A Revocable Group Signature Scheme with Scalability from Simple Assumptions and Its Application to Identity Management*

Keita Emura[§] and Takuya Hayashi[§]

[§]National Institute of Information and Communications Technology (NICT), Japan.
{k-emura, t-hayashi}@nict.go.jp

August 19, 2019

Abstract

Group signatures are signatures providing signer anonymity where signers can produce signatures on behalf of the group that they belong to. Although such anonymity is quite attractive considering privacy issues, it is not trivial to check whether a signer has been revoked or not. Thus, how to revoke the rights of signers is one of the major topics in the research on group signatures. In particular, scalability, where the signing and verification costs and the signature size are constant in terms of the number of signers N, and other costs regarding signers are at most logarithmic in N, is quite important. In this paper, we propose a revocable group signature scheme which is currently more efficient compared to previous all scalable schemes. Moreover, our revocable group signature scheme is secure under simple assumptions (in the random oracle model), whereas all scalable schemes are secure under q-type assumptions. We implemented our scheme by employing Barreto-Lynn-Scott curves of embedding degree 12 over a 455-bit prime field (`BLS-12-455`), and Barreto-Naehrig curves of embedding degree 12 over a 382-bit prime field (`BN-12-382`), respectively, by using the RELIC library. We showed that the online running times of our signing algorithm were approximately 14 msec (`BLS-12-455`) and 11 msec (`BN-12-382`), and those of our verification algorithm were approximately 20 msec (`BLS-12-455`) and 16 msec (`BN-12-382`), respectively. Finally, we showed that our scheme is applied to an identity management system proposed by Isshiki et al.

1 Introduction

1.1 Revocable Group Signature

Group signatures [26] have been widely recognized as an extension of digital signatures. In conventional signature schemes, a signer-specific public key is used for verifying signatures whereas in a group signature scheme, a group public key is used. Thus signers are anonymous since a verifier just verifies that a signer belongs to the group. Although such anonymity is quite attractive considering privacy issues, on the other hand, it makes it difficult to provide revocation function. Furthermore, how to revoke the rights of signers in group signatures is not trivial. In the early stage, either the signing cost or the verification cost depend on the number of revoked signers R. Camenisch and Lysyanskaya [25] proposed a method to revoke users by using accumulators, where the signing algorithm requires $O(R)$

[*] An extended abstract appears in the 21st Information Security Conference (ISC) 2018 [35]. This is the full version. We additionally considered weak opening soundness [71] and showed that our scheme can be applied to an identity management system [46]. See Section 6 for details. We also additionally gave the implementation result of the Judge algorithm.

computations. Boneh and Shacham [20] proposed group signatures with verifier-local revocation (VLR-GS). In VLR-GS [20, 53, 59, 58, 63, 62, 77, 75], no signer is involved to the revocation procedure. As a drawback, the verification cost is $O(R)$, and is not constant. Moreover, VLR group signature schemes provide a weaker security level called selfless anonymity.[1] In 2009, Nakanishi, Fujii, Hira, and Funabiki [61] broke this barrier by proposing a revocable group signature scheme with constant costs for signing and verifying. One drawback of their scheme is that the public key size is $O(\sqrt{N})$ where N is the maximal number of signers. Later, Fan, Hsu, and Manulis [39] proposed a revocable group signature scheme with not only constant signing/verification costs but also constant size public key. However, the size of the revocation list is $O(N)$. Slamanig et al. [73] proposed linking-based revocation, and gave an instantiation based on the Delerablée-Pointcheval group signature scheme [32] by employing a generic compiler [72]. They introduced a dedicated authority, which they call revocation authority (RA), that can extract a revocation token from signatures by using a secret linking key. By using a simple look-up operation (and cuckoo hashing), the constant-time revocation check is realized. However, signatures are not publicly verifiable in the sense that the revocation check requires the secret linking key. Recently, some VLR-type schemes realized sub-linear/constant verification costs [51, 69, 34, 36]. As a drawback, these schemes do not provide unlinkability, that is, they employed linkable parts contained in signatures for efficiently executing verification procedure.

A major breakthrough was implemented by Libert, Peters, and Yung (LPY) [56] in 2012. In the LPY scheme, a group manager periodically publishes a revocation list that contains ciphertexts of broadcast encryption which will be decrypted by non-revoked signers. A non-revoked signer proves the decryption ability of a ciphertext. The LPY scheme is scalable in the sense that it provides not only constant signing and verification costs, but also other costs regarding signers are at most logarithmic in N. They gave two schemes based on the complete subtree (CS) and the subset difference (SD) methods [65]. They further improved the efficiency of the LPY schemes by proposing a revocable group signature scheme with constant-size certification [55]. As followers of the LPY works, revocable group signatures with compact revocation list size were proposed [6, 7, 64, 70].

1.2 Actual Efficiency of Revocable Group Signatures with Scalability

Although the LPY scheme and other similar schemes are “asymptotically” very efficient, these schemes are not sufficiently efficient in practice. One reason for its inefficiency is that these schemes did not rely on random oracles, but rather employed the Groth-Sahai proofs [43]. Of course, avoiding random oracles and constructing schemes in the standard model are quite meaningful from a theoretical point of view. However, in general, random oracles yield efficient schemes, especially, in the group signature context. For example, the signature size of the LPY scheme is approximately 100 group elements, and even if we exclude revocation functionality, the signature size of the Groth scheme [42] and the (non-revocable) Libert-Peters-Yung scheme [57], which are recognized as the most efficient group signature schemes in the standard model, are approximately 50 group elements. On the other hand, we can construct (non-revocable) group signature schemes whose signature size are less than 10 group elements when random oracles are introduced, e.g., Boneh-Boyen-Shacham [19], Furukawa-Imai [41, 45], Delerablée-Pointcheval [32], Bichsel et al. [18], Pointcheval-Sanders [68], Derler-Slamanig [33], and Libert-Mouhartem-Peters-Yung [54]. [2]

In terms of the running time of signing and verification, Begum et al. [13] gave an implementation of the Nakanishi-Funabiki scheme [64], which is a revocable group signature with scalability secure in the standard model, where the running time of the signing algorithm and the verification algorithm are approximately 500 msec and 900 msec, respectively. They employed Barreto-Naehrig (BN) curves [12]

[1] Recently, Ishida et al. [44] showed that fully anonymous VLR group signatures can be realized. However, they employed general NIZK proofs and no concrete instantiation has been proposed so far.

[2] As an exception, very recently, Clarisse and Sanders [29] proposed an efficient (non-revocable) group signature scheme secure in the standard model. The signature size is just 5 group elements.

Table 1: Comparison of Revocable Group Signature Schemes with Scalability

Scheme	Public key Size	Signature Size†	Certificate Size	Revocation list Size	Signing Cost	Verification Cost	Std/ ROM§	Assumption
LPY1(CS) [56]	$O(1)$	$O(1)$ (96)	$O(\log N)$	$O(R \cdot \log(N/R))$	$O(1)$	$O(1)$	Std	q-Type
LPY2(SD) [56]	$O(\log N)$	$O(1)$ (96)	$O(\log^3 N)$	$O(R)$	$O(\log N)^\ddagger$	$O(1)$	Std	q-Type
LPY3 [55]	$O(\log N)$	$O(1)$ (144)	$O(1)$	$O(R)$	$O(1)$	$O(1)$	Std	q-Type
AEHS [6, 7]	$O(1)$	$O(1)$ (98)	$O(R_{\mathsf{max}})$	$O(1)$	$O(R)^\ddagger$	$O(1)$	Std	q-Type
NF [64]	$O(T \log N)$	$O(1)$ (143)	$O(T)$	$O(R/T)$	$O(T)^\ddagger$	$O(1)$	Std	q-Type
SN [70]	$O(T + \log N)$	$O(1)$ (299)	$O(1)$	$O(R/T)$	$O(T)^\ddagger$	$O(1)$	Std	q-Type
Ohara et al. [66]	$O(1)$	$O(1)$ (18)	$O(\log N)$	$O(R \cdot \log(N/R))$	$O(1)$	$O(1)$	ROM	q-Type
Our Scheme	$O(1)$	$O(1)$ (16)	$O(\log N)$	$O(R \cdot \log(N/R))$	$O(1)$	$O(1)$	ROM	Simple

N: The maximum number of group members.
R: The number of revoked signers.
R_{max}: The maximum number of revoked signers.
T: The parameter of the accumulated/vector commitment value in [64, 70].
† We denote the number of group elements contained in a group signature on (). This number contains both the number of $\mathbb{G}$ elements and $\mathbb{Z}_p$ elements.
‡ This complexity is only required at the first signature generation of each revocation epoch.
§ Standard Model / Random Oracle Model

over a 254-bit prime field and the embedding degree is 12, and utilized a library based on the "Cross-twisted χ-based Ate (Xt-Xate) pairing" [4]. On the other hand, Emura, Hayashi, and Ishida [37] proposed a group signature scheme with time-bound keys secure in the random oracle model, where each signing key is associated with an expiry time, and they showed that the running time of their signing and verification algorithms were less than 4 msec and 12 msec, respectively. They also employed the BN curves over a 254-bit prime field, and utilized the RELIC library [5]. Of course we cannot directly compare these two implementation results due to differences in the functionalities and the selection of the underlying elliptic curves and parameters. However, these results somewhat indicate that group signature schemes in the random oracle model are significantly more efficient than those in the standard model.

In actual usage, Intel Software Guard Extensions (SGX) [2] employs the Intel Enhanced Privacy Identification (EPID) scheme [1, 23], and the EPID scheme builds on top of the Boneh-Boyen-Shacham group signature scheme [19] and the Furukawa-Imai group signature scheme [41, 45]. These group signature schemes are secure in the random oracle model. Thus, improving efficency of revocable group signature schemes in the random oracle model seems meaningful for a practical usage. To the best of our knowledge, the Ohara et al. revocable group signature scheme [66] is the only scheme that provides scalability in the random oracle model. The costs of the Ohara et al. scheme are asymptotically the same as those of the CS-based LPY group signature scheme [56]. Moreover the Ohara et al. scheme is significantly more efficient than the LPY scheme due to the random oracle. For example, the signature size of the Ohara et al. scheme is 18 group elements whereas that of the LPY scheme is 98 group elements. One drawback of the Ohara et al. scheme is the underlying complexity assumptions, i.e., their scheme relies on a q-type assumption. Due to the Cheon attack [27], employing q-type assumption should be avoided as much as possible. Thus, proposing an efficient revocable group signature scheme with scalability from simple assumptions is still an open problem.

1.3 Our Contribution

In this paper, we propose a revocable group signature scheme with scalability from simple assumptions, and give its implementation results. Our scheme is more efficient than previous all scalable schemes. We summarized the efficiency of scalable schemes in Table 1. As in [56, 66], we only count arithmetic operations here. We employed the methodology proposed by Ohara et al. [66], where the group manager publishes a revocation list containing signatures of non-revoked signers, and a signer proves that a signature corresponding to the signer is contained in the revocation list. In addition to this, we employed the signature scheme proposed by Libert-Mouhartem-Peters-Yung (LMPY) [54] which is secure under a simple assumption. The signature size of the LMPY scheme is constant regardless of the number of message blocks due to the Kiltz-Wee quasi-adaptive non-interactive zero-knowledge (QA-NIZK) arguments for linear subspaces [50]. Libert et al. proposed a group signature scheme based on the LMPY signature scheme. Since the scheme does not provide revocation functionality, our scheme can also be seen as a modification of the Libert et al. group signature scheme by adding revocation functionality without additional complexity assumptions.

Finally, we implemented our scheme by employing Barreto-Lynn-Scott curves [11] of embedding degree 12 over a 455-bit prime field (`BLS-12-455`), and Barreto-Naehrig curves [12] of embedding degree 12 over a 382-bit prime field (`BN-12-382`), respectively, by using the RELIC library. We showed that the online running times of our signing algorithm were approximately 14 msec (`BLS-12-455`) and 11 msec (`BN-12-382`), and those of our verification algorithm were approximately 20 msec (`BLS-12-455`) and 16 msec (`BN-12-382`), respectively.

As differences from proceedings version [35], we additionally considered weak opening soundness [71] and showed that our scheme can be applied to an identity management system [46]. See Section 6 for details. We also additionally gave the implementation result of the Judge algorithm.

2 Preliminaries

2.1 Cryptographic Assumptions

In this subsection, we give the definitions of the Decisional Diffie-Hellman (DDH) assumption, the Symmetric eXternal Diffie-Hellman (SXDH) assumption, and the Symmetric Discrete Logarithm (SDL) assumption. Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\mathbb{G}$, $\widehat{\mathbb{G}}$, and $\mathbb{G}_T$ be groups with prime order $p > 2^\lambda$, and $e : \mathbb{G} \times \widehat{\mathbb{G}} \to \mathbb{G}_T$ be a bilinear map. For $g \in \mathbb{G}$ and $\widehat{h} \in \widehat{\mathbb{G}}$, $e(g, \widehat{h}) \neq 1_{\mathbb{G}_T}$ holds unless $g \neq 1_{\mathbb{G}}$ and $\widehat{h} \neq 1_{\widehat{\mathbb{G}}}$.

Definition 2.1 (DDH Assumption). *Let $a, b \xleftarrow{\$} \mathbb{Z}_p^*$ and $Z \xleftarrow{\$} \mathbb{G} \setminus \{g^{ab}\}$. We say that the DDH assumption holds in $\mathbb{G}$ if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{DDH}(\lambda) := |\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) \to true] - \Pr[\mathcal{A}(g, g^a, g^b, Z) \to true]|$ is negligible.*

Definition 2.2 (SXDH Assumption). *We say that the SXDH assumption holds if the DDH assumption holds in both $\mathbb{G}$ and $\widehat{\mathbb{G}}$.*

Definition 2.3 (SDL Assumption [54]). *Let $a \xleftarrow{\$} \mathbb{Z}_p^*$. We say that the SDL assumption holds in $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$ if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{SDL}(\lambda) := \Pr[\mathcal{A}(g, \widehat{g}, g^a, \widehat{g}^a) \to a]$ is negligible.*

2.2 QA-NIZK Arguments for Linear Subspaces

In this subsection, we introduce the Kiltz-Wee QA-NIZK arguments for linear subspaces [50] that prove membership in the row space of a matrix $\mathbf{M}$. As in [54], we assume that all algorithms take as input the description of common public parameters $\mathsf{cp} = (\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, p)$. In QA-NIZK proofs, the common reference string (CRS) may depend on the language to be proved. In the Kiltz-Wee case, it depends a matrix $\mathbf{M}$. As in [54], for soundness, $\mathbf{M}$ is required to be witness-samplable where the reduction has to know the discrete logarithms of the group elements of $\mathbf{M}$.

Bold capital letters, such as $\mathbf{M}$, denote matrices, and bold lowercase letters, such as $\mathbf{v}$, denote vectors. For $\mathbf{M} \in \mathbb{G}^{t \times n}$, we denote $\mathbf{M} = (M_{i,j})_{i \in [1,t], j \in [1,n]} = (\vec{M}_1, \ldots, \vec{M}_t)^{\mathrm{T}}$ where $M_{i,j} \in \mathbb{G}$ for $i \in [1, t]$ and $j \in [1, n]$ and $\vec{M}_i = (M_{i,1}, M_{i,2}, \ldots, M_{i,n})$ for $i \in [1, t]$.

$\mathsf{QA.KeyGen}(\mathsf{cp}, \mathbf{M})$: This CRS and trapdoor generation algorithm takes as input cp and a matrix $\mathbf{M}$ where $\mathbf{M} = (M_{i,j})_{i \in [1,t], j \in [1,n]} \in \mathbb{G}^{t \times n}$. Choose $\widehat{g}_z \xleftarrow{\$} \widehat{\mathbb{G}}$ and a trapdoor $\mathsf{tk} = (\chi_1, \ldots, \chi_n) \xleftarrow{\$} \mathbb{Z}_p^n$.

Compute $\widehat{g}_j = \widehat{g}_z^{\chi_j}$ for all $j \in [1,n]$. Compute $z_i = \prod_{j=1}^{n} M_{i,j}^{-\chi_j}$ for all $i \in [1,t]$. Output the common reference string $\mathsf{crs} = (\{z_i\}_{i=1}^{t}, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^{n}) \in \mathbb{G}^t \times \widehat{\mathbb{G}}^{n+1}$ and the trapdoor $\mathsf{tk} \in \mathbb{Z}_p^n$.

