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

Keita Emura\textsuperscript{5} and Takuya Hayashi\textsuperscript{5}

\textsuperscript{5}National Institute of Information and Communications Technology (NICT), Japan.
{\text{\{k-emura, t-hayashi\}@nict.go.jp}}

January 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Although such anonymity is quite attractive considering privacy issues, it is not trivial to check whether a signer has been revoked or not.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 [25] 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 [24] 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 [33]. This is the full version. We additionally considered weak opening soundness [68] and showed that our scheme can be applied to an identity management system [43]. See Section 6 for details. We also additionally gave the implementation result of the Judge algorithm.
computations. Boneh and Shacham [19] proposed group signatures with verifier-local revocation (VLR-GS). In VLR-GS [19, 50, 55, 60, 73, 72], 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 [58] 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 [37] 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. [70] proposed linking-based revocation, and gave an instantiation based on the Delerablée-Pointcheval group signature scheme [30] by employing a generic compiler [69]. 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 [48, 66, 32, 34]. 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) [53] 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 [62]. They further improved the efficiency of the LPY schemes by proposing a revocable group signature scheme with constant-size certification [52]. As followers of the LPY works, revocable group signatures with compact revocation list size were proposed [6, 7, 61, 67].

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 [41]. 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 [40] and the (non-revocable) Libert-Peters-Yung scheme [54], 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 [18], Furukawa-Imai [39], Delerablée-Pointcheval [30], Bichsel et al. [17], Pointcheval-Sanders [65], Derler-Slamanig [31], and Libert-Mouhartem-Peters-Yung [51].

In terms of the running time of signing and verification, Begum et al. [13] gave an implementation of the Nakanishi-Funabiki scheme [61], 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] over a 254-bit prime field and the embedding degree is 12, and utilized a library based on the “Cross-twisted \( \chi \)-based Ate (Xt-Xate) pairing” [4]. On the other hand, Emura, Hayashi, and Ishida [35]

\(^1\)Recently, Ishida et al. [42] showed that fully anonymous VLR group signatures can be realized. However, they employed general NIZK proofs and no concrete instantiation has not been proposed so far.
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, 22], and the EPID scheme builds on top of the Boneh-Boyen-Shacham group signature scheme [18] and the Furukawa-Imai group signature scheme [39]. These group signature schemes are secure in the random oracle model. Thus, improving efficiency 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 [63] 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 [53]. 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 [26], 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. We employed the methodology proposed by Ohara et al. [63], 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) [51] 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 [47]. Libert et al. proposed a group signature based on the LMPY signature scheme. Since the scheme does not provide

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Public key Size</th>
<th>Signature Size(^1)</th>
<th>Certificate Size</th>
<th>Revocation list Size</th>
<th>Signing Cost</th>
<th>Verification Cost</th>
<th>Std/ ROM(^3)</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPY1(CS) [53]</td>
<td>(O(1))</td>
<td>(O(1)) (96)</td>
<td>(O(\log N))</td>
<td>(O(R - \log(N/R)))</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>Std</td>
<td>(q)-Type</td>
</tr>
<tr>
<td>LPY2(SD) [53]</td>
<td>(O(\log N))</td>
<td>(O(1)) (96)</td>
<td>(O(\log^2 N))</td>
<td>(O(R))</td>
<td>(O(\log N)^1)</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>Std</td>
</tr>
<tr>
<td>LPY3 [52]</td>
<td>(O(\log N))</td>
<td>(O(1)) (144)</td>
<td>(O(1))</td>
<td>(O(R))</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>Std</td>
<td>(q)-Type</td>
</tr>
<tr>
<td>AEHS [6, 7]</td>
<td>(O(1))</td>
<td>(O(1)) (98)</td>
<td>(O(R_{\text{max}}))</td>
<td>(O(1))</td>
<td>(O(R)^1)</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>Std</td>
</tr>
<tr>
<td>NF [61]</td>
<td>(O(T \log N))</td>
<td>(O(1)) (143)</td>
<td>(O(T))</td>
<td>(O(R/T))</td>
<td>(O(T)^1)</td>
<td>(O(1))</td>
<td>Std</td>
<td>(q)-Type</td>
</tr>
<tr>
<td>SN [67]</td>
<td>(O(T + \log N))</td>
<td>(O(1)) (299)</td>
<td>(O(1))</td>
<td>(O(R/T))</td>
<td>(O(T)^1)</td>
<td>(O(1))</td>
<td>Std</td>
<td>(q)-Type</td>
</tr>
<tr>
<td>Ohara et al. [63]</td>
<td>(O(1))</td>
<td>(O(1)) (18)</td>
<td>(O(\log N))</td>
<td>(O(R - \log(N/R)))</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>ROM</td>
<td>(q)-Type</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>(O(1))</td>
<td>(O(1)) (16)</td>
<td>(O(\log N))</td>
<td>(O(R - \log(N/R)))</td>
<td>(O(1))</td>
<td>(O(1))</td>
<td>ROM</td>
<td>Simple</td>
</tr>
</tbody>
</table>

\(N\): The maximum number of group members.
\(R\): The number of revoked signers.
\(R_{\text{max}}\): The maximum number of revoked signers.
\(^1\) We denote the number of group elements contained in a group signature on \((\cdot)\). This number contains both the number of \(G\) elements and \(Z_p\) elements.
\(^1\) This complexity is only required at the first signature generation of each revocation epoch.
\(^5\) Standard Model / Random Oracle Model.

---

\(O\): The parameter of the accumulated/vector commitment value in [61, 67].
\(T\): The number of revoked signers.
\(R\): The maximum number of group members.
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 [33], we additionally considered weak opening soundness [68] and showed that our scheme can be applied to an identity management system [43]. 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 \( G, \hat{G}, \) and \( G_T \) be groups with prime order \( p > 2^\lambda \), and \( e : G \times \hat{G} \to G_T \) be a bilinear map. For \( g \in G \) and \( \hat{h} \in \hat{G} \), \( e(g, \hat{h}) \neq 1_{G_T} \) holds unless \( g \neq 1_G \) and \( \hat{h} \neq 1_{\hat{G}}. \)

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

**Definition 2.2** (SXDH Assumption). Let \( a \xleftarrow{\$} \mathbb{Z}_p^* \). We say that the SXDH assumption holds in \( (G, \hat{G}, G_T) \) if for any PPT adversary \( A \), the advantage
\[
\text{Adv}_{SXDH}(\lambda) := \Pr[A(g, \hat{g}, g^a, \hat{g}^a) \to a] \text{ is negligible.}
\]

**Definition 2.3** (SDL Assumption [51]). Let \( a \xleftarrow{\$} \mathbb{Z}_p^* \). We say that the SDL assumption holds in \( (G, \hat{G}, G_T) \) if for any PPT adversary \( A \), the advantage
\[
\text{Adv}_{SDL}(\lambda) := \Pr[A(g, \hat{g}, g^a, \hat{g}^a) \to a] \text{ is negligible.}
\]

2.2 QA-NIZK Arguments for Linear Subspaces

In this subsection, we introduce the Kiltz-Wee QA-NIZK arguments for linear subspaces [47] that prove membership in the row space of a matrix \( M \). As in [51], we assume that all algorithms take as input the description of common public parameters \( cp = (G, \hat{G}, G_T, p) \). In QA-NIZK proofs, the common reference string (CRS) may depend on the language to be proved. In the Kiltz-Wee case, it depends on \( M \). As in [51], for soundness, \( M \) is required to be witness-samplable where the reduction has to know the discrete logarithms of the group elements of \( M \).

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

**QA.KeyGen(cp, M):** This CRS and trapdoor generation algorithm takes as input \( cp \) and a matrix \( M \) where \( M = (M_{i,j})_{i \in [1,t], j \in [1,n]} \in G_t^{t \times n} \). Choose \( \hat{g}_Z \xleftarrow{\$} \hat{G} \) and a trapdoor \( tk = (\chi_1, \ldots, \chi_n) \xleftarrow{\$} Z_p^n. \) Compute \( \hat{g}_j = \hat{g}_Z^\chi_j \) 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 \( \text{crs} = (\{z_i\}_{i=1}^t, \hat{g}_Z, \{\hat{g}_j\}_{j=1}^n) \in G_t \times \hat{G}_{n+1} \) and the trapdoor \( tk \in Z_p^n. \)
As we can see, signature pairings. 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) [51] 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 $\ell$. Moreover, the verification algorithm requires just 5 pairings.

