain α and β. Then $\mathcal{A}_{II}'$ can derive $psk_i = psk_i' - \alpha\beta \bmod q$, and a valid delegation signing key $s = psk_i - x_i H_0(m_\omega^*, R, W_o)$.

LEMMA 3.2. *If there exists an adversary $\mathcal{A}_{II}'$ who can forge a valid delegation signing key, then we can construct another algorithm $\mathcal{B}$ who can solve the DL problem.*

The proof is the same as that of Lemma 2.2.

By combining Lemmas 3.1 and 3.2, we can directly obtain Theorem 3.1. □

DEFINITION 3.1 ((DDH Problem)). *Let (G, q, P) be defined as in the DL problem. Given $(P, xP, yP, zP) \in G^4$, decide whether $z = xy$. The advantage of an algorithm $\mathcal{A}$ to solve the DDH is defined as*

$$Adv_{\mathcal{A}}^{DDH}(k) = \Pr[\mathcal{A}(P, xP, yP, xyP) = 1] - \Pr[\mathcal{A}(P, xP, yP, rP) = 1],$$

where x, y, r are randomly chosen in $\mathbb{Z}_q$.

THEOREM 3.2. *If there exists an outsider adversary $\mathcal{D}$, who can break the untraceability of the proposed anonymous proxy signature scheme, we can construct another adversary $\mathcal{B}$ who can solve the DDH problem.*

Proof. If there exists an adversary $\mathcal{D}$ who can correctly guess b with an non-negligible advantage ϵ, we can construct another algorithm $\mathcal{B}$ that can solve the DDH problem. Let (P, aP, bP, zP) be a given instance of the DDH problem. We construct $\mathcal{B}$ as follows:

Setup: $\mathcal{B}$ generates the user public and private keys (Y_0, x_0), $(Y_1, x_1), \ldots, (Y_n, x_n)$ for the original signer and the proxy signers by running the key generation algorithm. $\mathcal{B}$ then gives all the public keys $Y_0, Y_1, \ldots, Y_n$ to the adversary $\mathcal{D}$.

Key selection: $\mathcal{D}$ outputs a warrant m_ω, a tag $L = (issue, Y_N)$, and (Y_i, Y_j) as the two target proxy signer's public keys. $\mathcal{B}$ then sets $W_o = aP$ and $F = H(L) = bP$, randomly selects $r \in \mathbb{Z}_q$ and computes $R = rP$ and $s = r + x_0 H_0(m_\omega, R, W_o)$. $\mathcal{B}$ also randomly selects $b \in \{i, j\}$, and answers $\mathcal{D}$'s queries as follows.

Hash queries: All the hash queries made by $\mathcal{D}$ are answered as in the previous proof where $\mathcal{B}$ maintains a hash table for each hash function.

Proxy signing queries: When $\mathcal{D}$ makes a proxy signing query to $\mathbf{Sig}_{psk_i}$ on message m, $\mathcal{B}$ randomly selects $\beta \in \mathbb{Z}_q$, and computes $W_p = (\beta W_o, \beta F)$ and $\sigma_i = \beta z P + psk_i F$. $\mathcal{B}$ generates A_0, A_1 and $\sigma_t (t \neq i)$ by following the proxy signing algorithm. $\mathcal{B}$ also simulates the NIZK proof for language $\mathcal{L}$ using the following simulator.

NIZK Simulator for $\mathcal{L}$:

(1) For all $i \in N$, uniformly pick up at random $z_i, c_i \in \mathbb{Z}_q$, and compute $a_i = z_i P + c_i pk'_i$, $b_i = z_i F + c_i \sigma_i \in G$.
(2) Set $H''(L, m, A_0, A_1, W_o, W_p, a_N, b_N)$ as $c := \Sigma_{i \in N} c_i$ where $a_N = (a_1, a_2, \ldots, a_n)$ and $b_N = (b_1, b_2, \ldots, b_n)$.
(3) Output (c_N, z_N), where $c_N = (c_1, \ldots, c_n)$ and $z_N = (z_1, \ldots, z_n)$.

Similarly, $\mathcal{B}$ also simulates the NIZK proof $(\tilde{c}, \tilde{z})$ for language $\mathcal{L}'$, and the NIZK proof $(\hat{c}, \hat{z})$ for language $\mathcal{L}''$. Finally, $\mathcal{B}$ returns $\sigma = (A_1, R, W_o, W_p, c_N, z_N, \tilde{c}, \tilde{z}, \hat{c}, \hat{z})$ to $\mathcal{D}$.

$\mathcal{B}$ also uses the same method to simulate signing query to $\mathbf{Sig}_{psk_j}$. Note that since $b \in \{i, j\}$, the signing queries to $\mathbf{Sig}_{psk_b}$ are simulated using either $\mathbf{Sig}_{psk_i}$ or $\mathbf{Sig}_{psk_j}$ based on the value of b.