$\mathsf{Prove}(\mathsf{crs}, \mathbf{v}, \{\omega_i\}_{i=1}^{t})$**:** The proof generation algorithm takes as input $\mathsf{crs} = (\{z_i\}_{i=1}^{t}, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^{n})$, a vector $\mathbf{v}$, and witnesses $\{\omega_i\}_{i=1}^{t}$ where $\mathbf{v} = \vec{M}_1^{\omega_1} \cdot \vec{M}_2^{\omega_2} \cdots \vec{M}_t^{\omega_t} = (\prod_{i=1}^{t} M_{i,1}^{\omega_i}, \ldots, \prod_{i=1}^{t} M_{i,n}^{\omega_i})$ holds. Output a proof $\pi = \prod_{i=1}^{t} z_i^{\omega_i}$ which proves that $\mathbf{v}$ is a linear combination of the rows of $\mathbf{M}$.

$\mathsf{Sim}(\mathsf{tk}, \mathbf{v})$**:** The simulation algorithm takes as input $\mathsf{tk} = (\chi_1, \ldots, \chi_n)$ and $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{G}^n$, and output a simulated proof $\pi = \prod_{j=1}^{n} v_j^{-\chi_j}$.

$\mathsf{Verify}(\mathsf{crs}, \mathbf{v}, \pi)$**:** The verify algorithm takes as input $\mathsf{crs} = (\{z_i\}_{i=1}^{t}, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^{n})$, $\mathbf{v} = (v_1, \ldots, v_n)$, and π, and output 1 if $(v_1, \ldots, v_n) \neq (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}})$ and $1_{\mathbb{G}_T} = e(\pi, \widehat{g}_z) \prod_{j=1}^{n} e(v_j, \widehat{g}_j)$ hold, and 0 otherwise.

As we can see, $\prod_{j=1}^{n} e(v_j, \widehat{g}_j) = \prod_{j=1}^{n} e(v_j, \widehat{g}_z^{\chi_j}) = e(\prod_{j=1}^{n} v_j^{\chi_j}, \widehat{g}_z)$ hold. Since $\pi = \prod_{j=1}^{n} v_j^{-\chi_j}$, the equation above holds. For the sake of clarity, π is described as

$$\begin{aligned}
\pi &= \prod_{i=1}^{t}(\prod_{j=1}^{n} M_{i,j}^{-\chi_j})^{\omega_i} \\
&= \prod_{i=1}^{t}(M_{i,1}^{-\chi_1} M_{i,2}^{-\chi_2} \cdots M_{i,n}^{-\chi_n})^{\omega_i} \\
&= (M_{1,1}^{-\chi_1} M_{1,2}^{-\chi_2} \cdots M_{1,n}^{-\chi_n})^{\omega_1} \cdots (M_{t,1}^{-\chi_1} M_{t,2}^{-\chi_2} \cdots M_{t,n}^{-\chi_n})^{\omega_t} \\
&= (M_{1,1}^{\omega_1} M_{2,1}^{\omega_2} \cdots M_{t,1}^{\omega_t})^{-\chi_1} \cdots (M_{1,n}^{\omega_1} M_{2,n}^{\omega_2} \cdots M_{t,n}^{\omega_t})^{-\chi_n} \\
&= \prod_{j=1}^{n} v_j^{-\chi_j}
\end{aligned}$$

It is particularly worth noting that the proof size is constant regardless of the dimensions of the considered linear subspace.

2.3 The LMPY Signature Scheme

In this subsection, we introduce a signature scheme proposed by Libert-Mouhartem-Peters-Yung (LMPY) [54] which is unforgeable under the SXDH assumption. The signature scheme can efficiently sign block messages in $\mathbb{Z}_p^{\ell}$. By employing the Kiltz-Wee QA-NIZK arguments, the signature size is constant regardless of the number of blocks ℓ. Moreover, the verification algorithm requires just 5 pairings.

$\mathsf{Sig.KeyGen}(\lambda, \ell)$**:** The key generation algorithm takes as input a security parameter λ and the block size ℓ. Choose bilinear groups $\mathsf{cp} = (\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, p)$ where $p > 2^{\lambda}$ and $g \overset{\$}{\leftarrow} \mathbb{G}$ and $\widehat{g} \overset{\$}{\leftarrow} \widehat{\mathbb{G}}$. Choose $\omega, a \overset{\$}{\leftarrow} \mathbb{Z}_p$ and set $h = g^{\omega}$ and $\Omega = h^{\omega}$. Choose $\mathbf{v} = (v_1, \ldots, v_{\ell}, W) \overset{\$}{\leftarrow} \mathbb{G}^{\ell+1}$. For $g \in \mathbb{G}$ and the identity matrix $\mathbf{I}_{\ell+1}$, we denote $g^{\mathbf{I}_{\ell+1}}$ as the $(\ell+1) \times (\ell+1)$ matrix whose diagonal components are g and other all elements are $1_{\mathbb{G}}$. Let $\mathbf{1}_{\ell+1} = (1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}}) \in \mathbb{G}^{\ell+1}$. Set a matrix $\mathbf{M} \in \mathbb{G}^{(\ell+2)\times(2\ell+4)}$ as

$$\mathbf{M} = \begin{pmatrix} g & \mathbf{1}_{\ell+1} & \mathbf{1}_{\ell+1} & h \\ \mathbf{v}^{\mathrm{T}} & g^{\mathbf{I}_{\ell+1}} & h^{\mathbf{I}_{\ell+1}} & \mathbf{1}_{\ell+1}^{\mathrm{T}} \end{pmatrix}$$

Run $\mathsf{QA.KeyGen}(\mathsf{cp}, \mathbf{M})$, and get $\mathsf{crs} = (\{z_i\}_{i=1}^{\ell+2}, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^{2\ell+4})$ and $\mathsf{tk} \in \mathbb{Z}_p^{2\ell+4}$. Output the signing key $\mathsf{sk} = \omega$, and the verification key $\mathsf{vk} = (\mathsf{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega, \mathsf{crs})$.

$\mathsf{Sig.Sign}(\mathsf{vk}, \mathsf{sk}, \mathbf{m} = (m_1, \ldots, m_\ell))$**:** The signing algorithm takes as input $\mathsf{vk} = (\mathsf{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega, \mathsf{crs})$, $\mathsf{sk} = \omega$, and messages to be signed $\mathbf{m} = (m_1, \ldots, m_\ell)$. Choose $s \xleftarrow{\$} \mathbb{Z}_p$ and compute $\sigma_1 = g^\omega(v_1^{m_1} \cdots v_\ell^{m_\ell} \cdot W)^s$, $\sigma_2 = g^s$, and $\sigma_3 = h^s$. Set $\mathbf{v} = (\sigma_1, \sigma_2^{m_1}, \ldots, \sigma_2^{m_\ell}, \sigma_2, \sigma_3^{m_1}, \ldots, \sigma_3^{m_\ell}, \sigma_3, \Omega) \in \mathbb{G}^{2\ell+4}$, and run the Prove algorithm to generate a proof π where $\mathbf{v}$ is in the row space of $\mathbf{M}$. The QA-NIZK proof $\pi \in \mathbb{G}$ is described as $\pi = z_1^\omega(z_2^{m_1} \cdots z_{\ell+1}^{\ell} \cdot z_{\ell+2})^s$. Output the signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, \pi) \in \mathbb{G}^4$.

$\mathsf{Sig.Verify}(\mathsf{vk}, \sigma, \mathbf{m})$**:** The verification algorithm takes as input $\mathsf{vk} = (\mathsf{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega, \mathsf{crs})$, $\sigma = (\sigma_1, \sigma_2, \sigma_3, \pi)$, and $\mathbf{m} = (m_1, \ldots, m_\ell)$. Output 1 if the following holds, and 0 otherwise.

$$\begin{aligned}&e(\Omega, \widehat{g}_{2\ell+4})^{-1}\\ =&e(\pi, \widehat{g}_z)e(\sigma_1, \widehat{g}_1)e(\sigma_2, \widehat{g}_2^{m_1} \cdots \widehat{g}_{\ell+1}^{m_\ell} \cdot \widehat{g}_{\ell+2})e(\sigma_3, \widehat{g}_{\ell+3}^{m_1} \cdots \widehat{g}_{2\ell+2}^{m_\ell} \cdot \widehat{g}_{2\ell+3})\end{aligned}$$

The following theorem was given in [54].

Theorem 2.1 ([54])**.** *The LMPY signature scheme is existentially unforgeable under chosen-message attacks (EUF-CMA) if the SXDH assumption holds in* $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$*.*

We remark that a signature is publicly re-randomizable, i.e., for a valid signature-message pair $\sigma = (\sigma_1, \sigma_2, \sigma_3, \pi)$ and $\mathbf{m} = (m_1, \ldots, m_\ell)$, and a randomness $s \in \mathbb{Z}_p$,

$$(\sigma_1 \cdot (v_1^{m_1} \cdots v_\ell^{m_\ell} \cdot W)^s, \sigma_2 \cdot g^s, \sigma_3 \cdot h^s, \pi \cdot (z_2^{m_1} \cdots z_{\ell+1}^{\ell} \cdot z_{\ell+2})^s)$$

is also a valid signature on $\mathbf{m}$.

2.4 Sigma Protocols

Sigma protocols are three-move honest-verifier zero-knowledge protocols. First the prover takes as input a statement and a witness, and sends a commitment com to the verifier. Next, the verifier, that also takes as input the statement, sends a challenge chall to the prover. Finally, the prover sends a response resp to the verifier. We require special soundness, where given two accepted transcripts $(\mathsf{com}_1, \mathsf{chall}_1, \mathsf{resp}_1)$ and $(\mathsf{com}_2, \mathsf{chall}_2, \mathsf{resp}_2)$ with the condition $\mathsf{com}_1 = \mathsf{com}_2$ and $\mathsf{chall}_1 \neq \mathsf{chall}_2$, there is an extractor $\mathsf{Extract}$ that takes as input a statement s and two transcripts above, and outputs a witness ω that satisfies $L(\mathsf{s}, \omega) = 1$. Moreover, we also require special honest verifier zero knowledge (SHVZK), where there is a simulator Sim that takes as input a statement s and a challenge chall, and outputs a transcript $(\mathsf{com}, \mathsf{chall}, \mathsf{resp})$ that is indistinguishable from transcripts produced by the prover and the verifier as above. See [54] for the formal definition.

Moreover, we require sigma protocols to have quasi unique responses. Informally, for a statement s, and first two moves of the protocol, com and chall, no adversary can find responses resp and resp' which are both accepted but $\mathsf{resp} \neq \mathsf{resp}'$. If the success probability is zero, then it is called unique responses, or it is also known as strict soundness [74]. See [40] for the formal definition. Faust et al. [40] proved that if a sigma protocol has quasi unique responses, then the NIZK proof system derived via the Fiat-Shamir transformation is simulation-sound. Simulation soundness guarantees that even after seeing accepting proofs produced by the simulator, for both true and false statements, soundness holds. In our group signature scheme, the underlying NIZK proof system is required to be simulation sound for providing CCA anonymity where an adversary is allowed to access the opening oracle.

2.5 Complete Subtree Method

In this section, we introduce the Complete Subtree (CS) method [65]. Let $\mathcal{N}$ be the set of all signers, and $\mathcal{R} \subset \mathcal{N}$ be the set of revoked signers. By using the CS method, $\mathcal{N} \setminus \mathcal{R}$ is divided into $\mathtt{num}$ disjoint sets such as $\mathcal{N} \setminus \mathcal{R} = S_1 \cup \cdots \cup S_{\mathtt{num}}$, and $\mathtt{num} = O(R \cdot \log(N/R))$ where $N = |\mathcal{N}|$ and $R = |\mathcal{R}|$.

Definition 2.4 (CS Algorithm)**.** *The CS algorithm takes as input a binary tree* BT *and a set of revoked signers* $\mathcal{R}_t$ *where* $i \in \mathcal{R}_t$ *when a signer with index* i *is revoked at time* t*, and outputs a set of nodes. The description of* CS *is given below.*

```
$\mathsf{CS}(\mathsf{BT}, \mathcal{R}_t)$ :
    $\mathtt{X}, \mathtt{Y} \leftarrow \emptyset$;
    $\forall i \in \mathcal{R}_t$
        Add $\mathsf{Path}(i)$ to $\mathtt{X}$;
    $\forall x \in \mathtt{X}$
        If $x_{\mathrm{left}} \notin \mathtt{X}$ then add $x_{\mathrm{left}}$ to $\mathtt{Y}$;
        If $x_{\mathrm{right}} \notin \mathtt{X}$ then add $x_{\mathrm{right}}$ to $\mathtt{Y}$;
    If $|RL_t| = 0$ then add $\mathtt{root}$ to $\mathtt{Y}$;
    Return $\mathtt{Y}$;
```

3 Revocable Group Signatures

In this section, we give the syntax and correctness definitions of revocable group signature. We use the LPY definitions [56, 55] which are modified from the Kiayias-Yung (KY) model [48, 47] to match the revocation functionality. Lifetime is divided into revocation epochs. At the beginning of each revocation epoch t, the group manager (GM) publishes an revocation list RL_t.

A revocable group signature scheme $\mathcal{R}\text{-}\mathcal{GS}$ consists of 6 algorithms ($\mathsf{Setup}, \mathsf{Join}, \mathsf{Revoke}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Open}$) as follows:

Definition 3.1 (Revocable Group Signature)**.**

$\mathsf{Setup}(1^\lambda, N)$**:** *The setup algorithm takes as inputs a security parameter* $\lambda \in \mathbb{N}$ *and a maximal number of members* $N \in \mathbb{N}$*, and outputs a group public key* gpk*, the GM private key for revocation* $\mathcal{S}_{\mathsf{GM}}$*, and the opening authority (OA) private key for opening* $\mathcal{S}_{\mathsf{OA}}$*. Moreover, the algorithm initializes a public state* St *comprising a set data structure* $St_{\mathsf{users}} = \emptyset$ *and a string data structure* $St_{\mathsf{trans}} = \epsilon$*.*

$\mathsf{Join}^{\mathsf{GM},\mathcal{U}_i}$**:** *The interactive protocol for joining between GM and a signer* $\mathcal{U}_i$ *(whose identity is* ID_i*) involves two interactive Turing machines* $\mathsf{J}_{\mathsf{user}}$ *and* J_{GM} *which execution is denoted as* $[\mathsf{J}_{\mathsf{user}}(\mathsf{gpk}), \mathsf{J}_{\mathsf{GM}}(St, \mathsf{gpk}, \mathcal{S}_{\mathsf{GM}})]$*.* $\mathcal{U}_i$ *obtains a membership secret* sec_i *and a membership certificate* cert_i*. We assume that* ID_i *is contained in* sec_i*. If the protocol is successfully done, GM updates* $St_{\mathsf{users}} \leftarrow St_{\mathsf{users}} \cup \{\mathsf{ID}_i\}$ *and* $St_{\mathsf{trans}} \leftarrow St_{\mathsf{trans}} || \langle i, \mathsf{transcript}_i \rangle$*.*

$\mathsf{Revoke}(\mathsf{gpk}, \mathcal{S}_{\mathsf{GM}}, t, \mathcal{R}_t \subset St_{\mathsf{users}})$**:** *The revocation algorithm takes as input* gpk*,* $\mathcal{S}_{\mathsf{GM}}$*, a revocation epoch* t*, and a set of revoked signers* $\mathcal{R}_t \subset St_{\mathsf{users}}$*, and outputs an updated revocation list* RL_t *which contains* $\mathcal{R}_t$*.*

$\mathsf{Sign}(\mathsf{gpk}, t, RL_t, \mathsf{cert}, \mathsf{sec}, M)$**:** *The signing algorithm takes as input* gpk*, a time* t*,* RL_t*,* cert*,* sec*, and a message* M *to be signed, and outputs* $\perp$ *if* $\mathsf{ID} \in \mathcal{R}_t$*, and a group signature* Σ*, otherwise.*

$\mathsf{Verify}(\mathsf{gpk}, t, RL_t, \Sigma, M)$**:** *The verification algorithm takes as input* gpk, t, RL_t, Σ, *and* M, *and outputs 1 or 0 which mean valid or invalid, respectively.*

$\mathsf{Open}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t, \Sigma, M, St)$**:** *The opening algorithm takes as input* gpk, $\mathcal{S}_{\mathsf{OA}}$, t, Σ, M, *and* $St := (St_{\mathsf{users}}, St_{\mathsf{trans}})$, *and outputs* $\perp$ *if the opening is failure, and* i *such that* $\mathsf{ID}_i \in St_{\mathsf{users}} \cup \{\perp\}$, *otherwise.*

Next, we define correctness. Let St be a public state, and St is said to be valid if it can be reached from $St = (\emptyset, \epsilon)$ by a Turing machine having oracle access to J_{GM}. A state St' is said to be extended another state St if it can be reached from St. As in [47, 48, 55, 56] we use the notation $\mathsf{cert}_i \rightleftharpoons_{\mathsf{gpk}} \mathsf{sec}_i$ to express that there exist coin tosses ϖ for J_{GM} and $\mathsf{J}_{\mathsf{user}}$ such that, for some valid state St', the execution of $[\mathsf{J}_{\mathsf{user}}(\mathsf{gpk}), \mathsf{J}_{\mathsf{GM}}(St, \mathsf{gpk}, \mathcal{S}_{\mathsf{GM}})](\varpi)$ provides $\mathsf{J}_{\mathsf{user}}$ with $\langle i, \mathsf{cert}_i, \mathsf{sec}_i\rangle$.