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

$$M = \begin{pmatrix} g & 1_{\ell+1} & 1_{\ell+1} & 1_{\ell+1} & h & 1_{\ell+1} \\ v^T & g_{\ell+1} & h_{\ell+1} & h_{\ell+1} & 1_{\ell+1}^T \\ \end{pmatrix}$$

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

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

\textbf{Verify(crs, v, \pi):} The verify algorithm takes as input $\text{crs} = (\{z_i\}_{i=1}^{\ell+2}, \tilde{g}_z, \{\tilde{g}_j\}_{j=1}^{2\ell+4}, v = (v_1, \ldots, v_n)$, and $\pi$, and output 1 if $\pi$ holds. For the sake of clarity, $\pi$ is described as

$$\pi = \prod_{i=1}^t \prod_{j=1}^n M_{i,j}^{-\chi_j} \omega_i$$

\textbf{Prove(crs, v, \{\omega_i\}_{i=1}^{\ell+2}):} The proof generation algorithm takes as input $\text{crs} = (\{z_i\}_{i=1}^{\ell+2}, \tilde{g}_z, \{\tilde{g}_j\}_{j=1}^{2\ell+4})$, a vector $v$, and witnesses $\{\omega_i\}_{i=1}^{\ell+2}$ where $v = \tilde{M}_1 \tilde{M}_2 \ldots \tilde{M}^{\ell+2}$ holds. Output a proof $\pi = \prod_{i=1}^t \prod_{j=1}^n z_i^\omega_i$ which proves that $v$ is a linear combination of the rows of $M$. 

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

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

\[
e(\Omega, \hat{g}_2^{\ell+4})^{-1} = e(\pi, \hat{g}_z) e(\sigma_1, \hat{g}_1) e(\sigma_2, \hat{g}_2^{m_1} \cdots \hat{g}_{\ell+1}^{m_\ell} \cdot \hat{g}_{\ell+2}) e(\sigma_3, \hat{g}_{\ell+3}^{m_1} \cdots \hat{g}_{2\ell+2}^{m_\ell} \cdot \hat{g}_{2\ell+3})
\]

The following theorem was given in [51].

**Theorem 2.1** ([51]). The LMPY signature scheme is existentially unforgeable under chosen-message attacks (EUF-CMA) if the SXDH assumption holds in \((G, \mathbb{G}, \mathbb{G}_T)\).

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 \( m = (m_1, \ldots, m_\ell) \), and a randomness \( s \in \mathbb{Z}_p \),

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

is also a valid signature on \( m \).

### 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 \( \text{com} \) to the verifier. Next, the verifier, that also takes as input the statement, sends a challenge \( \text{chall} \) to the prover. Finally, the prover sends a response \( \text{resp} \) to the verifier. We require special soundness, where given two accepted transcripts \((\text{com}_1, \text{chall}_1, \text{resp}_1)\) and \((\text{com}_2, \text{chall}_2, \text{resp}_2)\) with the condition \( \text{com}_1 = \text{com}_2 \) and \( \text{chall}_1 \neq \text{chall}_2 \), there is an extractor \( \text{Extract} \) that takes as input a statement \( s \) and two transcripts above, and outputs a witness \( \omega \) that satisfies \( L(s, \omega) = 1 \). Moreover, we also require special honest verifier zero knowledge (SHVZK), where there is a simulator \( \text{Sim} \) that takes as input a statement \( s \) and a challenge \( \text{chall} \), and outputs a transcript \( \text{com}, \text{chall}, \text{resp} \) that is indistinguishable from transcripts produced by the prover and the verifier as above. See [51] 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, \( \text{com} \) and \( \text{chall} \), no adversary can find responses \( \text{resp} \) and \( \text{resp}' \) which are both accepted but \( \text{resp} \neq \text{resp}' \). If the success probability is zero, then it is called unique responses, or it is also known as strict soundness [71]. See [38] for the formal definition. Faust et al. [38] 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 [62]. 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 \( \text{num} \) disjoint sets such as \( \mathcal{N} \setminus \mathcal{R} = S_1 \cup \cdots \cup S_{\text{num}} \), and \( \text{num} = O(R \cdot \log(N/R)) \) 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 \( R_t \) where \( i \in R_t \) when a signer with index \( i \) is revoked at time \( t \), and outputs a set of nodes. The description of CS is given below.

\[
\text{CS}(\text{BT}, R_t) :
\begin{align*}
X, Y &\leftarrow \emptyset; \\
\forall i \in R_t &\quad \text{Add Path}(i) \text{ to } X; \\
\forall x \in X &\quad \text{If } x_{\text{left}} \notin X \text{ then add } x_{\text{left}} \text{ to } Y; \\
&\quad \text{If } x_{\text{right}} \notin X \text{ then add } x_{\text{right}} \text{ to } Y; \\
&\quad \text{If } |RL_t| = 0 \text{ then add root to } Y; \\
&\quad \text{Return } Y;
\end{align*}
\]

In our group signature scheme, a signer, who has the identity ID, and whose path is \( \{u_0, u_1, \ldots, u_t\} \), has LMPY signatures on messages \( (\text{ID}, u_j) \) as certificates for all \( u_j \in \{u_0, u_1, \ldots, u_t\} \). The revocation list at time \( t \) also contains LMPY signatures on messages \( (t, u'_j) \) for all \( u'_j \in \{u'_0, u'_1, \ldots, u'_\text{num}\} \) which is determined by the CS method. If the signer is not revoked at \( t \), then there exists a node \( u \) such that \( u \in \{u_0, u_1, \ldots, u_t\} \cap \{u'_0, u'_1, \ldots, u'_\text{num}\} \). The signer proves that the knowledge of two LMPY signatures on \( (\text{ID}, u) \) and \( (t, u') \), and \( u = u' \). Here, the current time \( t \) is not required to be hidden, and is not a witness.

3 Revocable Group Signatures

In this section, we give the syntax and correctness definitions of revocable group signature. We use the LPY definitions [53, 52] which are modified from the Kiayias-Yung (KY) model [45, 44] to match the revocation functionality.

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

Definition 3.1 (Revocable Group Signature).

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 \( \text{gpk} \), the group manager (GM) private key for revocation \( \text{SGM} \), and the opening authority (OA) private key for opening \( \text{SOA} \). Moreover, the algorithm initializes a public state \( \text{St} \) comprising a set data structure \( \text{St}_{\text{users}} = \emptyset \) and a string data structure \( \text{St}_{\text{trans}} = \epsilon \).

\( \text{Join}^{\text{GM}/\text{U}} \): The interactive protocol for joining between GM and a signer \( \mathcal{U}_i \) (whose identity is \( \text{ID}_i \)) involves two interactive Turing machines \( \text{J}_{\text{user}} \) and \( \text{J}_{\text{GM}} \) which execution is denoted as \([\text{J}_{\text{user}}(\text{gpk}), \text{J}_{\text{GM}}(\text{St}, \text{gpk}, \text{SGM})]\). \( \mathcal{U}_i \) obtains a membership secret \( \text{sec}_i \) and a membership certificate \( \text{cert}_i \). We assume that \( \text{ID}_i \) is contained in \( \text{sec}_i \). If the protocol is successfully done, GM updates \( \text{St}_{\text{users}} \leftarrow \text{St}_{\text{users}} \cup \{\text{ID}_i\} \) and \( \text{St}_{\text{trans}} \leftarrow \text{St}_{\text{trans}} \cup \{t, \text{transcript}_t\} \).

\( \text{Revoke}(\text{gpk}, \text{SGM}, t, R_t \subset \text{St}_{\text{users}}) \): The revocation algorithm takes as input \( \text{gpk} \), \( \text{SGM} \), a revocation epoch \( t \), and a set of revoked signers \( R_t \subset \text{St}_{\text{users}} \), and outputs an updated revocation list \( RL_t \) which contains \( R_t \).

\( \text{Sign}(\text{gpk}, t, RL_t, \text{cert}, \text{sec}, M) \): The signing algorithm takes as input \( \text{gpk} \), a time \( t \), \( RL_t \), \( \text{cert} \), \( \text{sec} \), and a message \( M \) to be signed, and outputs \( \bot \) if \( \text{ID} \in R_t \), and a group signature \( \Sigma \), otherwise.
that, we introduce variables and oracles as follows:

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

1. In a valid state $St = (St_{users}, St_{trans})$, the condition $|St_{users}| = |St_{trans}|$ holds, and no two entries of $St_{trans}$ can contain certificates with the same tag.

2. If $[J_{user}(gpk), J_{GM}(St, gpk, S_{GM})]$ is honestly run by both parties and $(i, cert_i, sec_i)$ is obtained by $J_{user}$, then $cert_i = gpk \cdot sec_i$ holds.

3. For each $t$ and any $(i, cert_i, sec_i)$ satisfying condition 2,

   \[ \text{Verify}(gpk, t, RL_t, Sign(gpk, t, RL_t, cert, M), M) = 1 \]

   holds if $i \not\in R_t$.

4. For any $(i, cert_i, sec_i)$ resulting from the interaction $[J_{user}(\cdot, \cdot), J_{GM}(\cdot, St, \cdot, \cdot)]$ for some valid state $St$, any $t$ s.t. $i \not\in R_t$, $Open(gpk, S_{OA}, t, \Sigma, M, St) = i$ holds where $\Sigma \leftarrow Sign(gpk, t, RL_t, cert, sec, M)$.

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

- **$state_I$:** This is a data structure which is initialized as $state_I = (St, gpk, S_{GM}, S_{OA}) \leftarrow Setup(1^\lambda, N)$. 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).

- **$n = |St_{users}|$:** This is the current cardinality of the group.