Output: Finally, $\mathcal{D}$ outputs b'. If $b = b'$, $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

Analysis: if (P, aP, bP, zP) is a DDH tuple, then the simulation is identical to the original game, and hence the adversary $\mathcal{D}$ has probability $1/2 + \epsilon$ to guess b correctly. On the other hand, if (P, aP, bP, zP) is not a DDH tuple, i.e. z is a random element of $\mathbb{Z}_q$, then the simulation does not reveal any information of b, and hence $\mathcal{D}$ has probability 12 to guess b correctly. Therefore, the advantage of $\mathcal{B}$ to solve the DDH problem is at least ϵ. □

4. ANONYMOUS PROXY SIGNATURE WITH ORIGINAL SIGNER TRACEABILITY

In this section, we further restrict the traceability to ensure that only the original signer can trace dishonest proxy signers.

4.1. Definition

An anonymous proxy signature with original signer traceability consists of the same set of algorithms with the following differences.

Delegation sign ($\mathcal{DS}$): Given a warrant m_ω for n proxy signers, the original signer computes n delegation signing key $\{\sigma_i\}(1 \leq i \leq n)$ and securely distributes them to the respective proxy signers.

Trace ($\mathcal{TR}$): On input two message-signature pairs (m, σ) and (m', σ') with respect to the same tag L and warrant m_ω, and all the delegation signing keys $\{\sigma_i\}(1 \leq i \leq n)$ for m_ω, it outputs either 'indep', 'linked', or $Y_i \in Y_N$.

4.2. Security model

The only security model that needs to be changed is the untraceability model since now we require that only the original signer can trace dishonest proxy signers.

4.2.1. *Untraceability against proxy signers*

It is defined via the following game.

Setup: Unchanged.

Key selection: The adversary $\mathcal{A}$ outputs a warrant m_ω, a tag $L = (issue, Y_N)$, and (Y_i, Y_j) as the two target proxy signer's public keys. For each proxy signer $t \notin \{i, j\}$, the challenger sends to $\mathcal{A}$ the proxy secret key x_t and delegation signing key σ_t for m_ω. The challenger also randomly selects $b \in \{i, j\}$.

Proxy signing query: $\mathcal{A}$ may access three signing oracles: $\mathbf{Sig}_{psk_b}$, $\mathbf{Sig}_{psk_i}$ and $\mathbf{Sig}_{psk_j}$ for the warrant m_ω and the tag L where

(i) $\mathbf{Sig}_{psk_b}$ is the signing oracle with respect to proxy signer b (notice that $b \in \{i, j\}$) who has a valid proxy signing key psk_b;
(ii) $\mathbf{Sig}_{psk_i}$ (respectively, $\mathbf{Sig}_{psk_j}$) is the signing oracle with respect to proxy signer i (resp. proxy signer j) who has a valid proxy signing key psk_i (respectively, psk_j).

Output: Finally, $\mathcal{A}$ outputs a bit b'. $\mathcal{A}$ wins the game if $b' = b$. Define the advantage of the adversary as

$$\mathbf{Adv}_{\mathcal{A}}^{UT2}(k) = \Pr[b' = b] - \tfrac{1}{2}.$$

4.3. Scheme 3 with original signer traceability

In our Scheme 2, the original signer shares a single secret α with all the proxy signers. Therefore, not only the original signer but also all the proxy signers who have α can trace a dishonest proxy signer.

In order to ensure that only the original signer can perform the trace operation, we should use different tracing secrets rather than the same one for different proxy signers. One simple way to do so is to generate a unique α_i for each proxy signer. However, this simple solution brings a problem: the scheme is no longer anonymous against the original signer, who can use the information related to α_i (more precisely, $\alpha_i P$) in Scheme 2 to identify a proxy signer even if the proxy signer is honest. Below we propose an *oblivious key distribution protocol* to solve the problem. The protocol allows the original singer to distribute a unique delegation signing key to a legitimate proxy signer without knowing the identity of the proxy signer.

Delegation sign. The original signer first generates a warrant m_ω for n proxy signers. Then for each $1 \leq i \leq n$, the original signer randomly chooses $\alpha_i \in \mathbb{Z}_q$ and computes $W_i = \alpha_i P$. After that, the original signer picks another random number $r_i \in \mathbb{Z}_q$ and computes $R_i = r_i P$, $s_i = r_i + x_0 H_0(m_\omega, R_i, W_i) \bmod q$. The original signer then obliviously distributes the delegation signing keys $\{\sigma_i = (m_\omega, \alpha_i, W_i, R_i, s_i)\}_{1 \leq i \leq n}$ to the proxy signers $(u_1, u_2, \ldots, u_n)$ as follows.