Definition 3.2 (Correctness). *We say that a revocable group signature scheme* $\mathcal{R}\text{-}\mathcal{GS}$ *is correct if:*

1. *In a valid state* $St = (St_{\mathsf{users}}, St_{\mathsf{trans}})$, *the condition* $|St_{\mathsf{users}}| = |St_{\mathsf{trans}}|$ *holds, and no two entries of* St_{trans} *can contain certificates with the same tag.*
2. *If* $[\mathsf{J}_{\mathsf{user}}(\mathsf{gpk}), \mathsf{J}_{\mathsf{GM}}(St, \mathsf{gpk}, \mathcal{S}_{\mathsf{GM}})]$ *is honestly run by both parties and* $\langle i, \mathsf{cert}_i, \mathsf{sec}_i\rangle$ *is obtained by* $\mathsf{J}_{\mathsf{user}}$, *then* $\mathsf{cert}_i \rightleftharpoons_{\mathsf{gpk}} \mathsf{sec}_i$ *holds.*
3. *For each* t *and any* $\langle i, \mathsf{cert}_i, \mathsf{sec}_i\rangle$ *satisfying condition 2,*

$$\mathsf{Verify}(\mathsf{gpk}, t, RL_t, \mathsf{Sign}(\mathsf{gpk}, t, RL_t, \mathsf{cert}, \mathsf{sec}, M), M) = 1$$

   holds if $i \notin \mathcal{R}_t$.

4. *For any* $\langle i, \mathsf{cert}_i, \mathsf{sec}_i\rangle$ *resulting from the interaction* $[\mathsf{J}_{\mathsf{user}}(\cdot,\cdot), \mathsf{J}_{\mathsf{GM}}(\cdot, St, \cdot, \cdot)]$ *for some valid state* St, *any* t *s.t.* $i \notin \mathcal{R}_t$, $\mathsf{Open}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t, \Sigma, M, St) = i$ *holds where* $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, t, RL_t, \mathsf{cert}, \mathsf{sec}, M)$.

We introduce three security definitions, misidentification, non-frameability, and anonymity. Before that, we introduce variables and oracles as follows:

- $\mathsf{state}_{\mathcal{I}}$: This is a data structure which is initialized as $\mathsf{state}_{\mathcal{I}} = (St, \mathsf{gpk}, \mathcal{S}_{\mathsf{GM}}, \mathcal{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(1^\lambda, N)$. This structure represents the state of the interface as the adversary invokes the various oracles, and includes a counter t which indicates the number of signer revocation queries so far (i.e., the current revocation epoch).
- $n = |St_{\mathsf{users}}|$: This is the current cardinality of the group.
- Sigs: This is a set of signatures Sigs created by the signing oracle. Each entry is represented as (i, t, M, Σ), where Σ is a group signature on M signed by $\mathcal{U}_i$ at t.
- U^a: This is the set of corrupted signers who were introduced by the adversary $\mathcal{A}$ via an execution of the join protocol.
- U^b: This is the set of honest signers who were added in the system by the join protocol with the adversary $\mathcal{A}$ who acts a dishonest GM. $\mathcal{A}$ can obtain the transcript of the join protocol, but $\mathcal{A}$ cannot obtain sec.
- Q_{pub}, Q_{keyGM}, and Q_{keyOA}: When these oracles are invoked, the interface looks up $\mathsf{state}_{\mathcal{I}}$, and returns gpk, $\mathcal{S}_{\mathsf{GM}}$, or $\mathcal{S}_{\mathsf{OA}}$, respectively.
- $Q_{\mathsf{a\text{-}join}}$: This is the join oracle for a corrupted signer. On behalf of GM, the interface runs J_{GM} in interaction with $\mathsf{J}_{\mathsf{user}}$ which is run by the adversary. If this protocol successfully ends, the interface increments $n \leftarrow n + 1$, add ID_n to U^a, and updates St s.t. $St_{\mathsf{users}} \leftarrow St_{\mathsf{users}} \cup \{\mathsf{ID}_n\}$ and $St_{\mathsf{trans}} \leftarrow St_{\mathsf{trans}} || \langle n, \mathsf{transcript}_n\rangle$.

- $Q_{\mathsf{b\text{-}join}}$: This is the join oracle for an honest signer. On behalf of a signer, the interface runs $\mathsf{J}_{\mathsf{user}}$ in interaction with J_{GM} which is run by the adversary. If this protocol successfully ends, the interface increments $n \leftarrow n+1$, add ID_n to U^b, and updates St s.t. $St_{\mathsf{users}} \leftarrow St_{\mathsf{users}} \cup \{\mathsf{ID}_n\}$ and $St_{\mathsf{trans}} \leftarrow St_{\mathsf{trans}}||\langle n, \mathsf{transcript}_n\rangle$. Moreover, the interface stores cert_n and sec_n in a private part of $\mathsf{state}_{\mathcal{I}}$.

- Q_{sig}: This is the signing oracle. Given (i, M), the interface checks whether the private area of $\mathsf{state}_{\mathcal{I}}$ contains $(\mathsf{cert}_i, \mathsf{sec}_i)$ or not, and also checks $i \notin \mathcal{R}_t$, where t is the current revocation epoch. In no such $(\mathsf{cert}_i, \mathsf{sec}_i)$ with $i \notin \mathcal{R}_t$ exist or $\mathsf{ID}_i \notin U^b$, then return $\perp$. Otherwise, the interface runs $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, t, RL_t, \mathsf{cert}_i, \mathsf{sec}_i, M)$, updates $\mathsf{Sigs} \leftarrow \mathsf{Sigs}||(i, t, M, \Sigma)$, and returns Σ.

- Q_{open}: This is the opening oracle. Given (M, Σ), the interface runs $\mathsf{Open}(M, \Sigma, \mathsf{gpk}, t, \mathcal{S}_{\mathsf{OA}}, St)$ using the current state St, and returns its output result.

- $Q^{\neg S}_{\mathsf{open}}$: This is the restricted opening oracle. Let S be a set with the form (M, Σ, t). Given (M, Σ, t) the oracle returns the result of $\mathsf{Open}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t, \Sigma, M, St)$ if $(M, \Sigma, t) \notin S$.

- Q_{read} and Q_{write}: These are reading and writing oracles, respectively, in order to read/write $\mathsf{state}_{\mathcal{I}}$. Q_{read} outputs the whole $\mathsf{state}_{\mathcal{I}}$ but the public/private keys and the private part of $\mathsf{state}_{\mathcal{I}}$ where membership secrets are stored after $Q_{\mathsf{b\text{-}join}}$ queries. The adversary can modify $\mathsf{state}_{\mathcal{I}}$ via Q_{write} at will as long as it does not remove or alter elements of St_{users}, St_{trans}, or invalidate the public state St.

- Q_{revoke}: This is the revocation oracle. Given an index $i \in \mathbb{N}$ such that $\mathsf{ID}_i \in St_{\mathsf{users}}$, the interface checks whether ID_i is contained in the appropriate user set (i.e., either U^a or U^b) or not, and whether $\langle i, \mathsf{transcript}_i\rangle$ s.t. $i \notin \mathcal{R}_t$ is contained in St_{trans} or not, where t is the current revocation epoch. If not, then return $\perp$. Otherwise, the interface increments $t \leftarrow t+1$, adds i to $\mathcal{R}_t$, and updates RL_t. We assumed that the adversary only revokes one signer per query to Q_{revoke}. However, it can be easily extended to allow multiple signers revocation at once.

Moreover, we define the $\mathsf{IsRevoked}$ algorithm. This algorithm takes as input $(\mathsf{sec}, \mathsf{cert}, RL_t)$, and outputs 1 if a signer who has $(\mathsf{sec}, \mathsf{cert})$ is contained in RL_t, and 0 otherwise.

Next we introduce three security definitions, misidentification, non-frameability, and anonymity. Briefly, misidentification guarantees that no adversary (who does not have $\mathcal{S}_{\mathsf{GM}}$) can produce a valid group signature whose opening result is in outside of the set of non-revoked adversarially-controlled signers. Non-frameability guarantees that no adversary (who can corrupt GM and OA) can produce a group signature whose opening result is an honest signer. Anonymity guarantees that no adversary (who does not have $\mathcal{S}_{\mathsf{OA}}$) can distinguish whether signers of two group signatures are the same or not.

Definition 3.3 (Misidentification)**.** *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{state}_{\mathcal{I}} = (St, \mathsf{gpk}, \mathcal{S}_{\mathsf{GM}}, \mathcal{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(1^\lambda, R)$. $\mathcal{A}$ is allowed to access Q_{pub}, $Q_{\mathsf{a\text{-}join}}$, Q_{revoke}, Q_{read}, and Q_{keyOA}. Finally, $\mathcal{A}$ outputs (M^*, Σ^*). We say that $\mathcal{A}$ wins if (1) $\mathsf{Verify}(\mathsf{gpk}, t^*, RL_{t^*}, \Sigma^*, M^*) = 1$, where t^* is the challenge revocation epoch, and (2) for $i^* \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t^*, \Sigma^*, M^*, St')$, $i^* \notin U^a \setminus \mathcal{R}_{t^*}$. Let $\mathsf{Adv}^{\mathsf{mis-id}}_{\mathcal{A}}(\lambda) := \Pr[\mathcal{A}$ wins$]$. We say that $\mathcal{R}$-$\mathcal{GS}$ is secure against misidentification attack if for all PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{mis-id}}_{\mathcal{A}}(\lambda)$ is negligible.*

Definition 3.4 (Non-frameability)**.** *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{state}_{\mathcal{I}} = (St, \mathsf{gpk}, \mathcal{S}_{\mathsf{GM}}, \mathcal{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(1^\lambda, R)$. $\mathcal{A}$ is allowed to access Q_{pub}, Q_{KeyGM}, Q_{keyOA}, $Q_{\mathsf{b\text{-}join}}$, Q_{revoke}, Q_{sig}, Q_{read}, and Q_{write}. Finally, $\mathcal{A}$ outputs $(M^*, \Sigma^*, t^*, RL_{t^*})$. We say that $\mathcal{A}$ wins if (1) $\mathsf{Verify}(\mathsf{gpk}, t^*, RL_{t^*}, \Sigma^*, M^*) = 1$, and (2) for $i^* \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t^*, \Sigma^*, M^*, St')$, $i^* \in U^b$ and $(i^*, t^*, M^*, \Sigma^*) \notin \mathsf{Sigs}$. Let $\mathsf{Adv}^{\mathsf{nf}}_{\mathcal{A}}(\lambda) := \Pr[\mathcal{A}$ wins$]$. We say that $\mathcal{R}$-$\mathcal{GS}$ is non-frameable if for all PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{nf}}_{\mathcal{A}}(\lambda)$ is negligible.*

Definition 3.5 (Anonymity). *Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\mathsf{state}_{\mathcal{I}} = (St, \mathsf{gpk}, \mathcal{S}_{\mathsf{GM}}, \mathcal{S}_{\mathsf{OA}}) \leftarrow \mathsf{Setup}(1^\lambda, R)$. $\mathcal{A}$ is allowed to access Q_{pub}, Q_{KeyGM}, Q_{revoke}, Q_{open}, Q_{read}, and Q_{write}. $\mathcal{A}$ outputs $(aux, M^*, t^*, RL_{t^*}, (\mathsf{cert}_0^*, \mathsf{sec}_0^*), (\mathsf{cert}_1^*, \mathsf{sec}_1^*))$. For $d \in \{0,1\}$, if $(\mathsf{cert}_d^* \rightleftharpoons_{\mathsf{gpk}} \mathsf{sec}_d^*)$, $\mathsf{IsRevoked}(\mathsf{sec}_d^*, \mathsf{cert}_d^*, RL_{t^*}) = 0$, and $\mathsf{cert}_0^* \neq \mathsf{cert}_1^*$, then $\mathcal{C}$ chooses $b \overset{\$}{\leftarrow} \{0,1\}$, computes $\Sigma^* \leftarrow \mathsf{Sign}(t^*, RL_{t^*}, \mathsf{cert}_b^*, \mathsf{sec}_b^*, M^*)$, and sends Σ^* to $\mathcal{A}$. Then $\mathcal{A}$ is allowed to access Q_{pub}, Q_{KeyGM}, Q_{open}, Q_{read}, and Q_{write}, with one exception that $\mathcal{A}$ is not allowed to send (M^*, Σ^*, t^*) to Q_{open}. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$. Let $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(\lambda) := |\Pr[b = b'] - \frac{1}{2}|$. We say that $\mathcal{R}$-$\mathcal{GS}$ is anonymous if all PPT $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anon}}(\lambda)$ is negligible.*

4 Proposed Revocable Group Signature Scheme

In this section, we give our proposed revocable group signature scheme.

Employing the Ohara et al. Methodology for LMPY Signatures: We employed the methodology proposed by Ohara et al. [66], where the group manager publishes a revocation list containing signatures of non-revoked signers, and a signer proves that a signature corresponding to the signer is contained in the revocation list. More concretely, each signer is assigned to a leaf node of a binary tree. A signer, who has the identity ID, and whose path is $\{u_0, u_1, \ldots, u_\ell\}$, has LMPY signatures on messages (ID, u_j) as certificates for all $u_j \in \{u_0, u_1, \ldots, u_\ell\}$. Thus, each user needs to manage $O(\log N)$-size certificates. The revocation list at time t also contains LMPY signatures on messages (t, u'_j) for all $u'_j \in \{u'_0, u'_1, \ldots, u'_{\mathsf{num}}\}$ which is determined by the CS method. Thus, the size of revocation list is $O(R \cdot \log(N/R))$. If the signer is not revoked at t, then there exists a node u such that $u \in \{u_0, u_1, \ldots, u_\ell\} \cap \{u'_0, u'_1, \ldots, u'_{\mathsf{num}}\}$. The signer proves that the knowledge of two LMPY signatures on (ID, u) and (t, u'), and $u = u'$. Thus, we setup two LMPY signature schemes with $\ell = 2$ in our group signature scheme. Let $\mathsf{sk} = \omega$ and $\mathsf{vk} = (\mathsf{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega, \mathsf{crs})$, where $\mathsf{crs} = (\{z_i\}_{i=1}^4, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^8)$, are for the first scheme, and $\mathsf{sk}' = \omega'$ and $\mathsf{vk}' = (\mathsf{cp}, g', h', \widehat{g}', \mathbf{v}', \Omega', \mathsf{crs}')$, where $\mathsf{crs}' = (\{z'_i\}_{i=1}^4, \widehat{g}'_z, \{\widehat{g}'_j\}_{j=1}^8)$, are for the second scheme. The first scheme signs $\mathbf{m} = (\mathsf{ID}, u)$ where ID is the identity of a signer, and u is a node of the binary tree. Let $(\sigma_1, \sigma_2, \sigma_3, \pi)$ be its signature. The second scheme signs $\mathbf{m}' = (t, u)$ where t is the current time and u is a node of the binary tree. Let $(\sigma'_1, \sigma'_2, \sigma'_3, \pi')$ be its signature. Informally, for the NP language L_{LMPY} induced by the relation $R_{\mathsf{LMPY}}(\mathsf{vk}, \mathsf{vk}', (\widetilde{\sigma}, \mathbf{m}), (\widetilde{\sigma}', \mathbf{m}')) = 1$ iff $\mathsf{Sig.Verify}(\mathsf{vk}, \widetilde{\sigma}, \mathbf{m}) = 1$, $\mathsf{Sig.Verify}(\mathsf{vk}', \widetilde{\sigma}', \mathbf{m}') = 1$, and $m_2 = m'_2 (= u)$, the verifier accepts if $L_{\mathsf{LMPY}}(\mathsf{s}, (\widetilde{\sigma}, \mathbf{m}), (\widetilde{\sigma}', \mathbf{m}')) = 1$ where the statement s here is the verification equation of the LMPY scheme, and $((\widetilde{\sigma}_1, \widetilde{\pi}), (\mathsf{ID}, u))$ and $((\widetilde{\sigma}'_1, \widetilde{\pi}'), u)$ are witnesses. Here, the current time t is not required to be hidden, and is not a witness. We convert the sigma protocol via the Fiat-Shamir transformation.