- **Sigs:** This is a set of signatures $\text{Sigs}$ created by the signing oracle. Each entry is represented as $(i, t, M, \Sigma)$, where $\Sigma$ is a group signature on $M$ signed by $U_i$ at $t$.

- **$U^a$:** This is the set of corrupted signers who were introduced by the adversary $A$ 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 $A$ who acts a dishonest GM. $A$ can obtain the transcript of the join protocol, but $A$ cannot obtain $sec$.

- **$Q_{pub}, Q_{keyGM},$ and $Q_{keyOA}$:** When these oracles are invoked, the interface looks up $state_I$, and returns $gpk, S_{GM},$ or $S_{OA}$, respectively.

- **$Q_{a-join}$:** This is the join oracle for a corrupted signer. On behalf of GM, the interface runs $J_{GM}$ in interaction with $J_{user}$ 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. $St_{users} \leftarrow St_{users} \cup \{ID_n\}$ and $St_{trans} \leftarrow St_{trans} \cup \{n, transcript_n\}$. 

We next 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 [44, 45, 52, 53] we use the notation $cert_i = gpk \cdot sec_i$ to express that there exist coin tosses $\omega$ for $J_{GM}$ and $J_{user}$ such that, for some valid state $St'$, the execution of $[J_{user}(gpk), J_{GM}(St, gpk, S_{GM})](\omega)$ provides $J_{user}$ with $(i, cert_i, sec_i)$.
Definition 3.3 (Misidentification). Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\text{state}_\mathcal{I} = (\text{St}, \text{gpk}, \text{SGM}, \text{SOA}) \leftarrow \text{Setup}(\lambda, R)$. $\mathcal{A}$ is allowed to access $Q_{\text{pub}}$, $Q_{\text{a-join}}$, $Q_{\text{revoke}}$, $Q_{\text{read}}$, and $Q_{\text{keyOA}}$. Finally, $\mathcal{A}$ outputs $(M^*, \Sigma^*)$. We say that $\mathcal{A}$ wins if (1) Verify$(\text{gpk}, t^*, \text{RL}_t, \Sigma^*, M^*) = 1$, where $t^*$ is the challenge revocation epoch, and (2) for $i^* \leftarrow \text{Open}(\text{gpk}, \text{SOA}, t^*, \Sigma^*, M^*, \text{St}^t)$, $i^* \notin U^a \setminus \mathcal{R}_t$. Let $\text{Adv}_{\mathcal{A}}^{\text{mis-id}}(\lambda) := \text{Pr}[\mathcal{A} \text{ wins}].$ We say that $\mathcal{R}$-GS is secure against misidentification attack if for all PPT $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{mis-id}}(\lambda)$ is negligible.

Definition 3.4 (Non-frameability). Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be the challenger. $\mathcal{C}$ runs $\text{state}_\mathcal{I} = (\text{St}, \text{gpk}, \text{SGM}, \text{SOA}) \leftarrow \text{Setup}(\lambda, R)$. $\mathcal{A}$ is allowed to access $Q_{\text{pub}}$, $Q_{\text{KeyGM}}$, $Q_{\text{keyOA}}$, $Q_{\text{b-join}}$, $Q_{\text{revoke}}$, $Q_{\text{sig}}$, $Q_{\text{read}}$, and $Q_{\text{write}}$. Finally, $\mathcal{A}$ outputs $(M^*, \Sigma^*, t^*, \text{RL}_t)$. We say that $\mathcal{A}$ wins if (1) Verify$(\text{gpk}, t^*, \text{RL}_t, \Sigma^*, M^*) = 1$, and (2) for $i^* \leftarrow \text{Open}(\text{gpk}, \text{SOA}, t^*, \Sigma^*, M^*, \text{St}^t)$, $i^* \notin U^b$ and $(i^*, t^*, M^*, \Sigma^*) \notin \text{Sigs}$. Let $\text{Adv}_{\mathcal{A}}^{\text{nf}}(\lambda) := \text{Pr}[\mathcal{A} \text{ wins}].$ We say that $\mathcal{R}$-GS is secure against misidentification attack if for all PPT $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{nf}}(\lambda)$ is negligible.
Definition 3.5 (Anonymity). Let $A$ be an adversary and $C$ be the challenger. $C$ runs $\text{state}_T = (\text{St}, \text{gpk}, \text{S}_{\text{GM}}, \text{S}_{\text{OA}}) \leftarrow \text{Setup}(1^\lambda, R)$. $A$ is allowed to access $Q_{\text{pub}}$, $Q_{\text{KeyGM}}$, $Q_{\text{revoke}}$, $Q_{\text{open}}$, $Q_{\text{read}}$, and $Q_{\text{write}}$. $A$ outputs $(\text{aux}, M^t, RL^t, (\text{cert}_0^t, \text{sec}_0^t), (\text{cert}_1^t, \text{sec}_1^t))$. For $d \in \{0,1\}$, if $(\text{cert}_d^t = \text{gpk} \text{sec}_d^t)$, $\text{IsRevoked}(\text{sec}_d^t, \text{cert}_d^t, RL_L^t) = 0$, and $\text{cert}_0^t \neq \text{cert}_1^t$, then $C$ chooses $b \leftarrow \{0,1\}$, computes $\Sigma^* \leftarrow \text{Sign}(t^*, RL_L^t, \text{cert}_b^t, \text{sec}_b^t, M^*)$, and sends $\Sigma^*$ to $A$. Then $A$ is allowed to access $Q_{\text{pub}}$, $Q_{\text{KeyGM}}$, $Q_{\text{open}}$, $Q_{\text{read}}$, and $Q_{\text{write}}$, with one exception that $A$ is not allowed to send $(M^*, \Sigma^*, t^*)$ to $Q_{\text{open}}$. Finally, $A$ outputs $b' \in \{0,1\}$. Let $\text{Adv}_{\text{A anon}}^A(\lambda) := |\Pr[b = b'] - \frac{1}{2}|$. We say that $R$-GS is anonymous if all PPT $A$, $\text{Adv}_{\text{A anon}}^A(\lambda)$ is negligible.

4 Proposed Revocable Group Signature Scheme

In this section, we give our proposed revocable group signature scheme. In our group signature scheme, two LMPY signature schemes with $\ell = 2$ are setup. Let $\text{sk} = \omega$ and $\text{vk} = (\text{cp}, g, h, \hat{g}, \text{v}, \Omega, \text{crs})$, where $\text{crs} = (\{z_i\}_{i=1}^3, \tilde{g}_z, \{\tilde{g}_j\}_{j=1}^3)$, are for the first scheme, and $\text{sk}' = \omega'$ and $\text{vk}' = (\text{cp}, g', h', \hat{g}', \text{v}', \Omega', \text{crs}')$, where $\text{crs}' = (\{z_i\}_{i=1}^3, \tilde{g}_z, \{\tilde{g}_j\}_{j=1}^3)$, are for the second scheme. The first scheme signs $m = (\text{ID}, u)$ where $\text{ID}$ 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 $m' = (t, u)$ 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.

In the signing algorithm, first, the signer re-randomizes $(\sigma_1, \sigma_2, \sigma_3, \pi)$ and $(\sigma_1', \sigma_2', \sigma_3', \pi')$, and let $\tilde{\sigma} = (\bar{\sigma}_1, \bar{\sigma}_2, \bar{\sigma}_3, \bar{\pi})$ and $\tilde{\sigma}' = (\bar{\sigma}_1', \bar{\sigma}_2', \bar{\sigma}_3', \bar{\pi}')$ be the signatures after re-randomization. Then, $(\bar{\sigma}_2, \bar{\sigma}_3)$ and $(\bar{\sigma}_2', \bar{\sigma}_3')$ are independent from the signed messages and other signatures. Thus, these can be directly included in a group signature. Informally, for the NP language $L_{\text{LMPY}}$ induced by the relation $R_{L_{\text{LMPY}}}(\text{vk}, \text{vk}', (\tilde{\sigma}, \text{m}), (\tilde{\sigma}', \text{m}')) = 1$ iff $\text{Sig Verify}(\text{vk}, \tilde{\sigma}, \text{m}) = 1$, $\text{Sig Verify}(\text{vk}', \tilde{\sigma}', \text{m}') = 1$, and $m_2 = m_2' (= u)$, the verifier accepts if $L_{\text{LMPY}}(s, (\bar{\sigma}, \text{m}), (\bar{\sigma}', \text{m}')) = 1$ where the statement $s$ here is the verification equation of the LMPY scheme, and $((\bar{\sigma}_1, \bar{\pi}), (\text{ID}, u))$ and $((\bar{\sigma}_1', \bar{\pi}'), u)$ are witnesses. We convert the signature protocol via the Fiat-Shamir transformation. To hide the other part, the signer encrypts $(\bar{\sigma}_1, \bar{\pi})$ and $(\bar{\sigma}_1', \bar{\pi}')$ via the Cramer-Shoup encryption scheme [29] by using the public key of the group manager. 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 [51, 63] 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 $\theta$ is re-used for encrypting plural messages. In addition to these LMPY signatures, the signer also encrypts $\text{V}_{\text{ID}} = \text{V}_{\text{ID}}^0$ and $\text{V}_u = \text{V}_u^0$. The former is required to search the corresponding certificate $\text{cert}_i = (i, \text{Path}(i), \text{V}_{\text{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, \text{V}_u^{(j)})\}_{u_j \in \text{Path}(i)})$ from joining transcripts in the open algorithm. The latter is also required to search such that $\text{V}_u = \text{V}_u^j$ for obtaining $u_j \in \text{Path}(i)$ in the open algorithm. Finally, the signer proves the knowledge of $\text{ID}$, $u$, and $\theta$, and also proves that two LMPY signatures sign on the same node $u$.