(1) The original signer first sends the warrant m_ω, which includes the public keys Y_N of all the proxy signers, to each proxy signer. The original signer also maintains an encryption key table T, which is initially empty.
(2) Upon receiving the warrant, a proxy signer first generates an ephemeral key pair (pk_i, sk_i) for a secure public key encryption scheme (e.g. the Cramer–Shoup encryption scheme [21]). Then the proxy signer uses his/her long-term public/private key pair (Y_i, x_i) to generate a Fujisaki–Suzuki [2] traceable ring signature Δ_i for the message pk_i and the tag $L = (\text{'Key Delegation'}, Y_N)$. The proxy signer sends (pk_i, Δ_i) to the original singer.
(3) Upon receiving a message (pk_i, Δ_i) from an (unknown) proxy signer, the original signer first checks if pk_i exists in the table T. If not, the original signer verifies the validity of the ring signature. If the signature is valid, then the original singer runs the **Trace** algorithm of the Fujisaki–Suzuki traceable ring signature between (pk_i, Δ_i) and each entry (pk_j, Δ_j) in the table T. If the output of the **Trace** algorithm is 'indep' for each entry in the table T, (pk_i, Δ_i) is added into T. If any of the above checks fails, the original signer aborts the protocol.
(4) After receiving N public keys in the table T, the original signer encrypts each delegation signing key $C_i = \mathcal{E}(pk_i, \sigma_i)$ using the public key encryption scheme, and sends $\{(pk_i, C_i)\}(1 \leq i \leq N)$ to each proxy signer.
(5) The proxy signer then uses the decryption key generated in Step 2 to decrypt one of ciphertexts to obtain a delegation signing key.

The above delegation signing key distribution protocol achieves the following properties. First, each public key in the table T must be sent by a legitimate proxy signer due to the unforgeability of the Fujisaki–Suzuki traceable ring signature scheme. Secondly, the original signer cannot tell the owner of a public key pk_i due to the anonymity property of the traceable ring signature scheme. Thirdly, each proxy signer can only submit one public key due to the traceability of the traceable ring signature scheme. Therefore, after running the above protocol, each proxy signer can obtain one of the delegation signing keys without letting the original signer or other proxy signers know which one.

TABLE 1. The computation cost for n proxy signers.

Scheme	DS	DV	PS	PV	TR
1	1e	3e	$(5n-1)$e	$6n$e	$2n$e
2	2e	4e	$(5n+5)$e	$(6n+6)$e	$(2n+2)$e
3	$2n$e	4e	$(5n+5)$e	$(6n+6)$e	$(2n+2)$e

The rest of the algorithms are the same as in Scheme 2 except that we replace W_o by W_i, α by α_i, s by s_i, and R by R_i for proxy signer i. We omit the details of these algorithms here.

The security analysis also follows that for Scheme 2. Note that since now each proxy signer only knows his own secret tracing key α_i, they cannot trace other proxy signers. On the other hand, the original signer has all the secret tracing keys, and hence can trace any dishonest proxy signer.

5. EFFICIENCY ANALYSIS

We give an efficiency analysis for the computation cost of the proposed anonymous proxy signatures with different levels of traceability in Table 1. Our proposed schemes do not involve any expensive pairing operations, so we only count the number of modular exponentiations (denoted by e) in the Delegation Sign (DS), Delegation Verification (DV), Proxy Sign (PS), Proxy Verify (PV) and Trace (TR) operations. From the table, we can see that all the tree schemes have linear (i.e. $O(n)$) complexity in proxy sign, proxy verify and trace operations.

6. CONCLUSION

In this paper, we proposed three anonymous proxy signature schemes with different levels of (namely, public, internal and original signer) traceability. We also defined the formal security models and proved the security of our schemes under some standard assumptions.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] Mambo, M., Usuda, K. and Okamoto, E. (1996) Proxy Signatures for Delegating Signing Operation. *Proc. 3rd ACM Conf. on Computer and Communications Security, CCS'96*, New Delhi, India, March 14–16, pp. 48–57. ACM, New York.

[2] Fujisaki, E. and Suzuki, K. (2008) Traceable ring signature. *IEICE Trans.*, **91-A**, 83–93.

[3] Kim, S., Park, S. and Won, D. (1997) Proxy Signatures, Revisited. *Proc. 1st Int. Conf. on Information and Communication Security, ICICS'97*, Beijing, China, November 11–14, Lecture Notes in Computer Science 1334, pp. 223–232. Springer, London.