High-level Description of the Signing Algorithm: We employ the Sign-then-Enc-then-Prove (SEP) paradigm [15] where LMPY signatures are encrypted and the validity of the signatures is proved without showing the signatures. In the signing algorithm, before encrypting LMPY signatures, the signer re-randomizes $(\sigma_1, \sigma_2, \sigma_3, \pi)$ and $(\sigma'_1, \sigma'_2, \sigma'_3, \pi')$, and let $\widetilde{\sigma} = (\widetilde{\sigma}_1, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\pi})$ and $\widetilde{\sigma}' = (\widetilde{\sigma}'_1, \widetilde{\sigma}'_2, \widetilde{\sigma}'_3, \widetilde{\pi}')$ be the signatures after re-randomization. Then, $(\widetilde{\sigma}_2, \widetilde{\sigma}_3)$ and $(\widetilde{\sigma}'_2, \widetilde{\sigma}'_3)$ are independent from the signed messages and other signatures. Thus, these can be directly included in a group signature, and we can reduce the size of group signature. Next, the signer encrypts (a part of) signatures $(\widetilde{\sigma}_1, \widetilde{\pi})$ and $(\widetilde{\sigma}'_1, \widetilde{\pi}')$ via the Cramer-Shoup encryption scheme [31] by using the public key of the group manager. In addition to these LMPY signatures, the signer also encrypts $V_{\mathsf{ID}} = v_1^{\mathsf{ID}}$ and $V_u = v_2^u$. The former is required to search the corresponding certificate $\mathsf{cert}_i = (i, \mathsf{Path}(i), V_{\mathsf{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \mathsf{Path}(i)})$ from joining transcripts in the open algorithm. The latter is also required to search j such that $V_u = V_u^{(j)} = v_2^{u_j}$ for obtaining $u_j \in \mathsf{Path}(i)$ in the open algorithm. We remark that, in the original Cramer-Shoup scheme, designated verifier NIZK proofs are employed for the validity check of ciphertexts. That is, the validity of ciphertexts can be checked by the decryptor who has the decryption key. Since group signatures are required to be publicly verifiable, as in [54, 66] we employ publicly verifiable NIZK proofs constructed from sigma protocols via the Fiat-Shamir transformation for the validity check. Moreover, for efficiency purposes, we employ the Cramer-Shoup encryption scheme with a randomness-reuse variant [14], and a randomness θ is re-used for encrypting plural messages. The signer proves the knowledge of ID, u, and θ, and also proves that two LMPY signatures sign on the same node u. The verification algorithm checks the validity of NIZK proofs. All procedures do not depend on the number of users, and thus we can achieve $O(1)$ signing/verification costs.

We give our revocable group signature scheme as follows.

$\mathsf{Setup}(1^\lambda, N)$**:** Choose bilinear groups $\mathsf{cp} = (\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, p)$ where $p > 2^\lambda$ and $g \overset{\$}{\leftarrow} \mathbb{G}$ and $\widehat{g} \overset{\$}{\leftarrow} \widehat{\mathbb{G}}$. Choose $g' \overset{\$}{\leftarrow} \mathbb{G}$ and $\widehat{g}' \overset{\$}{\leftarrow} \widehat{\mathbb{G}}$. Choose a hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ which is modeled as a random oracle.

1. *Generate Two Key Pairs of the LMPY Scheme*: Choose $\mathbf{v} = (v_1, v_2, W) \overset{\$}{\leftarrow} \mathbb{G}^3$ and $\mathbf{v}' = (v_1', v_2', W') \overset{\$}{\leftarrow} \mathbb{G}^3$. Choose $a, a' \overset{\$}{\leftarrow} \mathbb{Z}_p$, and compute $h = g^a$ and $h' = g'^{a'}$. Set $\ell = 2$ in the LMPY signature scheme, and set matrices $\mathbf{M}$ and $\mathbf{M}'$ as follows.

$$\mathbf{M} = \begin{pmatrix} g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h \\ v_1 & g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} \\ v_2 & 1_{\mathbb{G}} & g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h & 1_{\mathbb{G}} & 1_{\mathbb{G}} \\ W & 1_{\mathbb{G}} & 1_{\mathbb{G}} & g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h & 1_{\mathbb{G}} \end{pmatrix}$$

$$\mathbf{M}' = \begin{pmatrix} g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' \\ v_1' & g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} \\ v_2' & 1_{\mathbb{G}} & g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' & 1_{\mathbb{G}} & 1_{\mathbb{G}} \\ W' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' & 1_{\mathbb{G}} \end{pmatrix}$$

   Run $\mathsf{QA.KeyGen}(\mathsf{cp}, \mathbf{M})$ and $\mathsf{QA.KeyGen}(\mathsf{cp}, \mathbf{M}')$ of the QA-NIZK argument, and get $\mathsf{crs} = (\{z_i\}_{i=1}^4, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^8)$ and $\mathsf{crs}' = (\{z_i'\}_{i=1}^4, \widehat{g}_z', \{\widehat{g}_j'\}_{j=1}^8)$. Set $\mathsf{sk} = \omega$, $\mathsf{vk} = (\mathsf{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega = h^\omega, \mathsf{crs})$, $\mathsf{sk}' = \omega'$, and $\mathsf{vk}' = (\mathsf{cp}, g', h', \widehat{g}', \mathbf{v}', \Omega' = h'^{\omega'}, \mathsf{crs}')$.

2. *Generate a Key Pair of the Cramer-Shoup Scheme (with a Randomness-reuse Variant)*: Choose $x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x_z', y_z', x_\sigma', y_\sigma' \overset{\$}{\leftarrow} \mathbb{Z}_p$ and compute $X_z = g^{x_z} h^{y_z}$, $X_\sigma = g^{x_\sigma} h^{y_\sigma}$, $X_{ID} = g^{x_{ID}} h^{y_{ID}}$, $X_u = g^{x_u} h^{y_u}$, $X_z' = g^{x_z'} h^{y_z'}$, and $X_\sigma' = g^{x_\sigma'} h^{y_\sigma'}$.

3. Output $\mathsf{gpk} = (\mathsf{vk}, \mathsf{vk}', X_z, X_\sigma, X_{ID}, X_u, X_z', X_\sigma')$, $\mathcal{S}_{\mathsf{GM}} = (\mathsf{sk}, \mathsf{sk}')$, and $\mathcal{S}_{\mathsf{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x_z', y_z', x_\sigma', y_\sigma')$.

$\mathsf{Join}^{\mathsf{GM}, \mathcal{U}_i}$**:** A signer $\mathcal{U}_i$ and GM run the following interactive protocol.

1. $\mathcal{U}_i$ chooses $\mathsf{ID}_i \overset{\$}{\leftarrow} \mathbb{Z}_p$, computes $V_{\mathsf{ID}} = v_1^{\mathsf{ID}_i}$, $Z_{\mathsf{ID}} = z_2^{\mathsf{ID}_i}$, $\widehat{G}_{2,\mathsf{ID}} = \widehat{g}_2^{\mathsf{ID}_i}$, and $\widehat{G}_{5,\mathsf{ID}} = \widehat{g}_5^{\mathsf{ID}_i}$, and sends $(V_{\mathsf{ID}}, Z_{\mathsf{ID}}, \widehat{G}_{2,\mathsf{ID}}, \widehat{G}_{5,\mathsf{ID}})$ to GM.
2. If V_{ID} has been appeared in transcripts of St, then GM aborts. Otherwise, GM checks the following equations hold.

$$e(V_{\mathsf{ID}}, \widehat{g}_2) = e(v_1, \widehat{G}_{2,\mathsf{ID}}), e(Z_{\mathsf{ID}}, \widehat{g}_2) = e(z_2, \widehat{G}_{2,\mathsf{ID}})$$
$$e(V_{\mathsf{ID}}, \widehat{g}_5) = e(v_1, \widehat{G}_{5,\mathsf{ID}}),$$

   If all tests pass, GM samples a fresh index $i \in \mathbb{Z}_p$ and a fresh leaf node (then $\mathsf{Path}(i)$ is fixed), and sends i to $\mathcal{U}_i$. Otherwise, GM aborts.

3. *Prove the Knowledge of* ID_i: $\mathcal{U}_i$ runs an interactive zero-knowledge proof of knowledge of $\mathsf{ID}_i = \log_{v_1}(V_{\mathsf{ID}})$ in interaction with GM. We employ the Cramer-Damgård-MacKenzie transformation [30] which converts a sigma protocol into a perfect zero-knowledge proof of knowledge. We give the 4-round protocol in the Appendix.[3] Let $\pi_K(\mathsf{ID})$ be the interaction transcript.

4. *Generate LMPY Signatures as a Certificate*: For all $u_j \in \mathsf{Path}(i)$, GM chooses $s_j \xleftarrow{\$} \mathbb{Z}_p$ and computes a LMPY signature on messages (ID_i, u_j) by using $\mathsf{sk} = \omega$ such that $V_u^{(j)} = v_2^{u_j}$ and $\sigma_{j,1} = g^{\omega}(V_{\mathsf{ID}} \cdot V_u^{(j)} \cdot W)^{s_j}$, $\sigma_{j,2} = g^{s_j}$, $\sigma_{j,3} = h^{s_j}$, and $\pi_j = z_1^{\omega}(Z_{\mathsf{ID}} \cdot z_3^{u_j} \cdot z_4)^{s_j}$. Finally, GM sends $\mathsf{cert}_i = (i, \mathsf{Path}(i), V_{\mathsf{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \mathsf{Path}(i)})$ to $\mathcal{U}_i$.

5. Finally, GM stores $\mathsf{transcript}_i = ((Z_{\mathsf{ID}}, \widehat{G}_{2,\mathsf{ID}}, \widehat{G}_{5,\mathsf{ID}}), \pi_K(\mathsf{ID}), \mathsf{cert}_i)$ and $\mathcal{U}_i$ stores $(\mathsf{cert}_i, \mathsf{sec}_i)$ where $\mathsf{sec}_i = \mathsf{ID}_i$.

$\mathsf{Revoke}(\mathsf{gpk}, \mathcal{S}_{\mathsf{GM}}, t, \mathcal{R}_t \subset St_{\mathsf{users}})$**:** Run $Y \leftarrow \mathsf{CS}(\mathsf{BT}, \mathcal{R}_t)$. For all $u_j \in Y$, GM chooses $s_j \xleftarrow{\$} \mathbb{Z}_p$ and computes a LMPY signature on messages (t, u_j) by using $\mathsf{sk}' = \omega'$ such that $\sigma'_{j,1} = g'^{\omega'}(v_1'^{t} \cdot v_2'^{u_j} \cdot W')^{s_j}$, $\sigma'_{j,2} = g'^{s_j}$, $\sigma'_{j,3} = h'^{s_j}$, and $\pi'_j = z_1'^{\omega'}(z_2'^{t} \cdot z_3'^{u_j} \cdot z_4')^{s_j}$. Output $RL_t = (t, Y, \{(\sigma'_{j,1}, \sigma'_{j,2}, \sigma'_{j,3}, \pi'_j, u_j)\}_{u_j \in Y})$.

$\mathsf{Sign}(\mathsf{gpk}, t, RL_t, \mathsf{cert}_i, \mathsf{sec}_i, M)$**:** Let u be a node where $u \in \mathsf{Path}(i) \cap Y$. If u does not exist, then output $\perp$. Let $(\sigma_1, \sigma_2, \sigma_3, \pi)$ be a LMPY signature on (ID_i, u) contained in cert_i, and let $(\sigma'_1, \sigma'_2, \sigma'_3, \pi')$ be a LMPY signature on (t, u) contained in RL_t. Choose $s, s' \xleftarrow{\$} \mathbb{Z}_p$ and re-randomize signatures as follows.

$$\widetilde{\sigma}_1 = \sigma_1 \cdot (v_1^{\mathsf{ID}_i} \cdot v_2^{u} \cdot W)^s, \widetilde{\sigma}_2 = \sigma_2 \cdot g^s, \widetilde{\sigma}_3 = \sigma_3 \cdot h^s, \widetilde{\pi} = \pi \cdot (z_2^{\mathsf{ID}_i} \cdot z_3^{u} \cdot z_4)^s$$
$$\widetilde{\sigma}'_1 = \sigma'_1 \cdot (v_1'^{t} \cdot v_2'^{u} \cdot W')^{s'}, \widetilde{\sigma}'_2 = \sigma'_2 \cdot g'^{s'}, \widetilde{\sigma}'_3 = \sigma'_3 \cdot h'^{s'}, \widetilde{\pi}' = \pi' \cdot (z_2'^{t} \cdot z_3'^{u} \cdot z_4')^{s'}$$

Choose $\theta \xleftarrow{\$} \mathbb{Z}_p$ and compute a ciphertext of the Cramer-Shoup encryption $C_{\mathsf{Cramer\text{-}Shoup}} = (C_1, C_2, C_z, C_\sigma, C_{ID}, C_u, C'_z, C'_\sigma)$ such that

$$C_1 = g^\theta,\ C_2 = h^\theta,\ C_z = \widetilde{\pi} \cdot X_z^\theta,\ C_\sigma = \widetilde{\sigma}_1 \cdot X_\sigma^\theta$$
$$C_{ID} = v_1^{\mathsf{ID}_i} \cdot X_{ID}^\theta,\ C_u = v_2^u \cdot X_u^\theta, C'_z = \widetilde{\pi}' \cdot X_z'^{\theta},\ C'_\sigma = \widetilde{\sigma}'_1 \cdot X_\sigma'^{\theta}$$

Here, the randomness θ is re-used. Then, prove the knowledge of $(\mathsf{ID}_i, \theta, u)$. Namely, choose $r_{\mathsf{ID}}, r_\theta, r_u \xleftarrow{\$} \mathbb{Z}_p$, and compute $R_1 = g^{r_\theta}$, $R_2 = h^{r_\theta}$, $R_3 = v_1^{r_{\mathsf{ID}}} \cdot X_{\mathsf{ID}}^{r_\theta}$, $R_4 = {v_2}^{r_u} \cdot {X_u}^{r_\theta}$, and R_5 and R_6 such that

$$R_5 = \big(e(X_z, \widehat{g}_z)e(X_\sigma, \widehat{g}_1)\big)^{r_\theta}\big(e(\widetilde{\sigma}_2, \widehat{g}_2)e(\widetilde{\sigma}_3, \widehat{g}_5)\big)^{-r_{\mathsf{ID}}}\big(e(\widetilde{\sigma}_2, \widehat{g}_3)e(\widetilde{\sigma}_3, \widehat{g}_6)\big)^{-r_u}$$
$$R_6 = \big(e(X'_z, \widehat{g}'_z)e(X'_\sigma, \widehat{g}'_1)\big)^{r_\theta}\big(e(\widetilde{\sigma}'_2, \widehat{g}'_3)e(\widetilde{\sigma}'_3, \widehat{g}'_6)\big)^{-r_u}$$

Compute $c \leftarrow H(\mathsf{gpk}, t, C_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\sigma}'_2, \widetilde{\sigma}'_3, R_1, \ldots, R_6, M)$, $s_{\mathsf{ID}} = r_{\mathsf{ID}} + c \cdot \mathsf{ID}_i$, $s_\theta = r_\theta + c \cdot \theta$, and $s_u = r_u + c \cdot u$.[4]

Output a group signature $\Sigma = (C_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\sigma}'_2, \widetilde{\sigma}'_3, c, s_{\mathsf{ID}}, s_\theta, s_u) \in \mathbb{G}^{12} \times \mathbb{Z}_p^4$.