We give our revocable group signature scheme as follows.

Setup$(1^\lambda, N)$: Choose bilinear groups $\text{cp} = (G, \hat{G}, G_T, p)$ where $p > 2^8$ and $g \overset{\$}{\leftarrow} G$ and $\hat{g} \overset{\$}{\leftarrow} \hat{G}$. Choose $g' \overset{\$}{\leftarrow} G$ and $\hat{g}' \overset{\$}{\leftarrow} \hat{G}$. Choose a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, which is modeled as a random oracle.

1. Generate Two Key Pairs of the LMPY Scheme: Choose $v = (v_1, v_2, W) \overset{\$}{\leftarrow} G^3$ and $v' = (v_1', v_2', W') \overset{\$}{\leftarrow} G^3$. 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 $M$ and $M'$ as follows.
M = \begin{pmatrix}
g & 1_G & 1_G & 1_G & 1_G & 1_G & h \\
 v_1 & g & 1_G & 1_G & h & 1_G & 1_G \\
 v_2 & 1_G & g & 1_G & h & 1_G & 1_G \\
 W & 1_G & g & 1_G & h & 1_G \\
\end{pmatrix}

M' = \begin{pmatrix}
g' & 1_G & 1_G & 1_G & 1_G & h' \\
v_1' & g' & 1_G & 1_G & h' & 1_G \\
v_2' & 1_G & g' & 1_G & h' & 1_G \\
W' & 1_G & g' & 1_G & h' & 1_G \\
\end{pmatrix}

Run QA.KeyGen(cp, M) and QA.KeyGen(cp, M') of the QA-NIZK argument, and get crs = \{(v_1^i)_{i=1}^4, \tilde{g}_z, \{\tilde{g}_j\}_{j=1}^8\} and crs' = \{(v_1'^i)_{i=1}^4, \tilde{g}_z', \{\tilde{g}_j'\}_{j=1}^8\}. Set sk = \omega, vk = (cp, g, h, \tilde{g}, v, \Omega = h^\omega, crs), sk' = \omega', and vk' = (cp, g', h', \tilde{g}', v', \Omega' = h'^{\omega'}, crs').

2. Generate a Key Pair of the Cramer-Shoup Scheme (with a Randomness-reuse Variant):
Choose \(x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma \in \mathbb{Z}_p\) 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 \(gpk = (vk, vk', x_z, x_\sigma, x_{ID}, x_u, x'_z, x'_\sigma, y_z, y_\sigma, y_{ID}, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)\), \(S_{GM} = (sk, sk')\), and \(S_{OA} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)\).

Join\textsuperscript{GM,\mathcal{U}}: A signer \(U_i\) and GM run the following interactive protocol.

1. \(U_i\) chooses \(ID_i \leftarrow \mathbb{Z}_p\), computes \(V_{ID} = v_{1, ID}, Z_{ID} = z_{2, ID}, \tilde{G}_{2, ID} = \tilde{g}_{2, ID}, \) and \(\tilde{G}_{5, ID} = \tilde{g}_{5, ID}\), and sends \((V_{ID}, Z_{ID}, \tilde{G}_{2, ID}, \tilde{G}_{5, ID})\) to GM.

2. If \(V_{ID}\) has been appeared in transcripts of \(St\), then GM aborts. Otherwise, GM checks the following equations hold.
\[
e(V_{ID}, \tilde{g}_z) = e(v_1, \tilde{G}_{2, ID}), e(Z_{ID}, \tilde{g}_2) = e(z_2, \tilde{G}_{2, ID})
\]
\[
e(V_{ID}, \tilde{g}_5) = e(v_1, \tilde{G}_{5, ID}),
\]
If all tests pass, GM samples a fresh index \(i \in \mathbb{Z}_p\) and a fresh leaf node (then \(Path(i)\) is fixed), and sends \(i\) to \(U_i\). Otherwise, GM aborts.

3. Prove the Knowledge of \(ID_i\): \(U_i\) runs an interactive zero-knowledge proof of knowledge of \(ID_i = \log_{v_1}(V_{ID})\) in interaction with GM. We employ the Cramer-Damgård-Mackenzie transformation [28] which converts a sigma protocol into a perfect zero-knowledge proof of knowledge. We give the 4-round protocol in the Appendix.\textsuperscript{2} Let \(\pi_K(\text{ID})\) be the interaction transcript.

4. Generate LMPY Signatures as a Certificate: For all \(u_j \in Path(i)\), \(U_i\) chooses \(s_j \leftarrow \mathbb{Z}_p\) and computes a LMPY signature on messages \((ID_i, u_j)\) by using \(sk = \omega\) such that \(V_u^{(j)} = v_{2, u}^{s_j}\) and \(\sigma_{j, 1} = g^{\omega(V_{ID} \cdot V_u^{(j)} \cdot W)^{s_j}}, \sigma_{j, 2} = g^{s_j}, \sigma_{j, 3} = h^{s_j}, \) and \(\pi_j = z_5^{\omega}(Z_{ID} \cdot z_3^{h_j} \cdot z_4)^{s_j}\). Finally, GM sends \(\text{cert}_i = \{i, Path(i), V_{ID}, \{s_{j, 1}, s_{j, 2}, s_{j, 3}, \pi_j, V_u^{(j)}\}_{u_j \in Path(i)}\}\) to \(U_i\).

\textsuperscript{2}As mentioned by Libert et al. [51], we can remove this interaction by using an extractable commitment. For example, Delerablée and Pointcheval [30] employed the Paillier encryption [64] as the underlying extractable commitment scheme. Then, the decryption key allows the extraction.
5. Finally, GM stores transcript\(_i\) = ((Z\(_{ID}\), \(\hat{G}\), \(\hat{G}_{5, ID}\), \(\pi_K\) (ID), cert\(_i\)) and \(U_i\) stores (cert\(_i\), sec\(_i\)) where sec\(_i\) = ID\(_i\).

Revoke(gpk, \(S_{GM}\), \(t, \mathcal{R}_t \subset S_{\text{users}}\)): Run Y \(\leftarrow\) CS(BT, \(\mathcal{R}_t\)). For all \(u_j \in Y\), GM chooses \(s_j \overset{\$}{\leftarrow} \mathbb{Z}_p\) and computes a LMPY signature on messages \((t, u_j)\) by using sk\(_j' = \omega_j\) such that \(\sigma_j' = g^{\omega_j} (v_j^t \cdot v_j^u \cdot W)^{\omega_j}\), \(\sigma_j' = h^{\omega_j}\), and \(\pi_j' = z_1^\omega (z_2^t \cdot z_3^u \cdot z_4^j)^{\omega_j}\). Output RL\(_t\) = (t, Y, \{\(\sigma_j'\), \(\sigma_j'\), \(\sigma_j'\), \(\pi_j'\), u\}_{u_j \in Y}).

Sign(gpk, \(t, RL_t\), cert\(_i\), sec\(_i\), \(M\)): Let \(u\) be a node where \(u \in \text{Path}(i) \cap Y\). If \(u\) does not exist, then output \(\bot\). Let \((\sigma_1, \sigma_2, \sigma_3)\) be a LMPY signature on \((\text{ID}_i, u)\) contained in cert\(_i\), and let \((\sigma_1', \sigma_2', \sigma_3')\) be a LMPY signature on \((t, u)\) contained in RL\(_t\). Choose \(s, s' \overset{\$}{\leftarrow} \mathbb{Z}_p\) and re-randomize signatures as follows.

\[
\tilde{\sigma}_1 = \sigma_1 \cdot (v_1^{\text{ID}} \cdot v_2^u \cdot W)^{\tilde{s}}, \tilde{\sigma}_2 = \sigma_2 \cdot g^{s}, \tilde{\sigma}_3 = \sigma_3 \cdot h^{s}, \tilde{\pi} = \pi \cdot (z_2^{\text{ID}} \cdot z_3^u \cdot z_4^j)^{s}
\]

\[
\tilde{\sigma}_1' = \sigma_1' \cdot (v_1^t \cdot v_2^u \cdot W)^{s'}, \tilde{\sigma}_2' = \sigma_2' \cdot g^{s'}, \tilde{\sigma}_3' = \sigma_3' \cdot h^{s'}, \tilde{\pi}' = \pi' \cdot (z_2^t \cdot z_3^u \cdot z_4^j)^{s'}
\]

