Achieving Almost-Full Security for Lattice-based Fully Dynamic Group Signatures with Verifier-local Revocation

Maharage Nisansala Sevwandi Perera and Takeshi Koshiba

[1] Graduate School of Science and Engineering
Saitama University, Japan
perera.m.n.s.119@ms.saitama-u.ac.jp,
[2] Faculty of Education and Integrated Arts and Sciences
Waseda University, Japan
tkoshiba@waseda.jp

Abstract. This paper presents a lattice-based group signature scheme that provides both member registration and member revocation with verifier-local revocation. Verifier-local revocation (VLR) seems to be the most suitable revocation approach for any group since when a member is revoked VLR requires to update only verifiers. However, presenting a fully dynamic and fully secured lattice-based group signature scheme with verifier-local revocation is a significant challenge. Thus, we suggest a new security notion to prove the security of VLR schemes with member registration. As a result, we present a dynamical-almost-fully secured fully dynamic group signature scheme from lattices with VLR.

Keywords: lattice-based group signatures, verifier-local revocation, almost-full anonymity, dynamical-almost-full anonymity, member registration

1 Introduction

In the setting of group signature scheme, each group member is capable of signing messages on behalf of the group anonymously (*anonymity*). On the other hand, the group manager should be able to identify the misbehaved members (*traceability*). Group signatures were initially introduced by Chaum and van Heyst [13] in 1991 and later made it scalable and collusion-resistance by Ateniese et al. [3]. The model proposed by Bellare et al. [4] (BMW03 model) gave the formal and strong security notions, "*full-anonymity*" and "*full-traceability*" for static group signatures. Then, Bellare et al. [5] presented a scheme which uses the BMW03 model to deliver a dynamic group signature scheme. However, their scheme supports only member registration. Recently, Bootel et al. [7] provided a security definition for fully dynamic group signatures.

Lattice-based cryptography has been an interesting research topic during the last decade since it provides security against the threat of quantum attacks. The first lattice-based group signature scheme was proposed by Gordon et al. [16].

Then by achieving anonymity with a token system Camenisch et al. [12] extended the scheme proposed in [16]. Both of the schemes faced a problem of increasing the size of the signatures with the number of group members N. However, the scheme proposed by Laguillaumie et al. [18] was able to give a solution for the linear size problem. But, the scheme in [18] required relatively large parameters. Later, Ling et al. [22] provided a scheme with several prominent advantages such as simple construction and shorter size of keys and signatures. Moreover, their scheme has a strong security with the requirements of the BMW03 model. Nguyen et al. [26] also presented a simpler efficient scheme from lattices. However, none of the above schemes support dynamic groups.

The first lattice-based group signature that facilitates member revocation was proposed by Langlois et al. [19] in 2014. Their scheme manages the member revocation with *Verifier-local Revocation* (*VLR*) approach which requires updating only the verifiers with the latest information about revoked members. However, they were unable to maintain member registration. Moreover, the scheme in [19] relies on a weaker security notion called *selfless-anonymity*. The scheme proposed by Libert et al. [20] provides only member registration. However, full dynamicity is achieved by the scheme suggested by Ling et al. [23]. They have employed accumulators to update member information when a member is revoked or registered. Thus, the first lattice-based fully dynamic group signature scheme [23] uses Merkle tree accumulators. However, when a new member joins the group, the group manager has to update registration table, Merkle tree, and the user counter. Further, when a member is revoked, the group manager has to update both the registration table and the tree. When signing and verifying, the members and the verifiers have to download the respective information. Thus, it increases the workload of both the members and the verifiers. This yields constructing a group signature scheme based on lattices, which does not increase the workload of the current members, managers, and verifiers when achieving the full dynamicity.

1.1 Our Contribution

This paper presents a fully dynamic group signature scheme from lattices with verifier-local revocation and member registration.