[3] As mentioned by Libert et al. [54], we can remove this interaction by using an extractable commitment. For example, Delerablée and Pointcheval [32] employed the Paillier encryption [67] as the underlying extractable commitment scheme. Then, the decryption key allows the extraction.

[4] We can easily see that the underlying sigma protocol has unique responses. Let all values, except $\mathsf{resp} = (s_{\mathsf{ID}}, s_\theta, s_u)$, be fixed. Then, assume that an accepted response $(s'_{\mathsf{ID}}, s'_\theta, s'_u) \neq (s_{\mathsf{ID}}, s_\theta, s_u)$ exists. Then, from $g^{s_\theta} \cdot C_1^{-c} = g^{s'_\theta} \cdot C_1^{-c}$, $s_\theta = s'_\theta$ holds. From $v_1^{s_{\mathsf{ID}}} \cdot X_{\mathsf{ID}}^{s_\theta} \cdot C_{ID}^{-c} = v_1^{s'_{\mathsf{ID}}} \cdot X_{\mathsf{ID}}^{s'_\theta} \cdot C_{ID}^{-c}$ and $s_\theta = s'_\theta$, $s_{\mathsf{ID}} = s'_{\mathsf{ID}}$ holds. From ${v_2}^{s_u} \cdot {X_u}^{s_\theta} \cdot C_u^{-c} = {v_2}^{s'_u} \cdot {X_u}^{s'_\theta} \cdot C_u^{-c}$ and $s_\theta = s'_\theta$, $s_u = s'_u$ holds. Thus, $(s'_{\mathsf{ID}}, s'_\theta, s'_u) = (s_{\mathsf{ID}}, s_\theta, s_u)$ holds and this shows that the sigma protocol has unique responses, and the NIZK proof system converted by the Fiat-Shamir transformation is simulation sound.

$\mathsf{Verify}(\mathsf{gpk}, t, RL_t, \Sigma, M)$**:** Compute $\bar{R}_1 = g^{s_\theta} \cdot C_1^{-c}$, $\bar{R}_2 = h^{s_\theta} \cdot C_2^{-c}$, $\bar{R}_3 = v_1^{s_{\mathsf{ID}}} \cdot X_{\mathsf{ID}}^{s_\theta} \cdot C_{ID}^{-c}$, $\bar{R}_4 = {v_2}^{s_u} \cdot {X_u}^{s_\theta} \cdot C_u^{-c}$, and $\bar{R}_5$ and $\bar{R}_6$ such that

$$\begin{aligned}\bar{R}_5 &= \big(e(X_z, \widehat{g}_z) e(X_\sigma, \widehat{g}_1)\big)^{s_\theta} \big(e(\widetilde{\sigma}_2, \widehat{g}_2) e(\widetilde{\sigma}_3, \widehat{g}_5)\big)^{-s_{\mathsf{ID}}} \big(e(\widetilde{\sigma}_2, \widehat{g}_3) e(\widetilde{\sigma}_3, \widehat{g}_6)\big)^{-s_u} \\ &\quad \times \big(e(C_z, \widehat{g}_z) e(C_\sigma, \widehat{g}_1) e(\widetilde{\sigma}_2, \widehat{g}_4) e(\widetilde{\sigma}_3, \widehat{g}_7) e(\Omega, \widehat{g}_8)\big)^{-c} \\ \bar{R}_6 &= \big(e(X'_z, \widehat{g}'_z) e(X'_\sigma, \widehat{g}'_1)\big)^{s_\theta} \big(e(\widetilde{\sigma}'_2, \widehat{g}'_3) e(\widetilde{\sigma}'_3, \widehat{g}'_6)\big)^{-s_u} \\ &\quad \times \big(e(C'_z, \widehat{g}'_z) e(C'_\sigma, \widehat{g}'_1) e(\widetilde{\sigma}'_2, {\widehat{g}'_2}{}^t \cdot \widehat{g}'_4) e(\widetilde{\sigma}'_3, {\widehat{g}'_5}{}^t \cdot \widehat{g}'_7) e(\Omega', \widehat{g}'_8)\big)^{-c}\end{aligned}$$

Output 1 if $c = H(\mathsf{gpk}, t, C_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\sigma}'_2, \widetilde{\sigma}'_3, \bar{R}_1, \ldots, \bar{R}_6, M)$ and 0 otherwise.

$\mathsf{Open}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t, \Sigma, M, St)$**:**

1. If $\mathsf{Verify}(\mathsf{gpk}, t, RL_t, \Sigma, M) = 0$, then output $\perp$.
2. Otherwise, decrypt $(C_1, C_2, C_z, C_\sigma, C_{ID}, C_u, C'_z, C'_\sigma)$ by using $(x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)$ such that $\sigma_1 = C_\sigma \cdot C_1^{-x_\sigma} \cdot C_2^{-y_\sigma}$, $\pi = C_z \cdot C_1^{-x_z} \cdot C_2^{-y_z}$, $V_{\mathsf{ID}} = C_{ID} \cdot C_1^{-x_{ID}} \cdot C_2^{-y_{ID}}$, $V_u = C_u \cdot C_1^{-x_u} \cdot C_2^{-y_u}$, $\sigma'_1 = C'_\sigma \cdot C_1^{-x'_\sigma} \cdot C_2^{-y'_\sigma}$, and $\pi' = C'_z \cdot C_1^{-x'_z} \cdot C_2^{-y'_z}$.
3. Search V_{ID} from in the database of joining transcripts and get $(\widehat{G}_{2,\mathsf{ID}}, \widehat{G}_{5,\mathsf{ID}})$. If there is no such entry, then output $\perp$.
4. Let V_{ID} be contained in cert_i where $\mathsf{cert}_i = (i, \mathsf{Path}(i), V_{\mathsf{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \mathsf{Path}(i)})$. Search j such that $V_u = V_u^{(j)}$ and obtain $u_j \in \mathsf{Path}(i)$. If there is no such j, then output $\perp$.
5. Check whether $(\sigma_1, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \pi)$ and $(\sigma'_1, \widetilde{\sigma}'_2, \widetilde{\sigma}'_3, \pi')$ are valid LMPY signatures as follows.

$$\begin{aligned} e(\Omega, \widehat{g}_8)^{-1} &= e(\pi, \widehat{g}_z) e(\sigma_1, \widehat{g}_1) e(\widetilde{\sigma}_2, \widehat{G}_{2,\mathsf{ID}} \cdot \widehat{g}_3^{u_j} \cdot \widehat{g}_4) e(\widetilde{\sigma}_3, \widehat{G}_{5,\mathsf{ID}} \cdot \widehat{g}_6^{u_j} \cdot \widehat{g}_7) \\ e(\Omega', \widehat{g}'_8)^{-1} &= e(\pi', \widehat{g}'_z) e(\sigma'_1, \widehat{g}'_1) e(\widetilde{\sigma}'_2, {\widehat{g}'_2}{}^t \cdot {\widehat{g}'_3}{}^{u_j} \cdot \widehat{g}'_4) e(\widetilde{\sigma}'_3, {\widehat{g}'_5}{}^t \cdot {\widehat{g}'_6}{}^{u_j} \cdot \widehat{g}'_7)\end{aligned}$$

If the above equations hold, then output i and $\perp$ otherwise.

5 Security Analysis

Intuitively, security against misidentification attacks hold as follows. Due to revocation functionality the winning condition of the adversary is changed from $i^* \notin U^a$ to $i^* \notin U^a \setminus \mathcal{R}_{t^*}$ where i^* is the opening result of the group signature output by the adversary at time t^*, U^a is the set of signers who joined via $Q_{\mathsf{a\text{-}join}}$ queries, and $\mathcal{R}_{t^*}$ is the set of revoked signers at time t^*. Remark that i may be $\perp$. We divide the condition $i^* \notin U^a \setminus \mathcal{R}_{t^*}$ to (1) $i^* \notin U^a$ and (2) $i^* \in \mathcal{R}_{t^*}$. The first case is the same as that of the proof given by Libert et al. [54], i.e., the adversary produces a valid group signature whose opening result is in outside of the set of adversarially-controlled signers. This case is reduced to the unforgeability of the LMPY signature scheme on some (ID, u) where ID was not chosen in interactions between the adversary and the $Q_{\mathsf{a\text{-}join}}$ oracle. The second case is that the adversary can produce a valid group signature whose opening result is in the set of revoked signers. This case is also reduced to the unforgeability of the LMPY signature scheme on some (t^*, u) where u is a node and the signature is not contained in the revocation list RL_{t^*}. Since the LMPY signature is unforgeable under the SXDH assumption, Theorem 5.1 holds. For security against framing attacks, in the join protocol, a signer chooses ID and it is unknown to the group manager. Thus, from a forged signature output by the adversary of framing attacks, we can construct an algorithm that extracts such an unknown identity and uses it to solve the SDL problem. Moreover, due to the soundness of the QA-NIZK argument and the CCA security of the Cramer-Shoup encryption scheme, our scheme is anonymous.

Theorem 5.1. *The proposed group signature scheme is secure against misidentification attacks if the SXDH assumption holds in $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$ in the random oracle model.*

Proof: Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger of the LMPY signature scheme. We construct an algorithm $\mathcal{B}$ that breaks the unforgeability of the LMPY scheme by using $\mathcal{A}$ as follows.

Case (1): In this case, we assume that $i^* \notin U^a$. First, $\mathcal{B}$ chooses bilinear groups $\mathsf{cp} = (\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, p)$ where $p > 2^\lambda$, vectors $\mathbf{v}$ and $\mathbf{v}'$, and matrices $\mathbf{M}$ and $\mathbf{M}'$ with the discrete logarithms of the group elements of $\mathbf{M}$ and $\mathbf{M}'$ (i.e., $\mathbf{M}$ and $\mathbf{M}'$ are witness-samplable). $\mathcal{B}$ sends cp and $\mathbf{M}$ to $\mathcal{C}$ and obtains $\mathsf{crs} = (\{z_i\}_{i=1}^4, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^8)$. $\mathcal{B}$ runs $\mathsf{QA.KeyGen}(\mathsf{cp}, \mathbf{M}')$ and generates $\mathsf{sk}' = \omega'$ and $\mathsf{vk}' = (\mathsf{cp}, g', h', \widehat{g}', \mathbf{v}', \Omega' = h'^{\omega'}, \mathsf{crs}')$ where $\mathsf{crs}' = (\{z_i'\}_{i=1}^4, \widehat{g}_z', \{\widehat{g}_j'\}_{j=1}^8)$. $\mathcal{B}$ generates other components $(X_z, X_\sigma, X_{ID}, X_z', X_\sigma')$ with $\mathcal{S}_{\mathsf{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x_z', y_z', x_\sigma', y_\sigma')$, and sets $\mathsf{gpk} = (\mathsf{vk}, \mathsf{vk}', X_z, X_\sigma, X_{ID}, X_u, X_z', X_\sigma')$. For Q_{pub} and Q_{keyOA} queries, $\mathcal{B}$ sends gpk and $\mathcal{S}_{\mathsf{OA}}$ to $\mathcal{A}$ respectively.

$Q_{\mathsf{a\text{-}join}}$: $\mathcal{A}$ is allowed to run $\mathsf{J}_{\mathsf{user}}$. We show how to simulate J_{GM} as follows. First $\mathcal{A}$ sends $(V_{\mathsf{ID}}, Z_{\mathsf{ID}}, \widehat{G}_{2,\mathsf{ID}}, \widehat{G}_{5,\mathsf{ID}})$ to $\mathcal{B}$. If V_{ID} has been appeared in transcripts of St, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ checks the verification equations hold (see the step (2) of the joining procedure). If all tests pass, $\mathcal{B}$ samples a fresh index $i \in \mathbb{Z}_p$ and a fresh leaf node, and sends i to $\mathcal{A}$. Next, $\mathcal{B}$ uses the knowledge extractor of the proof of knowledge of $\mathsf{ID} = \log_{v_1}(V_{\mathsf{ID}})$ by rewinding $\mathcal{A}$, and obtains ID. For all $u_j \in \mathsf{Path}(i)$, $\mathcal{B}$ sends (ID, u_j) to $\mathcal{C}$ as a signing query , and finally $\mathcal{B}$ sends $\mathsf{cert}_i = (i, \mathsf{Path}(i), V_{\mathsf{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \mathsf{Path}(i)})$ to $\mathcal{A}$.

Q_{revoke}: Since $\mathcal{B}$ knows $\mathsf{sk}' = \omega'$, $\mathcal{B}$ just runs the Revoke algorithm as usual.

At some point t^*, $\mathcal{A}$ outputs a forged group signature $\Sigma^* = (C^*_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2^*, \widetilde{\sigma}_3^*, \widetilde{\sigma}_2'{}^*, \widetilde{\sigma}_3'{}^*, c^*, s^*_{\mathsf{ID}}, s^*_\theta, s^*_u)$ and its message M^*. By rewinding $\mathcal{A}$, due to the generalized Forking Lemma [16], $\mathcal{B}$ obtains two vaid group signatures $(C^*_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2^*, \widetilde{\sigma}_3^*, \widetilde{\sigma}_2'{}^*, \widetilde{\sigma}_3'{}^*, c^*, s^*_{\mathsf{ID}}, s^*_\theta, s^*_u)$ and $(C^*_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2^*, \widetilde{\sigma}_3^*, \widetilde{\sigma}_2'{}^*, \widetilde{\sigma}_3'{}^*, c^\dagger, s^\dagger_{\mathsf{ID}}, s^\dagger_\theta, s^\dagger_u)$ with $c^* \neq c^\dagger$, $s^*_{\mathsf{ID}} \neq s^\dagger_{\mathsf{ID}}$, and $s^*_u \neq s^\dagger_u$. $\mathcal{B}$ computes $\mathsf{ID}^* = (s^\dagger_{\mathsf{ID}} - s^*_{\mathsf{ID}})/(c^* - c^\dagger) \bmod p$ and $u^* = (s^\dagger_u - s^*_u)/(c^* - c^\dagger) \bmod p$. Moreover, $\mathcal{B}$ decrypts $C^*_{\mathsf{Cramer\text{-}Shoup}}$ and obtains (σ_1, π) where $(\sigma_1, \widetilde{\sigma}_2^*, \widetilde{\sigma}_3^*, \pi)$ is a valid LMPY signature on (ID^*, u^*), and due to the winning condition, $\mathcal{B}$ has not sent (ID^*, u^*) to $\mathcal{C}$ as a signing query. Thus, $\mathcal{B}$ outputs (ID^*, u^*) and $(\sigma_1, \widetilde{\sigma}_2^*, \widetilde{\sigma}_3^*, \pi)$ as a forgery and breaks the unforgeability of the LMPY scheme.

Case (2): In this case, we assume that $i^* \in \mathcal{R}_{t^*}$. As in the first case, $\mathcal{B}$ chooses cp, $\mathbf{v}$, $\mathbf{v}'$, $\mathbf{M}$, and $\mathbf{M}'$. $\mathcal{B}$ sends cp and $\mathbf{M}'$ to $\mathcal{C}$ and obtains $\mathsf{crs}' = (\{z_i'\}_{i=1}^4, \widehat{g}_z', \{\widehat{g}_j'\}_{j=1}^8)$. $\mathcal{B}$ runs $\mathsf{QA.KeyGen}(\mathsf{cp}, \mathbf{M})$ and generates $\mathsf{sk} = \omega$ and $\mathsf{vk} = (\mathsf{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega = h^\omega, \mathsf{crs})$ where $\mathsf{crs} = (\{z_i\}_{i=1}^4, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^8)$. $\mathcal{B}$ generates other components $(X_z, X_\sigma, X_{ID}, X_u, X_z', X_\sigma')$ with $\mathcal{S}_{\mathsf{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x_z', y_z', x_\sigma', y_\sigma')$, and sets $\mathsf{gpk} = (\mathsf{vk}, \mathsf{vk}', X_z, X_\sigma, X_{ID}, X_u, X_z', X_\sigma')$. For Q_{pub} and Q_{keyOA} queries, $\mathcal{B}$ sends gpk and $\mathcal{S}_{\mathsf{OA}}$ to $\mathcal{A}$ respectively.