Choose \(\theta \overset{\$}{\leftarrow} \mathbb{Z}_p\) and compute a ciphertext of the Cramer-Shoup encryption \(C_{CS} = (C_1, C_2, C_3, C_4, C_5, C_6, C_7)\) such that

\[
C_1 = g^\theta, \ C_2 = h^\theta, \ C_3 = \tilde{\pi} \cdot X_1, \ C_4 = \tilde{\sigma} \cdot X_2
\]

\[
C_{ID} = v_1^{\text{ID}} \cdot X_{ID},\ C_u = v_2^u \cdot X_u,\ C_z = \tilde{\sigma}_1' \cdot X_{\theta}^c
\]

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

\[
R_5 = (e(X_z, g_3^e) e(X_\sigma, g_1^e))^{r_\theta} (e(\tilde{\sigma}_2, g_3^e) e(\sigma_3, g_5^e))^{-r_{\text{ID}}} (e(\tilde{\sigma}_2, g_3^e) e(\sigma_3, g_6^e))^{-r_u}
\]

\[
R_6 = (e(X'_z, g_3^e) e(X'_\sigma, g_1^e))^{r_\theta} (e(\tilde{\sigma}'_2, g_3^e) e(\sigma'_3, g_6^e))^{-r_u}
\]

Compute \(c \leftarrow H(gpk, t, C_{CS}, \tilde{\sigma}_2, \tilde{\sigma}_3, \tilde{\sigma}_2', \tilde{\sigma}_3', R_1, \ldots, R_6, M), \) s\(_{ID} = r_{ID} + c \cdot \text{ID}_i, \ s_\theta = r_\theta + c \cdot \theta, \) and \(s_u = r_u + c \cdot u\).

Output a group signature \(\Sigma = (C_{CS}, \tilde{\sigma}_2, \tilde{\sigma}_3, \tilde{\sigma}_2', \tilde{\sigma}_3', c, s_{ID}, s_\theta, s_u) \in \mathbb{G}_1^{12} \times \mathbb{Z}_p^4\).

Verify(gpk, \(t, RL_t, \Sigma, M\)): Compute \(\bar{R}_1 = g^{s_{\theta}} \cdot C_1^c, \bar{R}_2 = h^{s_{\theta}} \cdot C_2^c, \bar{R}_3 = v_1^{\text{ID}} \cdot X_{ID}^{s_{\theta}} \cdot C_{ID}^c, \bar{R}_4 = v_2^u \cdot X_u^{s_{\theta}} \cdot C_u^c, \) and \(R_5\) and \(R_6\) such that

\[
\bar{R}_5 = (e(X_z, g_3^e) e(X_\sigma, g_1^e))^{s_{\theta}} (e(\tilde{\sigma}_2, g_3^e) e(\sigma_3, g_5^e))^{-s_{\text{ID}}} (e(\tilde{\sigma}_2, g_3^e) e(\sigma_3, g_6^e))^{-s_u}
\]

\[
\times (e(C_z, g_3^e) e(C_{\sigma}, g_1^e) e(\sigma_2, g_4^e) e(\sigma_3, g_7^e) e(\Omega, g_8^e))^{-c}
\]

\[
\bar{R}_6 = (e(X'_z, g_3^e) e(X'_\sigma, g_1^e))^{s_{\theta}} (e(\tilde{\sigma}'_2, g_3^e) e(\sigma'_3, g_6^e))^{-s_u}
\]

\[
\times (e(C'_z, g_3^e) e(C'_\sigma, g_1^e) e(\sigma'_2, g_4^e) e(\sigma'_3, g_7^e) e(\Omega, g_8^e))^{-c}
\]

Output 1 if \(c = H(gpk, t, C_{CS}, \tilde{\sigma}_2, \tilde{\sigma}_3, \tilde{\sigma}_2', \tilde{\sigma}_3', R_1, \ldots, R_6, M)\) and 0 otherwise.

\(^3\text{We can easily see that the underlying sigma protocol has unique responses. Let all values, except resp = (s_{ID}, s_\theta, s_u), be fixed. Then, assume that an accepted response (s_{ID}', s_\theta', s_u') \neq (s_{ID}, s_\theta, s_u) exists. Then, from g^{s_{\theta}} \cdot C_1^c = g^{s_{\theta}} \cdot C_1^c, s_{\theta}' = s_{\theta}' holds. From v_1^{\text{ID}} \cdot X_{ID}^{s_{\theta}} \cdot C_{ID}^c = v_1^{\text{ID}} \cdot X_{ID}^{s_{\theta}} \cdot C_{ID}^c and s_{ID}' = s_{ID}, s_{\theta}' = s_{\theta}' holds. From v_2^u \cdot X_u^{s_{\theta}} \cdot C_u^c = v_2^u \cdot X_u^{s_{\theta}} \cdot C_u^c and s_u' = s_u' holds. Thus, (s_{ID}', s_\theta', s_u') \neq (s_{ID}, s_\theta, s_u) holds and this shows that the sigma protocol has unique responses, and the NIZK proof system converted by the Fiat-Shamir transformation is simulation sound.}\)
Open(gpk, S_{OA}, t, \Sigma, M, St):

1. If Verify(gpk, t, RL_t, \Sigma, M) = 0, then output \perp.

2. Otherwise, decrypt \((C_1, C_2, C_3, C_{ID}, C_u, C'_u, C'_j, C'_t, C''_t)\) by using \((x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_t, y'_t)\) such that \(\sigma_1 = \sigma_\sigma \cdot C_1^{-x_z} \cdot C_2^{-y_z}, \pi = C_2 \cdot C_1^{-x_z} \cdot C_2^{-y_z}, V_{ID} = C_{ID} \cdot C^{-x_{ID}}, C_2^{-y_{ID}}, V_u = C_u \cdot C_1^{-x_u} \cdot C_2^{-y_u}, \sigma'_1 = C_{sigma} \cdot C_1^{-x'_z} \cdot C_2^{-y'_z}, \) and \(\pi' = C_2' \cdot C_1^{-x'_z} \cdot C_2^{-y'_z} \).

3. Search \(V_{ID}\) from in the database of joining transcriptes and get \((G_{2, ID}, G_{5, ID})\). If there is no such entry, then output \perp.

4. Let \(V_{ID}\) be contained in cert, where \(cert = (i, Path(i), V_{ID}, \{(\sigma_{j, 1}, \sigma_{j, 2}, \sigma_{j, 3}, \pi_j, V_u^{(j)})\}_{u_j \in Path(i)})\). Search \(j\) such that \(V_u = V_u^{(j)}\) and obtain \(u_j \in Path(i)\). If there is no such \(j\), then output \perp.

5. Check whether \((\sigma_1, \bar{\sigma}_2, \bar{\sigma}_3, \pi)\) and \((\sigma'_1, \bar{\sigma}'_2, \bar{\sigma}'_3, \pi')\) are valid LMPY signatures as follows.

\[
e(\Omega, \hat{g}_8)^{-1} = e(\pi, \hat{g}_z) e(\sigma_1, \hat{g}_1) e(\bar{\sigma}_2, \hat{g}_{2, ID} \cdot \hat{g}_{3, u} \cdot \hat{g}_4) e(\bar{\sigma}_3, \hat{g}_{5, ID} \cdot \hat{g}_{6, u} \cdot \hat{g}_7)
\]

\[
e(\Omega', \hat{g}'_8)^{-1} = e(\pi', \hat{g}'_z) e(\sigma'_1, \hat{g}'_1) e(\bar{\sigma}'_2, \hat{g}'_{2, ID} \cdot \hat{g}'_{3, u} \cdot \hat{g}'_4) e(\bar{\sigma}'_3, \hat{g}'_{5, ID} \cdot \hat{g}'_{6, u} \cdot \hat{g}'_7)
\]

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 \(i^* \notin U^a \setminus R_t\), 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 \(Q_{a, join}\) queries, and \(R_t\) is the set of revoked signers at time \(t^*\). Remark that \(i\) may be \(\perp\). We divide the condition \(i^* \notin U^a \setminus R_t\) to (1) \(i^* \notin U^a\) and (2) \(i^* \in R_t\). The first case is the same as that of the proof given by Libert et al. [51], 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 \(Q_{a, join}\) 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 \((G, \hat{G}, G_T)\) in the random oracle model.

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

**Case (1):** In this case, we assume that \(i^* \notin U^a\). First, \(B\) chooses bilinear groups \(cp = (G, \hat{G}, G_T, \hat{g})\) where \(p > 2^4\), vectors \(v\) and \(v'\), and matrices \(M\) and \(M'\) with the discrete logarithms of the group elements of \(M\) and \(M'\) (i.e., \(M\) and \(M'\) are witness-samplable). \(B\) sends \(cp\) and \(M\) to \(C\) and obtains \(crs = \{(z_i \mid i = 1)^4, \hat{g}_z, \{\hat{g}_j\}_{j=1}^8\}\). \(B\) runs QA.KeyGen\((cp, M')\) and generates \(sk' = \omega'\) and
follows. First, Proof: Let the proposed group signature scheme is secure against framing attacks under the SDL Theorem 5.2. Q sign$(\mathcal{C})$ its message $M$ (Case (2): In this case, we assume that $B$ outputs $(s, u^*)$ and sets $y^*$. Thus, $B$ decrypts $B$ outputs $(ID^*, u^*)$ as a forging and breaks the unforgeability of the LMPY scheme.