[4] Lee, B., Kim, H. and Kim, K. (2001) Strong Proxy Signature and its Applications. *Proc. 2001 Symposium on Cryptography and Information Security*, Oiso, Japan, January 23–26, pp. 603–608. IEICE.

[5] Zhang, F., Safavi-Naini, R. and Lin, C.-Y. (2004) Some New Proxy Signature Schemes from Pairings. *Progress on Cryptography*, pp. 59–66. Springer, USA.

[6] Xu, J., Zhang, Z. and Feng, D. (2005) Id-Based Proxy Signature using Bilinear Pairings. *Proc. Parallel and Distributed Processing and Applications—ISPA 2005 Workshops*, Nanjing, China, November 2–5, Lecture Notes in Computer Science 3759, pp. 359–367. Springer, Berlin.

[7] Wu, W., Mu, Y., Susilo, W., Seberry, J. and Huang, X. (2007) Identity-Based Proxy Signature from Pairings. *Proc. 4th Int. Conf. on Autonomic and Trusted Computing, ATC 2007*, Hong Kong, China, July 11–13, Lecture Notes in Computer Science 4610, pp. 22–31. Springer, Berlin.

[8] Zhang, K. (1997) Threshold Proxy Signature Schemes. *Proc. 1st Int. Workshop on Information Security, ISW'97*, Tatsunokuchi, Japan, September 17–19, Lecture Notes in Computer Science 1396, pp. 282–290. Springer, Berlin.

[9] Liu, J. and Huang, S. (2010) Identity-based threshold proxy signature from bilinear pairings. *Inform. Lith. Acad. Sci.*, **21**, 41–56.

[10] Zhang, F., Safavi-Naini, R. and Lin, C.-Y. (2003) New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairing. *IACR Cryptology ePrint Archive*, **2003**, 104.

[11] Mehta, M. and Harn, L. (2005) Efficient one-time proxy signatures. *IEE Proc., Commun.*, **152**, 129–133.

[12] Chaum, D. and van Heyst, E. (1991) Group Signatures. *Advances in Cryptology—Proc. EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques*, Brighton, UK, April 8–11, Lecture Notes in Computer Science 547, pp. 257–265. Springer, Berlin.

[13] Rivest, R.L., Shamir, A. and Tauman, Y. (2001) How to Leak a Secret. *Advances in Cryptology—ASIACRYPT 2001, Proc. 7th Int. Conf. on the Theory and Application of Cryptology and Information Security*, Gold Coast, Australia, December 9–13, Lecture Notes in Computer Science 2248, pp. 552–565. Springer, Berlin.

[14] Yang, G., Wong, D.S., Deng, X. and Wang, H. (2006) Anonymous Signature Schemes. *Public Key Cryptography—PKC 2006, Proc. 9th Int. Conf. on Theory and Practice of Public-Key Cryptography*, New York, NY, USA, April 24–26, Lecture Notes in Computer Science 3958, pp. 347–363. Springer, Berlin.

[15] Cheng, W., Lang, W., Yang, Z., Liu, G. and Tan, Y. (2004) An Identity-Based Proxy Ring Signature Scheme from Bilinear Pairings. *Proc. 9th IEEE Symposium on Computers and Communications* (*ISCC 2006*), Alexandria, Egypt, June 28–July 1, pp. 424–429. IEEE.

[16] Li, J., Chen, X., Yuen, T.H. and Wang, Y. (2006) Proxy Ring Signature: Formal Definitions, Efficient Construction and New Variant. *Proc. Int. Conf. on Computational Intelligence and Security*, Guangzhou, China, November 3–6, pp. 1259–1264. IEEE.

[17] Toluee, R., Asaar, M.R. and Salmasizadeh, M. (2012) An anonymous proxy signature scheme without random oracles. *IACR Cryptology ePrint Archive*, **2012**, 313.

[18] Kiayias, A., Tsiounis, Y. and Yung, M. (2004) Traceable Signatures. *Advances in Cryptology—EUROCRYPT 2004, Proc. Int. Conf. on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, May 2–6, Lecture Notes in Computer Science 3027, pp. 571–589. Springer, Berlin.

[19] Wei, J., Yang, G. and Mu, Y. (2014) Anonymous Proxy Signature with Restricted Traceability. *13th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, TrustCom 2014*, Beijing, China, September 24–26, pp. 575–581. IEEE.

[20] Schnorr, C. (1991) Efficient signature generation by smart cards. *J. Cryptol.*, **4**, 161–174.

[21] Cramer, R. and Shoup, V. (1998) A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. *Advances in Cryptology—CRYPTO'98, Proc. 18th Annual Int. Cryptology Conf.*, Santa Barbara, CA, USA, August 23–27, Lecture Notes in Computer Science 1462, pp. 13–25. Springer, Berlin.