$Q_{\mathsf{a\text{-}join}}$: Since $\mathcal{B}$ knows $\mathsf{sk} = \omega$, $\mathcal{B}$ just runs the $\mathsf{Join}^{\mathsf{GM}, \mathcal{U}_i}$ protocol as usual.

Q_{revoke}: First, $\mathcal{B}$ runs $Y \leftarrow \mathsf{CS}(\mathsf{BT}, \mathcal{R}_t)$. For all $u_j \in Y$, $\mathcal{B}$ sends (t, u_j) to $\mathcal{C}$ as a signing query. Finally, $\mathcal{B}$ outputs $RL_t = (t, Y, \{(\sigma'_{j,1}, \sigma'_{j,2}, \sigma'_{j,3}, \pi'_j, u_j)\}_{u_j \in Y})$.

As in the first case, at some point t^*, $\mathcal{A}$ outputs Σ^* and M^*. $\mathcal{B}$ rewinds $\mathcal{A}$ and obtains u^*. $\mathcal{B}$ decrypts $C^*_{\mathsf{Cramer\text{-}Shoup}}$ and obtains (σ_1', π') where $(\sigma_1', \widetilde{\sigma}_2'{}^*, \widetilde{\sigma}_3'{}^*, \pi')$ is a valid LMPY signature on (t^*, u^*). Due to the winning condition, $\mathcal{B}$ has not sent (t^*, u^*) to $\mathcal{C}$ as a signing query. Thus, $\mathcal{B}$ outputs (t^*, u^*) and $(\sigma_1', \widetilde{\sigma}_2'{}^*, \widetilde{\sigma}_3'{}^*, \pi')$ as a forgery and breaks the unforgeability of the LMPY scheme. □

Theorem 5.2. *The proposed group signature scheme is secure against framing attacks under the SDL assumption in the random oracle model.*

Proof: Let $\mathcal{A}$ be an adversary. We construct an algorithm $\mathcal{B}$ that solves the SDL problem by using $\mathcal{A}$ as follows. First, $\mathcal{B}$ takes an SDL instance $(g, \widehat{g}, g^a, \widehat{g}^a)$ as input. $\mathcal{B}$ chooses $g', v_2, v'_1, v'_2, W' \xleftarrow{\$} \mathbb{G}$, $\widehat{g}' \xleftarrow{\$} \widehat{\mathbb{G}}$, $\alpha_h, \alpha_v, \alpha_W, \alpha_z, a' \xleftarrow{\$} \mathbb{Z}_p$, and computes $h = g^{\alpha_h}$, $v_1 = g^{\alpha_v}$, $W = g^{\alpha_W}$, $\widehat{g}_z = \widehat{g}^{\alpha_z}$, and $h' = {g'}^{a'}$. In order to compute $\{z_i\}_{i=1}^4$ of crs and $\{z'_i\}_{i=1}^4$ of crs', $\mathcal{B}$ chooses $\mathsf{tk} = \{\chi_j\}_{j=1}^8$ and $\mathsf{tk}' = \{\chi'_j\}_{j=1}^8$ respectively. $\mathcal{B}$ setups other values as usual, i.e., $\mathcal{B}$ knows $\mathcal{S}_{\mathsf{GM}} = (\mathsf{sk}, \mathsf{sk}')$, $\mathcal{S}_{\mathsf{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)$, tk, and tk'. For Q_{pub}, Q_{keyGM}, and Q_{keyOA} queries, $\mathcal{B}$ sends gpk, $\mathcal{S}_{\mathsf{GM}}$, and $\mathcal{S}_{\mathsf{OA}}$ to $\mathcal{A}$ respectively.

$Q_{\mathsf{b\text{-}join}}$: $\mathcal{A}$ is allowed to run J_{GM}. We show how to simulate $\mathsf{J}_{\mathsf{user}}$ as follows. $\mathcal{B}$ chooses $\delta_i \xleftarrow{\$} \mathbb{Z}_p$, and implicitly sets $\mathsf{ID}_i = a \cdot \delta_i$ where a is the solution of the SDL problem. $\mathcal{B}$ computes

$$\begin{aligned}
V_{\mathsf{ID}} &= (g^a)^{\alpha_v \cdot \delta_i} = v_1^{\mathsf{ID}_i} \\
Z_{\mathsf{ID}} &= \big((g^a)^{\alpha_v \cdot \delta_i}\big)^{-\chi_1} \big((g^a)^{\delta_i}\big)^{-\chi_2} \big((g^a)^{\alpha_h \cdot \delta_i}\big)^{-\chi_5} \\
&= (v_1^{\mathsf{ID}_i})^{-\chi_1} (g^{\mathsf{ID}_i})^{-\chi_2} (h^{\mathsf{ID}_i})^{-\chi_5} = z_2^{\mathsf{ID}_i} \\
\widehat{G}_{2,\mathsf{ID}} &= \big((\widehat{g}^a)^{\alpha_z \cdot \delta_i}\big)^{\chi_2} = \big((\widehat{g}^{\alpha_z})^{\chi_2}\big)^{a \cdot \delta_i} = (\widehat{g}_z^{\chi_2})^{\mathsf{ID}_i} = \widehat{g}_2^{\mathsf{ID}_i} \\
\widehat{G}_{5,\mathsf{ID}} &= \big((\widehat{g}^a)^{\alpha_z \cdot \delta_i}\big)^{\chi_5} = \big((\widehat{g}^{\alpha_z})^{\chi_5}\big)^{a \cdot \delta_i} = (\widehat{g}_z^{\chi_5})^{\mathsf{ID}_i} = \widehat{g}_5^{\mathsf{ID}_i}
\end{aligned}$$

and sends $(V_{\mathsf{ID}}, Z_{\mathsf{ID}}, \widehat{G}_{2,\mathsf{ID}}, \widehat{G}_{5,\mathsf{ID}})$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ simulates the interactive proof of knowledge of $\mathsf{ID}_i = \log_{v_1}(V_{\mathsf{ID}})$ using the simulator. Finally, $\mathcal{B}$ obtains $\mathsf{cert}_i = (i, \mathsf{Path}(i), V_{\mathsf{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \mathsf{Path}(i)})$.

Q_{revoke}: Since $\mathcal{B}$ knows $\mathsf{sk}' = \omega'$, $\mathcal{B}$ just runs the Revoke algorithm as usual.

Q_{sig}: Since $\mathcal{B}$ knows cert_i for $i \in U^b$, $\mathcal{B}$ can compute $C_{\mathsf{Cramer\text{-}Shoup}}$ as usual. Note that for rerandomization process, $\mathcal{B}$ uses V_{ID} and Z_{ID}. $\mathcal{B}$ simulates $(c, s_{\mathsf{ID}}, s_\theta, s_u)$ as follows. $\mathcal{B}$ chooses $c, s_{\mathsf{ID}}, s_\theta, s_u \xleftarrow{\$} \mathbb{Z}_p$ and computes $R_1 = g^{s_\theta} \cdot C_1^{-c}$, $R_2 = h^{s_\theta} \cdot C_2^{-c}$, $R_3 = v_1^{s_{\mathsf{ID}}} \cdot X_{\mathsf{ID}}^{s_\theta} \cdot C_{ID}^{-c}$, $R_4 = {v_2}^{s_u} \cdot {X_u}^{s_\theta} \cdot C_u^{-c}$, and R_5 and R_6 such that

$$\begin{aligned}
R_5 &= \big(e(X_z, \widehat{g}_z) e(X_\sigma, \widehat{g}_1)\big)^{s_\theta} \big(e(\widetilde{\sigma}_2, \widehat{g}_2) e(\widetilde{\sigma}_3, \widehat{g}_5)\big)^{-s_{\mathsf{ID}}} \big(e(\widetilde{\sigma}_2, \widehat{g}_3) e(\widetilde{\sigma}_3, \widehat{g}_6)\big)^{-s_u} \\
&\quad \times \big(e(C_z, \widehat{g}_z) e(C_\sigma, \widehat{g}_1) e(\widetilde{\sigma}_2, \widehat{g}_4) e(\widetilde{\sigma}_3, \widehat{g}_7) e(\Omega, \widehat{g}_8)\big)^{-c} \\
R_6 &= \big(e(X'_z, \widehat{g}'_z) e(X'_\sigma, \widehat{g}'_1)\big)^{s_\theta} \big(e(\widetilde{\sigma}'_2, \widehat{g}'_3) e(\widetilde{\sigma}'_3, \widehat{g}'_6)\big)^{-s_u} \\
&\quad \times \big(e(C'_z, \widehat{g}'_z) e(C'_\sigma, \widehat{g}'_1) e(\widetilde{\sigma}'_2, {\widehat{g}'_2}{}^t \cdot \widehat{g}'_4) e(\widetilde{\sigma}'_3, {\widehat{g}'_5}{}^t \cdot \widehat{g}'_7) e(\Omega', \widehat{g}'_8)\big)^{-c}
\end{aligned}$$

$\mathcal{B}$ programs the random oracle H such that $c = H(\mathsf{gpk}, t, C_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\sigma}'_2, \widetilde{\sigma}'_3, R_1, \ldots, R_6, M)$, and returns $\Sigma = (C_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\sigma}'_2, \widetilde{\sigma}'_3, c, s_{\mathsf{ID}}, s_\theta, s_u)$ to $\mathcal{A}$.

At some point t^*, $\mathcal{A}$ outputs a forged group signature $\Sigma^* = (C^*_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}^*_2, \widetilde{\sigma}^*_3, {\widetilde{\sigma}'_2}{}^*, {\widetilde{\sigma}'_3}{}^*, c^*, s^*_{\mathsf{ID}}, s^*_\theta, s^*_u)$ and its message M^*. Let i^* be the opening result of Σ^*. Due to the winning condition, $i^* \in U^b$ and i^* did not sign M^*, i.e., $(i^*, t^*, M^*, \Sigma^*) \notin \mathsf{Sigs}$. By rewinding $\mathcal{A}$, due to the generalized Forking Lemma [16], $\mathcal{B}$ obtains two vaid group signatures $(C^*_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}^*_2, \widetilde{\sigma}^*_3, {\widetilde{\sigma}'_2}{}^*, {\widetilde{\sigma}'_3}{}^*, c^*, s^*_{\mathsf{ID}}, s^*_\theta, s^*_u)$ and $(C^*_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}^*_2, \widetilde{\sigma}^*_3, {\widetilde{\sigma}'_2}{}^*, {\widetilde{\sigma}'_3}{}^*, c^\dagger, s^\dagger_{\mathsf{ID}}, s^\dagger_\theta, s^\dagger_u)$ with $c^* \neq c^\dagger$, $s^*_{\mathsf{ID}} \neq s^\dagger_{\mathsf{ID}}$, and $s^*_u \neq s^\dagger_u$. $\mathcal{B}$ computes $\mathsf{ID}^* = (s^\dagger_{\mathsf{ID}} - s^*_{\mathsf{ID}})/(c^* - c^\dagger) \bmod p$. Since $i^* \in U^b$, $\mathcal{B}$ has chosen δ_{i^*} which satisfies $\mathsf{ID}^* = a \cdot \delta_{i^*}$. $\mathcal{B}$ computes $a = \mathsf{ID}^*/\delta_{i^*} \bmod p$, and solves the SDL problem. □

Theorem 5.3. *The proposed group signature scheme is anonymous in the random oracle model if the SXDH assumption holds in* $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$.

Proof: Let $\mathcal{A}$ be an adversary, and S_i be the event that $\mathcal{A}$ successfully guesses b in Game i.

Game 0: This is the anonymity game. $\mathcal{A}$ sends $(\mathsf{cert}_0^*, \mathsf{sec}_0^*)$ and $(\mathsf{cert}_1^*, \mathsf{sec}_1^*))$, and obtains Σ^* which is computed by $(\mathsf{cert}_b^*, \mathsf{sec}_b^*)$ for $b \overset{\$}{\leftarrow} \{0,1\}$.

Game 1: This game is the same as Game 0 except that if for $(\sigma_1^*, \sigma_2^*, \sigma_3^*, \pi^*)$ chosen by $\mathcal{A}$ as either cert_0^* or cert_1^*, $\log_g(\sigma_2^*) \neq \log_h(\sigma_3^*)$ holds, then the challenger aborts. This contradicts the soundness of the QA-NIZK argument, and as in [54], $|\Pr[S_1] - \Pr[S_0]|$ is bounded by the advantage of the DDH problem in $\widehat{\mathbb{G}}$, and is negligible.

Game 2: This game is the same as Game 1 except that we change the responce of a Q_{open} query. For $\Sigma = (C_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\sigma}_2', \widetilde{\sigma}_3', c, s_{\mathsf{ID}}, s_\theta, s_u)$ where $\log_{g'}(\widetilde{\sigma}_2') \neq \log_{h'}(\widetilde{\sigma}_3')$ even if the Open algorithm does not output $\perp$, the challenger outputs $\perp$. Again, this contradicts the soundness of the QA-NIZK argument, and $|\Pr[S_2] - \Pr[S_1]|$ is bounded by the advantage of the DDH problem in $\widehat{\mathbb{G}}$, and is negligible.

Game 3: This game is the same as Game 2 except that we change the way to compute the challenge group signature Σ^*. For $(s_{\mathsf{ID}}^*, s_\theta^*, s_u^*)$, choose $c, s_{\mathsf{ID}}, s_\theta, s_u \overset{\$}{\leftarrow} \mathbb{Z}_p$ and computes from R_1 to R_6 and program the random oracle H as in the simulation of Q_{sig} in the proof of security against framing attacks. Since the probability that $(c, s_{\mathsf{ID}}, s_\theta, s_u)$ has been chosen before is at most $1/p^4$, $|\Pr[S_3] - \Pr[S_2]|$ is negligible.

Game 4: This game is the same as Game 3 except that we use $\mathcal{S}_{\mathsf{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x_z', y_z', x_\sigma', y_\sigma')$ to compute $C^*_{\mathsf{Cramer\text{-}Shoup}} = (C_1^*, C_2^*, C_z^*, C_\sigma^*, C_{ID}^*, C_u^*, {C'}_z^*, {C'}_\sigma^*)$. That is,

$$C_z^* = \widetilde{\pi} \cdot (C_1^*)^{x_z} (C_2^*)^{y_z}, \ C_\sigma^* = \widetilde{\sigma}_1 \cdot (C_1^*)^{x_\sigma} (C_2^*)^{y_\sigma}$$
$$C_{ID}^* = v_1^{\mathsf{ID}_i} \cdot (C_1^*)^{x_{ID}} (C_2^*)^{y_{ID}}, \ C_u^* = v_2^u \cdot (C_1^*)^{x_u} (C_2^*)^{y_u}$$
$${C'}_z^* = \widetilde{\pi}' \cdot (C_1^*)^{x_z'} (C_2^*)^{y_z'}, \ {C'}_\sigma^* = \widetilde{\sigma}_1' \cdot (C_1^*)^{x_\sigma'} (C_2^*)^{y_\sigma'}$$

This modification is conceptual, and $\Pr[S_4] = \Pr[S_3]$ holds.

Game 5: This game is the same as Game 4 except that the distribution of the challenge group signature is changed such that for $\theta' \overset{\$}{\leftarrow} \mathbb{Z}_p$, replace $C_2^* = h^\theta$ to $C_2^* = h^{\theta+\theta'}$. Due to Lemma 1 in [54], $|\Pr[S_5] - \Pr[S_4]|$ is bounded by the advantage of the DDH problem in $\mathbb{G}$, and is negligible.