Case (2): In this case, we assume that $i^* \in \mathcal{R}_{\mathcal{C}}$. As in the first case, $B$ chooses $\mathcal{C}$. $B$ sends $\mathcal{C}$ and $\mathcal{M}$ to $C$ and obtains $\mathcal{S}$. $B$ runs QA.KeyGen$(\mathcal{C}, \mathcal{M})$ and generates $sk = (\mathcal{C})$, $vk = (\mathcal{C})$, $\Omega = \mathcal{C}^*$, and sets $\mathcal{S}$. $B$ outputs $(ID^*, u^*)$. $B$ decrypts $(ID^*, u^*)$ and $(\sigma_1, \sigma_2', \sigma_3')$ as a forging and breaks the unforgeability of the LMPY scheme.

Q$_{\text{revoke}}$: Since $B$ knows $sk^* = \omega^*$, $B$ just runs the Revoke algorithm as usual.

At some point $t^*$, $A$ outputs a forged group signature $\Sigma^* = (C^*_{\mathcal{C}}, \sigma_1^*, \sigma_2^*, \sigma_3^*, c^*, s_{ID}^*, s_{u}^*, s_{a}^*)$ and its message $M^*$. By rewinding $A$, due to the generalized Forking Lemma [15], $B$ obtains two valid group signatures $(C^*_{\mathcal{C}}, \sigma_1^*, \sigma_2^*, \sigma_3^*, c^*, s_{ID}^*, s_{u}^*, s_{a}^*)$ and $(C^*_{\mathcal{C}}, \sigma_2^*, \sigma_3^*, c^*, s_{ID}^*, s_{u}^*, s_{a}^*)$. $B$ computes $(s_{1}^* \times s_{2}^*) / (c^* - c^*)$ and $u^* = (s_{1}^* - s_{2}^*) / (c^* - c^*)$. Moreover, $B$ decrypts $C^*_{\mathcal{C}}$ and obtains $(\sigma_1, \pi)$ where $(\sigma_1, \sigma_2', \sigma_3', \pi)$ is a valid LMPY signature on $(ID^*, u^*)$, and due to the winning condition, $B$ has not sent $(ID^*, u^*)$ as a signing query. Thus, $B$ outputs $(ID^*, u^*)$ as a forging and breaks the unforgeability of the LMPY scheme.

Q$_{\text{revoke}}$: First, $B$ runs $Y \leftarrow \mathcal{C}$(BT, $\mathcal{R}_{\mathcal{C}}$). For all $u_j \in Y$, $B$ sends $(t, u_j)$ to $C$ as a signing query. Finally, $B$ outputs $RL_t = (t, Y, \{(\sigma_{j1}, \sigma_{j2}, \sigma_{j3}, \pi, u_j)\})_{u_j \in Y}$.

As in the first case, at some point $t^*$, $A$ outputs $\Sigma^*$ and $M^*$. $B$ rewinds $A$ and obtains $u^*$. $B$ decrypts $C^*_{\mathcal{C}}$ and obtains $(\sigma_1^*, \pi^*)$ where $(\sigma_1^*, \sigma_2^*, \sigma_3^*, \pi^*)$ is a valid LMPY signature on $(t^*, u^*)$. Due to the winning condition, $B$ has not sent $(t^*, u^*)$ as a signing query. Thus, $B$ outputs $(t^*, u^*)$ as a forging 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 $A$ be an adversary. We construct an algorithm $B$ that solves the SDL problem by using $A$ as follows. First, $B$ takes an SDL instance $(\mathcal{g}, g', h, g', \mathcal{g})$ as input. $B$ chooses $g', v_2, v_1$, $v_2', W', \mathcal{G}$, $\mathcal{G}'$, $\mathcal{G}$, $\mathcal{S}_a, \alpha_\mathcal{A}, \alpha_\mathcal{W}$, and computes $h = g^{\alpha_\mathcal{A}}$, $v_1 = g^{\alpha_\mathcal{W}}$, $W = g^{\alpha_\mathcal{W}}$, $\mathcal{g}_2 = g^{\alpha_\mathcal{A}}$, and $h' = g^{\alpha_\mathcal{A}}$. In order to compute $\{z_i\}_{i=1}^4$ of $\mathcal{S}$ and $\{z_i\}_{i=1}^4$ of $\mathcal{S}$, $B$ chooses $tk = \{\chi_j\}_{j=1}^4$ and $tk' = \{\chi_j\}_{j=1}^4$ respectively. $B$ sets up other values as usual, i.e., $B$ knows $\mathcal{S}_\mathcal{A} = (\mathcal{C}, \mathcal{S}_\mathcal{A})$, $\mathcal{S}_\mathcal{A} = (x, y, x, y, x, y, x, y, x, y, x, y, x, y, x, y, x, y)$, $tk$, and $tk'$. For $Q_{\text{pub}}$, $Q_{\text{KeyGM}}$, and $Q_{\text{keyOA}}$ queries, $B$ sends $g_{\mathcal{P}_p}$, $\mathcal{S}_\mathcal{G}$, and $\mathcal{S}_\mathcal{A}$ to $A$ respectively.
\(Q_{\text{b-join}}\): \(A\) is allowed to run \(J_{\text{GM}}\). We show how to simulate \(J_{\text{user}}\) as follows. \(B\) chooses \(\delta_i \overset{\$}{\leftarrow} \mathbb{Z}_p\), and implicitly sets \(\text{ID}_i = a \cdot \delta_i\), where \(a\) is the solution of the SDL problem. \(B\) computes

\[
V_{\text{ID}_i} = (g^a)^{\alpha_v \cdot \delta_i} = v_{\text{ID}_i}^1
\]

\[
Z_{\text{ID}_i} = ((g^a)^{\alpha_v \cdot \delta_i})^{-\chi_1}((g^a)^{\delta_i})^{-\chi_2}(g^a)^{\alpha_h \cdot \delta_i} - \chi_5
\]

\[
= (v_{\text{ID}_i}^1)^{-\chi_1}(g^{\text{ID}_i})^{-\chi_2}(h)^{-\chi_5} = z_{\text{ID}_i}^2
\]

\[
\hat{G}_{2,\text{ID}_i} = ((\hat{g}_{2})^{\alpha_v \cdot \delta_i})^{\chi_2} = \left((\hat{g}_{2})^{\alpha_v \cdot \delta_i}\right)^{\alpha_v \cdot \delta_i} = (\hat{g}_{2}^{\chi_2}z_{\text{ID}_i}) = \hat{g}_{2}^{\text{ID}_i}
\]

\[
\hat{G}_{5,\text{ID}_i} = ((\hat{g}_{5})^{\alpha_v \cdot \delta_i})^{\chi_5} = \left((\hat{g}_{5})^{\alpha_v \cdot \delta_i}\right)^{\alpha_v \cdot \delta_i} = \hat{g}_{5}^{\chi_5}z_{\text{ID}_i} = \hat{g}_{5}^{\text{ID}_i}
\]

and sends \((V_{\text{ID}_i}, Z_{\text{ID}_i}, \hat{G}_{2,\text{ID}_i}, \hat{G}_{5,\text{ID}_i})\) to \(A\). Moreover, \(B\) simulates the interactive proof of knowledge of \(\text{ID}_i = \log_{e_1}(V_{\text{ID}_i})\) using the simulator. Finally, \(B\) obtains \(\text{cert}_i = (i, \text{Path}(i), V_{\text{ID}_i}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(i)})\}_{u \in \text{Path}(i)})\).

\(Q_{\text{revoke}}\): Since \(B\) knows \(sk' = \omega')\, B\) just runs the Revoke algorithm as usual.