Game 6: This game is the same as Game 5 except that we change the rejection rule of the opening query. Instead of choosing $h \overset{\$}{\leftarrow} \mathbb{G}$ in the setup phase, choose $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p$, set $h = g^\alpha$, and returns $\perp$ if $C_2 \neq C_1^\alpha$ for $\Sigma = (C_{\mathsf{Cramer\text{-}Shoup}}, \widetilde{\sigma}_2, \widetilde{\sigma}_3, \widetilde{\sigma}_2', \widetilde{\sigma}_3', c, s_{\mathsf{ID}}, s_\theta, s_u)$. Due to Lemma 2 in [54], the probability that this event happens is bounded by $q_O \cdot q_H / p$ where q_O is the number of opening queries and q_H is the number of hash queries.

In Game 6, Σ^* perfectly hides $(\widetilde{\pi}, \widetilde{\sigma}_1, v_1^{\mathsf{ID}}, v_2^u, \widetilde{\pi}', \widetilde{\sigma}_1')$, and $\Pr[S_6] = 1/2$. □

6 Application to Identity Management

Isshiki et al. [46] proposed an identity management system employing group signatures. In this section, we show that our revocable group signature scheme is also applied to their identity management system.

Briefly, their system is explained as follows. There are five entities: an issuing manager, who issues signing keys to users, a user-revocation manager, who has a role of revocation, an accounting manager, who can trace signers, an outsourcee, who provides a service, and users. First, a user joins a group and obtains a signing key from the issuing manager. When a user requests the service, the user generates a group signature and sends it to the outsourcee. The outsourcee provides the service if the signature is valid, and stores the group signature as the usage log. Due to anonymity, the outsourcee does not have to identify the user and also does not have to manage any identity list. At a certain interval, the accounting manager opens the group signature and charges the users according to their usage. When a user wants to leave the group or behaves maliciously, the user-revocation manager revokes the user. Since three managers are defined, Isshiki et al. required that the underlying group signature scheme can separate the keys for each manager. Fortunately, our scheme matches this requirement. Concretely, for $\mathcal{S}_{\mathsf{GM}} = (\mathsf{sk}, \mathsf{sk}')$, and $\mathcal{S}_{\mathsf{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)$, we can set sk as the issuing key, sk' as the revocation key, and $\mathcal{S}_{\mathsf{OA}}$ as the opening key.

Isshiki et al. employed the Camenisch-Groth group signature scheme [24] with a slight modification to provide a membership certificate update mechanism.[5] The Camenisch-Groth group signature scheme employs composite-order groups and elliptic curves, and for which Isshiki et al. set 2048-bit and 169-bit orders, respectively. Then, they showed that the running time of the signing algorithm was approximately 135 msec, and that of the verification algorithm was approximately 112 msec, which are slower than those of our scheme (see Section 7). Moreover, each user is required to update their membership certificate when revocation occurs. Isshiki et al. showed that if 1000 user revocations occur, then a user needs 3 seconds to update their membership certificate. In our scheme, no such certificate update is required.

Providing Weak Opening Soundness: In the Bellare-Shi-Zhang (BSZ) model [17], the OA provides a proof for opening, i.e., ownership of group signatures. Sakai et al. [71] considered opening soundness, which requires that it is infeasible to produce a proof of ownership of a valid group signature for any user except the original signer. They gave two definitions, weak opening soundness and opening soundness. Weak opening soundness can rule out the possibility that a malicious user can claim ownership of a signature produced by an honest user by forging an opening proof. Note that the malicious user does not collude with other entities, especially the GM and OA. As mentioned by Derler and Slamanig [33] and Sakai et al., weak opening soundness is reasonable in many applications. Since the LPY model, which we employed in this paper, does not support the ownership proof, it would be better to consider (weak) opening soundness. In particular, in the system of Isshiki et al., a user may claim that "I did not use the service" when the accounting manager opens a group signature and charges the user. Then, the accounting manager needs to prove that the signer of the group signature is the user. To capture this situation, weak opening soundness is enough since the user does not collude with the accounting manager. Thus, we modify our scheme to provide weak opening soundness as follows. First, we modify the Open algorithm and introduce the Judge algorithm, as in the Sakai et al. model as follows. To distinguish, we denote the Open algorithm as $\mathsf{Open}^{\mathsf{PoO}}$ when the algorithm outputs a non-interactive proof of the ownership τ. Let upk_i be the public key of user $\mathcal{U}_i$. We assume that upk_i is computed by $\mathcal{U}_i$ in the join phase.

$\mathsf{Open}^{\mathsf{PoO}}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t, \Sigma, M, St)$**:** The modified opening algorithm takes as input gpk, $\mathcal{S}_{\mathsf{OA}}$, t, Σ, M, and $St := (St_{\mathsf{users}}, St_{\mathsf{trans}})$, and outputs $\perp$ if the opening is failure, and (i, τ) such that $\mathsf{ID}_i \in St_{\mathsf{users}} \cup \{\perp\}$, otherwise. Here, τ is a non-interactive proof of the ownership of Σ.

$\mathsf{Judge}(\mathsf{gpk}, i, \mathsf{upk}_i, \Sigma, M, \tau)$**:** The judgement algorithm takes as input gpk, i, upk_i, Σ, M, and τ, and outputs 1 if (Σ, M) is provided by $\mathcal{U}_i$, and 0 otherwise.

Next, we modify our scheme as follows. Let $\mathsf{upk}_i = v_1^{\mathsf{ID}_i}$. Since $v_1^{\mathsf{ID}_i}$ has been contained in cert_i in the original scheme, publishing upk_i does not affect its security.[6] Moreover, we introduce another random oracle $H' : \{0,1\}^* \to \mathbb{Z}_p$.

[5] This group signature scheme is listed as Mechanism 5 in ISO/IEC 20008-2 [3].

[6] More concretely, in the definition of security against misidentification attacks, an adversary can obtain cert_i via the $Q_{\mathsf{a\text{-}join}}$ oracle. In the definition of Non-frameability, an adversary can obtain cert_i via the $Q_{\mathsf{b\text{-}join}}$ oracle. In the definition of Anonymity, an adversary can generate all certificates by myself by using $\mathcal{S}_{\mathsf{GM}}$.

$\mathsf{Open}^{\mathsf{PoO}}(\mathsf{gpk}, \mathcal{S}_{\mathsf{OA}}, t, \Sigma, M, St)$**:** In addition to the original opening procedure, the algorithm computes τ as follows when the original algorithm outputs i. Briefly, τ is a NIZK proof of (x_{ID}, y_{ID}) satisfying the relation $X_{ID} = g^{x_{ID}} h^{y_{ID}}$ and $\mathsf{upk}_i = C_{ID} \cdot C_1^{-x_{ID}} \cdot C_2^{-y_{ID}}$. That is, the relation shows that the decryption result of C_{ID} by using (x_{ID}, y_{ID}) contained in $\mathcal{S}_{\mathsf{OA}}$ is upk_i.

1. Choose $r'_{x_{ID}}, r'_{y_{ID}} \xleftarrow{\$} \mathbb{Z}_p$.
2. Compute $R_{ID} = g^{r'_{x_{ID}}} h^{r'_{y_{ID}}}$ and $R_{C_{ID}} = C_1^{-r'_{x_{ID}}} C_2^{-r'_{y_{ID}}}$.
3. Compute $c' = H'(\mathsf{gpk}, t, C_{ID}, C_1, C_2, R_{ID}, R_{C_{ID}})$ and $s'_{x_{ID}} = r'_{x_{ID}} + c' x_{ID}$ and $s'_{y_{ID}} = r'_{y_{ID}} + c' y_{ID}$.
4. Output (i, τ) where $\tau = (c', s'_{x_{ID}}, s'_{y_{ID}})$.

$\mathsf{Judge}(\mathsf{gpk}, i, \mathsf{upk}_i, \Sigma, M, \tau)$**:** If $\mathsf{Verify}(\mathsf{gpk}, t, RL_t, \Sigma, M) = 0$, then output 0. Otherwise, compute $\bar{R}_{ID} = g^{s'_{x_{ID}}} h^{s'_{y_{ID}}} X_{ID}^{-c'}$ and $\bar{R}_{C_{ID}} = C_1^{-s'_{x_{ID}}} C_2^{-s'_{y_{ID}}} (\mathsf{upk}_i / C_{ID})^{-c'}$. Output 1 if $c' = H'(\mathsf{gpk}, t, C_{ID}, C_1, C_2, \bar{R}_{ID}, \bar{R}_{C_{ID}})$, and 0 otherwise.

In the definition of weak opening soundness, an adversary modeled as the malicious user is given gpk, and declares a message to be signed M and two users, i and i^*. A signature Σ on M is honestly computed by using a signing key of i (in our notation, cert_i and sec_i), and is given to the adversary with the signing key of i^* (in our notation, cert_{i^*} and sec_{i^*}). Then the adversary outputs τ^*. The adversarey wins if $i \neq i^*$ and $\mathsf{Judge}(\mathsf{gpk}, i^*, \mathsf{upk}_{i^*}, \Sigma, M, \tau^*) = 1$.

If one can provide τ that is accepted by the Judge algorithm, we can construct an extractor that extracts (x_{ID}, y_{ID}) via the Forking Lemma. That is, the knowledge of (x_{ID}, y_{ID}) is indispensable to produce τ. This shows that the scheme provides weak opening soundness under the DL assumption. As a remark, we need to modify the definition of the Q_{open} oracle where it outputs τ in addition to the opening result. Then, in the proof of Anonymity, we need to construct a simulator that produces τ without knowing $\mathcal{S}_{\mathsf{OA}}$. This can be simply done by using the programmability of the random oracle H', and does not affect Anonymity.

7 Implementation

In this section, we give our implementation results of the proposed scheme. We chose two security parameters for our implementation: a Barreto-Lynn-Scott (BLS) curve [11] of embedding degree 12 over a 455-bit prime field (`BLS-12-455`) as a “conservative”128-bit security parameter, and a Barreto-Naehrig (BN) curve [12] of embedding degree 12 over a 382-bit prime field (`BN-12-382`) as an “optimistic”128-bit security parameter, both of them are categorized as 128-bit security parameters in the recent IETF internet-draft [76]. According to the Barbulescu-Duquesne security estimation [10], `BN-12-382` may not ensure 128-bit security, however, we followed the draft categorization, and adopt both optimistic and conservative parameters in this paper. Note that since our proposed scheme is based on the SXDH problem, it is necessary to employ elliptic curves that have type 3 pairings, such as BN and BLS curves. Our implementation environment was as follows: CPU: Core i7-7700K(4.20 GHz) and gcc 6.3.0. We employed the RELIC library [5].

Table 2 summarizes benchmarks of the elliptic curve and pairing operations on `BLS-12-455` and `BN-12-382` in our environment. Here, Mul($\mathbb{G}$,Type), Mul($\widehat{\mathbb{G}}$,Type) and Exp($\mathbb{G}_T$,Type) are scalar multiplication in $\mathbb{G}$ and $\widehat{\mathbb{G}}$, and exponentiation in $\mathbb{G}_T$, respectively. If a base point is previously known and fixed, then Type is set as K, otherwise it is set as U. Note that we always use Type U for exponentiations in $\mathbb{G}_T$ since the RELIC library does not support Type K in $\mathbb{G}_T$. Table 4 shows sizes of group elements: $\mathbb{Z}_p$, $\mathbb{G}$, $\widehat{\mathbb{G}}$, and $\mathbb{G}_T$, on parameters `BLS-12-455` and `BN-12-382`.

Table 2: Benchmarks of Group Operations

Operation		Time (μsec) `BLS-12-455` / `BN-12-382`
Mul($\mathbb{G}$,U)		248.729 / 209.145
Mul($\mathbb{G}$,K)		131.631 / 109.539
Mul($\widehat{\mathbb{G}}$,U)		530.114 / 438.078
Mul($\widehat{\mathbb{G}}$,K)		326.377 / 264.513
Exp($\mathbb{G}_T$,U)		743.482 / 632.369
Pairing	Miller loop	669.451 / 577.744
	Final exp.	779.022 / 418.154
	Total	1448.473 / 995.898

Table 3: Efficiency Comparison

	Sign
Ohara et al. [66]	27 Mul($\mathbb{G}$,K) + 2 Mul($\mathbb{G}$,U) + 4 ML + 2 FE
Ours	22 Mul($\mathbb{G}$,K) + 2 Mul($\mathbb{G}$,U) + 4 Mul($\widehat{\mathbb{G}}$,K) + 2 Exp($\mathbb{G}_T$,U) + 4 ML + 2 FE
	Verify
Ohara et al. [66]	23 Mul($\mathbb{G}$,K) + 11 Mul($\mathbb{G}$,U) + 4 ML + 2 FE
Ours	6 Mul($\mathbb{G}$,K) + 8 Mul($\mathbb{G}$,U) + 10 Mul($\widehat{\mathbb{G}}$,K) + 4 Exp($\mathbb{G}_T$,U) + 8 ML + 2 FE

ML: Miller Loop, FE: Final Exponentiation.

7.1 Comparison to Ohara et al. Scheme [66]

Before showing our implementation results, we concretely compare our scheme and Ohara et al. scheme [66] in aspects of computational performance and signature size.

We summarize the efficiency of the signing and verification algorithms in Table 3. For efficiency improvement and fair comparison, we optimized the computations of signing and verification algorithms of our and Ohara et al. schemes. We modified the equations to reduce the number of exponentiations in $\mathbb{G}_T$ and the number of pairing operations (Miller loops and final exponentiations) as much as possible, and also allowed to precompute some elements which are corresponding to public and static values, such as gpk. With the timing from Table 3, we can estimate the Ohara et al. scheme outperforms our scheme. This is considered as a trade-off between efficiency and complexity assumptions.

For signature size, our scheme contains 16 group elements (12 elements in $\mathbb{G}$ and 4 elements in $\mathbb{Z}_p$) whereas in the Ohara et al. scheme, a signature contains 18 group elements (5 elements in $\mathbb{G}$ and 13 elements in $\mathbb{Z}_p$). When the `BLS-12-455` is employed, our signature size is 852 bytes whereas that of the Ohara et al. scheme is 797 bytes. This is because, the signature of our scheme contains many elements in $\mathbb{G}$ compared to that of Ohara et al. scheme, and the size of element in $\mathbb{G}$ is slightly but significantly larger than that of a value in $\mathbb{Z}_p$ on `BLS-12-455`. On the other hand, when `BN-12-382` is employed, the size of a value in $\mathbb{Z}_p$ is almost the same as that of an element in $\mathbb{G}$ due to the structure of BN curves (See Table 4). In this case, our signature size is 780 bytes whereas that of the Ohara et al. scheme is 869 bytes. Table 5 shows our evaluation of signature size.

7.2 Implementation Results

Table 6 shows the running times of our scheme. Here, we set $N = 8192$, and assumed that 10% of users are revoked ($R = 819$). In the Sign, Verify, and Open algorithms, since some values are

Table 4: Element Size (Bytes)

	$\mathbb{Z}_p$	$\mathbb{G}$	$\widehat{\mathbb{G}}$	$\mathbb{G}_T$
BLS-12-455	39	58	115	456
BN-12-382	48	49	97	384

Table 5: Signature Size

	$\mathbb{G}$	$\mathbb{Z}_p$	Signature Size (Bytes) BLS-12-455/BN-12-382
Ohara et al. [66]	5	13	797/869
Ours	12	4	852/780

fixed and it is unnecessary to recompute them during time t, these values are first precomputed then used at $\mathsf{Sign}/\mathsf{Verify}/\mathsf{Open}$ in our implementation. In Table 6, "Precompute" rows show the timings of the precomputation phase, and "Online" rows show the actual $\mathsf{Sign}/\mathsf{Verify}/\mathsf{Open}$ timings after the precomputation. Note that the Open, $\mathsf{Open}^{\mathsf{PoO}}$, and Judge algorithms call the Verify algorithm. However, additional running time should be incurred, and thus the running times of these algorithms do not contain those of the Verify algorithm. Moreover, the running time of the $\mathsf{Open}^{\mathsf{PoO}}$ algorithm does not contain the running time of the Open algorithm when a proof τ is additionally produced.

8 Conclusion

In this paper, we proposed a revocable group signature scheme with scalability that is secure under the simple assumptions in the random oracle model. We implemented our scheme, and showed that it is feasible in practice. In addition, we can consider the following as future works.