\(Q_{\text{sig}}\): Since \(B\) knows \(\text{cert}_i\) for \(i \in U^b\), \(B\) can compute \(C_{\text{CS}}\) as usual. Note that for re-randomization process, \(B\) uses \(V_{\text{ID}}\) and \(Z_{\text{ID}}\). \(B\) simulates \((c, s_{\text{ID}}, s_{\theta}, s_u) \overset{\$}{\leftarrow} \mathbb{Z}_p\) and computes \(R_1 = g^{s_{\theta} \cdot C_1^{-c}}, R_2 = h^{s_{\theta} \cdot C_2^{-c}}, R_3 = v_{\text{ID}_i}^{s_{\theta} \cdot X_{\text{ID}_i}^{s_{\theta} \cdot C_1^{-c}}}, R_4 = v_{\text{ID}_i}^{s_{\theta} \cdot X_{\text{ID}_i}^{s_{\theta} \cdot C_2^{-c}}}, \) and \(R_5\) and \(R_6\) such that

\[
R_5 = (e(X_2, \hat{g}_{2})e(X_3, \hat{g}_{3}))^{s_{\theta}}(e(\hat{g}_{2}, \hat{g}_{3})e(\hat{g}_{3}, \hat{g}_{5}))^{s_{\theta}}(e(\hat{g}_{2}, \hat{g}_{3})e(\hat{g}_{3}, \hat{g}_{5}))^{-s_{\theta}}
\]

\[
\times (e(C_2, \hat{g}_{2})e(C_3, \hat{g}_{3})e(\hat{g}_{2}, \hat{g}_{3})e(\hat{g}_{3}, \hat{g}_{5}))^{-c}
\]

\[
R_6 = (e(X_2', \hat{g}_{2}')e(X_3', \hat{g}_{3}'))^{s_{\theta}}(e(\hat{g}_{2}', \hat{g}_{3}')e(\hat{g}_{3}', \hat{g}_{5}'))^{s_{\theta}}
\]

\[
\times (e(C_2', \hat{g}_{2}')e(C_3', \hat{g}_{3}')e(\hat{g}_{2}', \hat{g}_{3}')e(\hat{g}_{3}', \hat{g}_{5}'))^{-c}
\]

\(B\) programs the random oracle \(H\) such that \(c = H(gpk, t, C_{\text{CS}}, \hat{g}_{2}, \hat{g}_{3}, \hat{g}_{5}, R_1, \ldots, R_6, M)\), and returns \(\Sigma = (C_{\text{CS}}, \hat{g}_{2}, \hat{g}_{3}, \hat{g}_{5}, c, s_{\text{ID}}, s_{\theta}, s_u)\) to \(A\).

At some point \(t^*\), \(A\) outputs a forged group signature \(\Sigma^* = (C_{\text{CS}}^*, \hat{g}_{2}^*, \hat{g}_{3}^*, \hat{g}_{5}^*, c^*, s_{\text{ID}}^*, s_{\theta}^*, s_u^*)\) and its message \(M^*\). Let \(t^*\) be the opening result of \(\Sigma^*\). Due to the winning condition, \(t^* \in U^b\) and \(t^*\) did not sign \(M^*\), i.e., \((i^*, t^*, M^*, \Sigma^*) \notin \text{Sigs}\). By rewinding \(A\), due to the generalized Forking Lemma [15], \(B\) obtains two valid group signatures \((C_{\text{CS}}^*, \hat{g}_{2}^*, \hat{g}_{3}^*, \hat{g}_{5}^*, c^*, s_{\text{ID}}^*, s_{\theta}^*, s_u^*)\) and \((C_{\text{CS}}^*, \hat{g}_{2}^*, \hat{g}_{3}^*, \hat{g}_{5}^*, c^*, s_{\text{ID}}^*, s_{\theta}^*, s_u^*)\) with \(c^* \neq c^1\), \(s_{\text{ID}}^* \neq s_{\text{ID}}^1\), and \(s_u^* \neq s_u^1\). \(B\) computes \(ID^* = (s_{\text{ID}}^1 - s_{\text{ID}}^*)/(c^* - c^1)\) mod \(p\). Since \(i^* \in U^b\), \(B\) has chosen \(\delta_i^*\) which satisfies \(ID^* = a \cdot \delta_i^*\). \(B\) computes \(a = ID^*/\delta_i^*\) mod \(p\), and solves the SDL problem.

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

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

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

\textbf{Game 1:} This game is the same as Game 0 except that if for \((\sigma_j^*, \sigma_j^2, \sigma_j^3, \pi^*)\) chosen by \(A\) as either \(\text{cert}_0^*\) or \(\text{cert}_1^*\), \(\log_{g^a}(\sigma_j^2) \neq \log_{g^a}(\sigma_j^3)\) holds, then the challenger aborts. This contradicts the soundness of the QA-NIZK argument, and as in [51], \(|Pr[S_1] - Pr[S_0]|\) is bounded by the advantage of the DDH problem in \(\hat{G}\), and is negligible.
Game 2: This game is the same as Game 1 except that we change the response of a $Q_{\text{open}}$ query. For $\Sigma = (C_{\text{CS}}, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4, c, s_{\text{ID}}, s_\theta, s_u)$ where $\log_q(\tilde{s}_2) \neq \log_q(\tilde{s}_4)$ 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 $\mathbb{G}$, 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 $\Sigma^*$. For $(s_{\text{ID}}^*, s_\theta^*, s_u^*)$, choose $c, s_{\text{ID}}, s_\theta, s_u \leftarrow \mathbb{Z}_p$ and computes from $R_1$ to $R_6$ and program the random oracle $H$ as in the simulation of $Q_{\text{sig}}$ in the proof of security against framing attacks. Since the probability that $(c, s_{\text{ID}}, s_\theta, s_u)$ 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 $S_{\text{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)$ to compute $C_{\text{CS}}^* = (C_1^*, C_2^*, C_3^*, C_4^*, C_{ID}, C_u^*, C_{ID}^*, C_u^*)$. That is,

$$C_z^* = \tilde{x} \cdot (C_1^*)^{x_z}(C_2^*)^{y_z}, \quad C_\sigma^* = \tilde{x}_1 \cdot (C_1^*)^{x_\sigma}(C_2^*)^{y_\sigma}$$

$$C_{ID}^* = v_1^{ID} \cdot (C_1^*)^{x_{ID}}(C_2^*)^{y_{ID}}, \quad C_u^* = v_2 \cdot (C_1^*)^{x_u}(C_2^*)^{y_u}$$

$$C_{ID}^* = \tilde{x}' \cdot (C_1^*)^{x'_{ID}}(C_2^*)^{y'_{ID}}, \quad C_\sigma^* = \tilde{x}'_1 \cdot (C_1^*)^{x'_\sigma}(C_2^*)^{y'_\sigma}$$

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 $\theta \leftarrow \mathbb{Z}_p$, replace $C_2^* = h^\theta$ to $C_2^* = h^{\theta+\theta'}$. Due to Lemma 1 in [51], $|\Pr[S_5] - \Pr[S_4]|$ 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 $h \leftarrow \mathbb{G}$ in the setup phase, choose $\alpha \leftarrow \mathbb{Z}_p$, set $h = g^\alpha$, and returns $\perp$ if $C_2 \neq C_1^*$ for $\Sigma = (C_{\text{CS}}, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4, c, s_{\text{ID}}, s_\theta, s_u)$. Due to Lemma 2 in [51], the probability that this event happens is bounded by $q_0 \cdot q_H/p$ where $q_0$ is the number of opening queries and $q_H$ is the number of hash queries.

In Game 6, $\Sigma^*$ perfectly hides $(\tilde{x}, \tilde{x}_1, v_1^{ID}, v_2, \tilde{x}', \tilde{x}_1')$, and $\Pr[S_6] = 1/2$. 

6 Application to Identity Management

Ishikawa et al. [43] proposed an identity management system using 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, Ishikawa et al. required that the underlying group signature scheme can separate the keys for each manager. Fortunately, our scheme matches this requirement. Concretely, for $S_{GM} = (sk, sk')$, and $S_{OA} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)$, we can set $sk$ as the issuing key, $sk'$ as the revocation key, and $S_{OA}$ as the opening key.
Isshiki et al. employed the Camenisch-Groth group signature scheme [23] with a slight modification to provide a membership certificate update mechanism. The Camenisch-Groth group signature scheme employs composite-order groups and elliptic curves, and for which Ishiki et al. set 2048-bit and 169-bit orders, respectively. Then, they showed that the running time of the signing algorithm was approximately 135 mssec, and that of the verification algorithm was approximately 112 mssec, which are slower than those of our scheme (see Section 7). Moreover, each user is required to update their membership certificate when revocation occurs. Ishiki 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 [16], the OA provides a proof for opening, i.e., ownership of group signatures. Sakai et al. [68] 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 [31] 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 Ishiki 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. 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 OpenPoO when the algorithm outputs a non-interactive proof of the ownership \( \tau \). Let \( upk_i \) be the public key of user \( U_i \). We assume that \( upk_i \) is computed by \( U_i \) in the join phase.

\[ \text{OpenPoO}(gpk, S_{OA}, t, \Sigma, M, St): \text{The modified opening algorithm takes as input } gpk, S_{OA}, t, \Sigma, M, \text{and } St := (St_{\text{users}}, St_{\text{trans}}), \text{and outputs } \bot \text{ if the opening is failure, and } (i, \tau) \text{ such that } ID_i \in St_{\text{users}} \cup \{ \bot \}, \text{otherwise. Here, } \tau \text{ is a non-interactive proof of the ownership of } \Sigma. \]

\[ \text{Judge}(gpk, i, upk_i, \Sigma, M, \tau): \text{The judgement algorithm takes as input } gpk, i, upk_i, \Sigma, M, \text{and } \tau, \text{and outputs } 1 \text{ if } (\Sigma, M) \text{ is provided by } U_i, \text{and } 0 \text{ otherwise.} \]