Bootle et al. [21] considered full dynamicity, where signers can join and leave a group at any time, and mentioned that the LPY model is not fully dynamic; our scheme is also not fully dynamic for the same reason. They mentioned that a possible countermeasure against the above attack is to regard unassigned leaves as revoked until they are assigned. However, due to the complexity of the CS method, this does not provide scalability since the size of the revocation list linearly depends on N since R can be greater than $N/2$ [65]. It would be interesting to consider full dynamicity with scalable revocation. They also showed that the accountable ring signature scheme, which implies group signatures, proposed in [22] satisfies full dynamicity. Later, Lai et al. [52] proposed an accountable ring signature scheme that is secure in the standard model (which can also be seen as a fully dynamic group signature scheme). They also proposed a generic construction of sanitizable signatures with unlinkability based on accountable ring signatures. Recently, Ling et al. proposed a fully dynamic group signature scheme from lattices [60].

Backes, Hanzlik, and Schneider [8] considered membership privacy by pointing out that in the accountable ring signature scheme [22], group information contains a description of the active members at each epoch. They mentioned that this can be an issue when group signatures are used in an access control system for resources, e.g., phishing or DoS for a particular resource. They also proposed an efficient fully dynamic group signature scheme in the standard model that is secure under simple assumptions, where a group signature contains 28 $\mathbb{G}$ elements, 15 $\widehat{\mathbb{G}}$ elements, and one $\mathbb{Z}_p$ element. As another type of membership privacy, Emura et al. [38] considered hiding the number of revoked signers since if the number is revealed, then one may guess the reason behind such circumstances, which may lead to harmful rumors.

Kiayias and Zhou [49] and Chow et al. [28] proposed hidden identity-based signatures where

Table 6: Implementation Results

Algorithm		Time (msec)
		BLS-12-455 / BN-12-382
Setup		29.394 / 23.468
Join	GM	52.258 / 41.208
	User	8.895 / 7.145
	Total	61.153 / 48.353
Sign	Precompute	13.933 / 10.785
	Online	14.262 / 11.156
Verify	Precompute	20.761 / 15.099
	Online	20.096 / 16.461
Revoke		21.782 / 17.405
Open (w/o Verify)		17.851 / 14.017
$\mathsf{Open}^{\mathsf{PoO}}$ (w/o Open)		1.343 / 1.110
Judge (w/o Verify)		1.853 /1.519

opening only just requires the secret key of the OA and does not require any other secret members list. Thus, the membership list can be hidden from the OA. In our scheme, the OA needs to know the database of joining transcripts, which may cause a list exposure risk since the list is not a secret key by definition and secure storage for such a large list is relatively expensive. Thus, it is meaningful to remove such a list for the opening procedure. Considering these improvements is left as an important open problem.

Acknowledgement

This work was partially supported by the JSPS KAKENHI Grant Numbers JP16K00198 and JP19H04107.

References

[1] Intel Enhanced Privacy ID (EPID) Security Technology. https://software.intel.com/en-us/articles/intel-enhanced-privacy-id-epid-security-technology.

[2] Intel Software Guard Extensions (Intel SGX). https://software.intel.com/en-us/sgx.

[3] *ISO/IEC 20008-2. Information technology - security techniques - anonymous digital signatures - part 2: Mechanisms using a group public key.* 2013.

[4] M. Akane, Y. Nogami, and Y. Morikawa. Fast ate pairing computation of embedding degree 12 using subfield-twisted elliptic curve. *IEICE Transactions*, 92-A(2):508–516, 2009.

[5] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[6] N. Attrapadung, K. Emura, G. Hanaoka, and Y. Sakai. A revocable group signature scheme from identity-based revocation techniques: Achieving constant-size revocation list. In *ACNS*, pages 419–437, 2014.

[7] N. Attrapadung, K. Emura, G. Hanaoka, and Y. Sakai. Revocable group signature with constant-size revocation list. *Comput. J.*, 58(10):2698–2715, 2015.

[8] M. Backes, L. Hanzlik, and J. Schneider. Membership privacy for fully dynamic group signatures. *IACR Cryptology ePrint Archive*, 2018:641, 2018.

[9] R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *IACR Cryptology ePrint Archive*, 2017:334, 2017.

[10] R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *J. Cryptology*, 2018.

[11] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *SCN*, pages 257–267, 2002.

[12] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331, 2005.

[13] N. Begum, T. Nakanishi, S. Sadiah, and M. E. Islam. Implementation of a revocable group signature scheme with compact revocation list using accumulator. In *CANDAR*, pages 610–615, 2016.

[14] M. Bellare, A. Boldyreva, K. Kurosawa, and J. Staddon. Multirecipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Trans. Information Theory*, 53(11):3927–3943, 2007.

[15] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, pages 614–629, 2003.

[16] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS*, pages 390–399, 2006.

[17] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA*, pages 136–153, 2005.

[18] P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi. Get shorty via group signatures without encryption. In *SCN*, pages 381–398, 2010.

[19] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

[20] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM CCS*, pages 168–177, 2004.

[21] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. In *ACNS*, pages 117–136, 2016.

[22] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, J. Groth, and C. Petit. Short accountable ring signatures based on DDH. In *ESORICS*, pages 243–265, 2015.

[23] E. Brickell and J. Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. In *IEEE SocialCom*, pages 768–775, 2010.

[24] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. In *SCN*, pages 120–133, 2004.

[25] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, pages 61–76, 2002.

[26] D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[27] J. H. Cheon. Discrete logarithm problems with auxiliary inputs. *J. Cryptology*, 23(3):457–476, 2010.

[28] S. S. M. Chow, H. Zhang, and T. Zhang. Real hidden identity-based signatures. In *Financial Cryptography and Data Security*, pages 21–38, 2017.

[29] R. Clarisse and O. Sanders. Short group signature in the standard model. *IACR Cryptology ePrint Archive*, 2018:1115, 2018.

[30] R. Cramer, I. Damgård, and P. D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *Public Key Cryptography*, pages 354–373, 2000.

[31] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[32] C. Delerablée and D. Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT*, pages 193–210, 2006.

[33] D. Derler and D. Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In *ACM AsiaCCS*, pages 551–565, 2018.

[34] K. Emura and T. Hayashi. A light-weight group signature scheme with time-token dependent linking. In *LightSec*, pages 37–57, 2015.

[35] K. Emura and T. Hayashi. A revocable group signature scheme with scalability from simple assumptions and its implementation. In *ISC*, pages 442–460, 2018.

[36] K. Emura and T. Hayashi. Road-to-vehicle communications with time-dependent anonymity: A lightweight construction and its experimental results. *IEEE Trans. Vehicular Technology*, 67(2):1582–1597, 2018.

[37] K. Emura, T. Hayashi, and A. Ishida. Group signatures with time-bound keys revisited: A new model and an efficient construction. In *ACM AsiaCCS*, pages 777–788, 2017.

[38] K. Emura, A. Miyaji, and K. Omote. An r-hiding revocable group signature scheme: Group signatures with the property of hiding the number of revoked users. *J. Applied Mathematics*, 2014:983040:1–983040:14, 2014.

[39] C. Fan, R. Hsu, and M. Manulis. Group signature with constant revocation costs for signers and verifiers. In *CANS*, pages 214–233, 2011.

[40] S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the non-malleability of the Fiat-Shamir transform. In *INDOCRYPT*, pages 60–79, 2012.

[41] J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. *IEICE Transactions*, 89-A(5):1328–1338, 2006.

[42] J. Groth. Fully anonymous group signatures without random oracles. In *ASIACRYPT*, pages 164–180, 2007.

[43] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

[44] A. Ishida, Y. Sakai, K. Emura, G. Hanaoka, and K. Tanaka. Fully anonymous group signature with verifier-local revocation. In *SCN*, pages 23–42, 2018.

[45] A. Ishida, Y. Sakai, K. Emura, G. Hanaoka, and K. Tanaka. Proper usage of the group signature scheme in ISO/IEC 20008-2. In *ACM AsiaCCS*, pages 515–528, 2019.

[46] T. Isshiki, K. Mori, K. Sako, I. Teranishi, and S. Yonezawa. Using group signatures for identity management and its implementation. In *Workshop on Digital Identity Management*, pages 73–78, 2006.

[47] A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. *IACR Cryptology ePrint Archive*, 2004:76, 2004.

[48] A. Kiayias and M. Yung. Secure scalable group signature with dynamic joins and separable authorities. *IJSN*, 1(1/2):24–45, 2006.

[49] A. Kiayias and H. Zhou. Hidden identity-based signatures. *IET Information Security*, 3(3):119–127, 2009.

[50] E. Kiltz and H. Wee. Quasi-adaptive NIZK for linear subspaces revisited. In *EUROCRYPT*, pages 101–128, 2015.

[51] V. Kumar, H. Li, J. J. Park, K. Bian, and Y. Yang. Group signatures with probabilistic revocation: A computationally-scalable approach for providing privacy-preserving authentication. In *ACM CCS*, pages 1334–1345, 2015.

[52] R. W. F. Lai, T. Zhang, S. S. M. Chow, and D. Schröder. Efficient sanitizable signatures without random oracles. In *ESORICS*, pages 363–380, 2016.

[53] A. Langlois, S. Ling, K. Nguyen, and H. Wang. Lattice-based group signature scheme with verifier-local revocation. In *Public Key Cryptography*, pages 345–361, 2014.

[54] B. Libert, F. Mouhartem, T. Peters, and M. Yung. Practical "signatures with efficient protocols" from simple assumptions. In *ACM AsiaCCS*, pages 511–522, 2016.

[55] B. Libert, T. Peters, and M. Yung. Group signatures with almost-for-free revocation. In *CRYPTO*, pages 571–589, 2012.

[56] B. Libert, T. Peters, and M. Yung. Scalable group signatures with revocation. In *EUROCRYPT*, pages 609–627, 2012.

[57] B. Libert, T. Peters, and M. Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In *CRYPTO*, pages 296–316, 2015.

[58] B. Libert and D. Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In *CANS*, pages 498–517, 2009.

[59] S. Ling, K. Nguyen, A. Roux-Langlois, and H. Wang. A lattice-based group signature scheme with verifier-local revocation. *Theor. Comput. Sci.*, 730:1–20, 2018.

[60] S. Ling, K. Nguyen, H. Wang, and Y. Xu. Lattice-based group signatures: Achieving full dynamicity with ease. In *ACNS*, pages 293–312, 2017.

[61] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *Public Key Cryptography*, pages 463–480, 2009.

[62] T. Nakanishi and N. Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In *ASIACRYPT*, pages 533–548, 2005.

[63] T. Nakanishi and N. Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. In *IWSEC*, pages 17–32, 2006.

[64] T. Nakanishi and N. Funabiki. Revocable group signatures with compact revocation list using accumulators. *IEICE Transactions*, 98-A(1):117–131, 2015.

[65] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, pages 41–62, 2001.

[66] K. Ohara, K. Emura, G. Hanaoka, A. Ishida, K. Ohta, and Y. Sakai. Shortening the Libert-Peters-Yung revocable group signature scheme by using the random oracle methodology. *IEICE Transactions*, 2019, to appear. Available at `https://eprint.iacr.org/2016/477`.

[67] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[68] D. Pointcheval and O. Sanders. Short randomizable signatures. In *CT-RSA*, pages 111–126, 2016.

[69] S. Rahaman, L. Cheng, D. D. Yao, H. Li, and J. J. Park. Provably secure anonymous-yet-accountable crowdsensing with scalable sublinear revocation. *PoPETs*, 2017(4):384–403, 2017.

[70] S. Sadiah and T. Nakanishi. Revocable group signatures with compact revocation list using vector commitments. *IEICE Transactions*, 100-A(8):1672–1682, 2017.

[71] Y. Sakai, J. C. N. Schuldt, K. Emura, G. Hanaoka, and K. Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In *Public Key Cryptography*, pages 715–732, 2012.

[72] D. Slamanig, R. Spreitzer, and T. Unterluggauer. Adding controllable linkability to pairing-based group signatures for free. In *ISC*, pages 388–400, 2014.

[73] D. Slamanig, R. Spreitzer, and T. Unterluggauer. Linking-based revocation for group signatures: A pragmatic approach for efficient revocation checks. In *Mycrypt*, pages 364–388, 2016.

[74] D. Unruh. Quantum proofs of knowledge. In *EUROCRYPT*, pages 135–152, 2012.

[75] L. Wei and J. Liu. Shorter verifier-local revocation group signature with backward unlinkability. In *Pairing-Based Cryptography*, pages 136–146, 2010.

[76] S. Yonezawa, T. Kobayashi, and T. Saito. Pairing-Friendly Curves. Internet-Draft draft-yonezawa-pairing-friendly-curves-02, Internet Engineering Task Force, July 2019. Work in Progress.

[77] S. Zhou and D. Lin. Shorter Verifier-Local Revocation Group Signatures from Bilinear Maps. In *CANS*, pages 126–143, 2006.

Appendix

In this appendix, we introduce a 4-round protocol for proving the knowledge of $\mathsf{ID} = \log_{v_1}(V_{\mathsf{ID}})$ converted by the Cramer-Damgård-MacKenzie transformation [30]. The Cramer-Damgård-MacKenzie transformation converts a sigma protocol into a 4-round perfect zero-knowledge proof of knowledge. Briefly, a sigma protocol for the relation $\mathsf{ID} = \log_{v_1}(V_{\mathsf{ID}})$ (say Sigma) induces a commitment scheme. From this commitment scheme, a sigma protocol for the knowledge of the committed value can be constructed (say Sigma'). In the actual protocol, first the verifier commits a random value and proves knowledge of the value by using Sigma'. Next, the prover runs the OR-proof $\mathsf{Sigma} \vee \mathsf{Sigma}'$. We give a zero-knowledge proof of knowledge protocol for $\mathsf{ID} = \log_{v_1}(V_{\mathsf{ID}})$ as follows.

Verifier: Choose $\bar{\mathsf{ID}}, \bar{c} \xleftarrow{\$} \mathbb{Z}_p$ and compute $a = v_1^{\bar{\mathsf{ID}}} \cdot V_{\mathsf{ID}}^{-\bar{c}}$. Choose $r_{\bar{\mathsf{ID}}}, r_{\bar{c}} \xleftarrow{\$} \mathbb{Z}_p$ and compute $R' = v_1^{r_{\bar{\mathsf{ID}}}} \cdot V_{\mathsf{ID}}^{-r_{\bar{c}}}$. Send (a, R') to the prover.

Prover: Choose $\hat{c}, r_{\mathsf{ID}} \xleftarrow{\$} \mathbb{Z}_p$ and compute $R = v_1^{r_{\mathsf{ID}}}$. Choose $s'_{\mathsf{ID}}, s'_{\bar{c}}, \hat{c}' \xleftarrow{\$} \mathbb{Z}_p$ and compute $R'' = v_1^{s'_{\mathsf{ID}}} \cdot V_{\mathsf{ID}}^{-s'_{\bar{c}}} \cdot a^{-\hat{c}'}$. Send $(\hat{c}, R, R'')$ to the verifier.

Verifier: Compute $s_{\bar{\mathsf{ID}}} = r_{\bar{\mathsf{ID}}} + \hat{c} \cdot \bar{\mathsf{ID}}$ and $s_{\bar{c}} = r_{\bar{c}} + \hat{c} \cdot \bar{c}$. Choose $c \xleftarrow{\$} \mathbb{Z}_p$. Send $(s_{\bar{\mathsf{ID}}}, s_{\bar{c}}, c)$ to the prover.

Prover: Check $R' = v_1^{s_{\bar{\mathsf{ID}}}} \cdot V_{\mathsf{ID}}^{-s_{\bar{c}}} \cdot a^{-\hat{c}}$. If the equation does not hold, then abort. Compute $c^* = c - \hat{c}'$ and $s_{\mathsf{ID}} = r_{\mathsf{ID}} + c \cdot \mathsf{ID}$. Send $(c^*, \hat{c}', s_{\mathsf{ID}}, s'_{\mathsf{ID}}, s'_{\bar{c}})$ to the verifier.

Verifier: Check $c = c^* + \hat{c}'$, $R = v_1^{s_{\mathsf{ID}}} \cdot V_{\mathsf{ID}}^{-c}$, and $R'' = v_1^{s'_{\mathsf{ID}}} \cdot V_{\mathsf{ID}}^{-s'_{\bar{c}}} a^{-\hat{c}'}$. If these equations hold, then accept. Otherwise, reject.