Next, we modify our scheme as follows. Let \( upk_i = v_1^{ID_i} \). Since \( v_1^{ID_i} \) has been contained in \( cert_i \) in the original scheme, publishing \( upk_i \) does not affect its security. Moreover, we introduce another random oracle \( H' : \{0, 1\}^* \rightarrow \mathbb{Z}_p. \)

\[ \text{OpenPoO}(gpk, S_{OA}, t, \Sigma, M, St): \text{In addition to the original opening procedure, the algorithm computes } \tau \text{ as follows when the original algorithm outputs } i. \text{ Briefly, } \tau \text{ is a NIZK proof of } (x_{ID}, y_{ID}) \text{ satisfying the relation } X_{ID} = g^{x_{ID}} h^{y_{ID}} \text{ and } upk_i = C_{ID} \cdot C_1^{-x_{ID}} \cdot C_2^{-y_{ID}}. \text{ That is, the relation shows that the decryption result of } C_{ID} \text{ by using } (x_{ID}, y_{ID}) \text{ contained in } S_{OA} \text{ is } upk_i. \]

\begin{enumerate}
    \item Choose \( r'_{x_{ID}}, r'_{y_{ID}} \overset{\$}{\leftarrow} \mathbb{Z}_p. \)
    \item Compute \( R_{ID} = g^{r'_{x_{ID}}} h^{r'_{y_{ID}}} \) and \( R_{C_{ID}} = C_1^{-r'_{x_{ID}}} C_2^{-r'_{y_{ID}}} \).
\end{enumerate}

\[ ^4 \text{This group signature scheme is listed as Mechanism 5 in ISO/IEC 20008-2 [3].} \]

\[ ^5 \text{More concretely, in the definition of security against misidentification attacks, an adversary can obtain } cert_i \text{ via the } Q_{\text{join}} \text{ oracle. In the definition of Non-frameability, an adversary can obtain } cert_i \text{ via the } Q_{\text{join}} \text{ oracle. In the definition of Anonymity, an adversary can generate all certificates by myself using } S_{GM}. \]
3. Compute $c' = H'(gpk, t, C_{ID}, C_1, C_2, R_{ID}, R_{C_{ID}})$ and $s'_{x_{ID}} = r'_{x_{ID}} + c'x_{ID}$ and $s'_{y_{ID}} = r'_{y_{ID}} + c'y_{ID}$.

4. Output $(i, \tau)$ where $\tau = (c', s'_{x_{ID}}, s'_{y_{ID}})$.

Judge$(gpk, i, upk_i, \Sigma, M, \tau)$: If $\text{Verify}(gpk, t, R_{ID}, \Sigma, M) = 0$, then output 0. Otherwise, compute $R_{ID} = g'^{c_{ID}}h^{s_{ID}}C_{ID}^{c'}$ and $\tilde{R}_{C_{ID}} = C_{1}^{s_{1_{ID}}}C_{2}^{s_{2_{ID}}}$(upk$_i$/C$_{ID}$)$^{-c'}$. Output 1 if $c' = H'(gpk, t, C_{ID}, C_1, C_2, R_{ID}, \tilde{R}_{C_{ID}})$, 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 $\Sigma$ 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 $\tau^*$. The adversary wins if $i \neq i^*$ and Judge$(gpk, i^*, upk_{i^*}, \Sigma, M, \tau^*) = 1$.

If one can provide $\tau$ 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 $\tau$. This shows that the scheme provides weak opening soundness under the DL assumption. As a remark, we need to modify the definition of the Qopen oracle where it outputs $\tau$ in addition to the opening result. Then, in the proof of Anonymity, we need to construct a simulator that produces $\tau$ without knowing $S_{OA}$. 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 implemented our scheme by employing Barreto-Lynn-Scott (BLS) curves [11] of embedding degree 12 over a 455-bit prime field (BLS-12-455), which has around 128-bit security according to the Barbulescu-Duquesne security estimation [9, 10]. We also implemented the scheme by employing Barreto-Naehrig (BN) curves [12] of embedding degree 12 over a 382-bit prime field (BN-12-382). Although BN-12-382 cannot ensure 128-bit security from the security estimation, we also give the results of BN-12-382 because of its small signature size; actually, we will show that the signature size is smaller than that of the Ohara et al. scheme. 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($G_1$,Type), Mul($G_2$,Type) and Exp($G_T$,Type) are scalar multiplication in $G_1$ and $G_2$, and exponentiation in $G_T$, 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 $G_T$ since the RELIC library does not support Type K in $G_T$.

Table 3 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 fixed and it is unnecessary to recompute them during time $t$, these values are first precomputed then used at Sign/Verify/Open in our implementation. In Table 3, “Precompute” rows show the timings of the precomputation phase, and “Online” rows show the actual Sign/Verify/Open timings after the precomputation. Note that the Open, Open$^{PoO}$, 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 Open$^{PoO}$ algorithm does not contain the running time of the Open algorithm when a proof $\tau$ is additionally produced.

Next, we evaluated the signature size in Table 4. In our scheme, a signature contains 16 group elements (12 elements in $G$ and 4 elements in $Z_p$) whereas in the Ohara et al. scheme, a signature
contains 18 group elements (5 elements in $G$ and 13 elements in $\mathbb{Z}_p$). When the BLS-12-455 is employed, the sizes of the scalar value in $\mathbb{Z}_p$, an element in $G$, an element in $\hat{G}$, and an element in $G_T$ were 39 bytes, 58 bytes, 115 bytes, and 456 bytes, respectively. Then, the actual signature size of our scheme is slightly larger than that of the Ohara et al. scheme, 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 $G$ compared to that of Ohara et al. scheme, and the size of element in $G$ is slightly but significantly larger than that of a value in $\mathbb{Z}_p$ on BLS-12-455. On the other hand, when the BN-12-382 is employed, the size of a value in $\mathbb{Z}_p$ is almost the same as that of an element in $G$, indeed, the sizes of a scalar value in $\mathbb{Z}_p$, an element in $G$, an element in $\hat{G}$, and an element in $G_T$ are 48 bytes, 49 bytes, 97 bytes, and 384 bytes, respectively. In this case, our signature size is 780 bytes whereas that of the Ohara et al. scheme is 869 bytes. Remark that, as mentioned in the beginning of this section, the BN-12-382 seems to have less than 128-bit security. Therefore, we can say that, if such a slightly lower-level security is accepted, our signature size is smaller than that of the Ohara et al. scheme.
<table>
<thead>
<tr>
<th>Scheme</th>
<th>$G$</th>
<th>$Z_p$</th>
<th>Signature Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ohara et al. [63]</td>
<td>5</td>
<td>13</td>
<td>$797 \ (\text{BLS-12-455})/869 \ (\text{BN-12-382})$</td>
</tr>
<tr>
<td>Ours</td>
<td>12</td>
<td>4</td>
<td>$852 \ (\text{BLS-12-455})/780 \ (\text{BN-12-382})$</td>
</tr>
</tbody>
</table>

### 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. [20] 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 $N/2$ [62]. 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 [21] satisfies full dynamicity. Later, Lai et al. [49] 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 ununlinkability based on accountable ring signatures. Recently, Ling et al. proposed a fully dynamic group signature scheme from lattices [57].

Backes, Hanzlik, and Schneider [8] considered membership privacy by pointing out that in the accountable ring signature scheme [21], 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 $G$ elements, 15 $\hat{G}$ elements, and one $Z_p$ element. As another type of membership privacy, Emura et al. [36] considered hiding the number of revoked signers since if the number is revealed, one may guess the reason behind such circumstances, which may lead to harmful rumors.

Kiayias and Zhou [46] and Chow et al. [27] proposed hidden identity-based signatures where 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 Number JP16K00198.

### References


**Appendix**

In this appendix, we introduce a 4-round protocol for proving the knowledge of $ID = \log_v(V_{ID})$ converted by the Cramer-Damgård-MacKenzie transformation [28]. 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 $ID = \log_v(V_{ID})$ (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 Sigma ∨ Sigma'. We give a zero-knowledge proof of knowledge protocol for $ID = \log_v(V_{ID})$ as follows.

24
Veriﬁer: Choose $\text{ID}, \tilde{c} \xleftarrow{} Z_p$ and compute $a = v_1^{\text{ID}} \cdot V^{-\tilde{c}}_{\text{ID}}$. Choose $r_{\text{ID}}, \tilde{c} \xleftarrow{} Z_p$ and compute $R' = v_1^{r_{\text{ID}}} \cdot V^{-r_{\text{ID}}}$. Send $(a, R')$ to the prover.

Prover: Choose $\hat{c}, r_{\text{ID}} \xleftarrow{} Z_p$ and compute $R = v_1^{r_{\text{ID}}}$. Choose $s'_{\text{ID}}, s'_{\tilde{c}}, \tilde{c}' \xleftarrow{} Z_p$ and compute $R'' = v_1^{s'_{\text{ID}}} \cdot V^{-s'_{\text{ID}}} \cdot a^{-\tilde{c}'}$. Send $(\hat{c}, R, R'')$ to the veriﬁer.

Veriﬁer: Compute $s_{\text{ID}} = r_{\text{ID}} + \hat{c} \cdot \text{ID}$ and $s_{\tilde{c}} = r_{\tilde{c}} + \hat{c} \cdot \tilde{c}$. Choose $c \xleftarrow{} Z_p$. Send $(s_{\text{ID}}, s_{\tilde{c}}, c)$ to the prover.

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

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