However, gaining full anonymity (which was described in the BMW03 model [4] and which was used in [5]) for VLR scheme is a challenging problem. In case of the full-anonymity game between a challenger and an adversary, the challenger provides all the secret signing keys to the adversary. The previous VLR schemes constructed the revocation token by taking the part of the secret signing key of the relevant member. Thus, if we provide all the secret signing keys to the adversary, he can obtain the tokens of the members and check which member's index is used to generate the challenging signature. In our scheme, we separate the construction of the revocation tokens from the generation of the secret signing keys. Thus, we can provide all the secret signing keys to the adversary as in the full anonymity, without any issue.

Even though the full anonymity does not manage revocation queries, we allow the adversary to request revocation tokens. But, we will not provide challenged members' tokens to the adversary, or we will not generate challenged signatures for the member-indices whose tokens are already revealed by the adversary. This restricted version of the full-anonymity is known as *almost-full anonymity*. We adapt the almost-full anonymity suggested in [28] to cope with registration query to prove the security of our scheme. Thus, we propose a new security notion called *dynamical-almost-full anonymity* which is a restricted version of the full anonymity and extended version of the almost-full anonymity for fully dynamic group signatures with VLR and member registration.

Since the previous VLR schemes like the scheme in [19] have not considered member registration separately, they generated members' keys at the setup stage with the group public key. In our scheme, we separate member registration and allow new members to join the group with their secret keys as in [20]. Thus, in the member registration, we provide a simple method to generate keys for the members by using the group public key. First, we use trapdoors [15] to generate the group public key and the authority keys at the setup phase. Since the group manager needs to know the revoking member's revocation token, we allow new members to generate only their secret signing keys at the joining protocol. The group manager issues the revocation token with the member certification. When a member misbehaved, the group manager can revoke the misbehaved member by adding that member's token to the list called *revocation list* (RL) and updating the verifiers with the latest RL as any VLR scheme. When verifying a signature, verifiers have to check the validity of the signer using the latest RL.

Moreover, we provide an *explicit tracing algorithm* to trace signers. The *implicit tracing algorithm* presented in VLR requires executing Verify for each member until the relevant member is found. Since the time consumption is high in the implicit tracing algorithm, it is not convenient for large groups. Hence, if necessary the tracer can use the explicit tracing algorithm instead of using the implicit tracing algorithm for tracing signers.

2 Preliminaries

2.1 Notations

For any integer $k \geq 1$, we denote the set of integers $\{1, \ldots, k\}$ by $[k]$. We denote matrices by bold upper-case letters such as $\mathbf{A}$, and vectors by bold lower-case letters, such as $\mathbf{x}$. We assume that all vectors are in column form. The concatenation of matrices $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$ is denoted by $[\mathbf{A}|\mathbf{B}] \in \mathbb{R}^{n \times (m+k)}$. The concatenation of vectors $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^k$ is denoted by $(\mathbf{x}\|\mathbf{y}) \in \mathbb{R}^{m+k}$. If S is a finite set, $b \xleftarrow{\$} S$ means that b is chosen uniformly at random from S. If S is a probability distribution $b \xleftarrow{\$} S$ means that b is drawn according to S. The Euclidean norm of $\mathbf{x}$ is denoted by $||\mathbf{x}||$ and the infinity norm is denoted by $||\mathbf{x}||_\infty$. Let χ be a b-bounded distribution over $\mathbb{Z}$ (i.e., samples that output by χ is with norm at most b with overwhelming probability where $b = \sqrt{n}\omega(\log n)$).

2.2 Lattices

Let q be a prime and $\mathbf{B} = [\mathbf{b}_1|\cdots|\mathbf{b}_m] \in \mathbb{Z}_q^{r\times m}$ be linearly independent vectors in $\mathbb{Z}_q^r$. The r-dimensional lattice $\Lambda(\mathbf{B})$ for $\mathbf{B}$ is defined as

$$\Lambda(\mathbf{B}) = \{\mathbf{y} \in \mathbb{Z}^r \mid \mathbf{y} \equiv \mathbf{B}\mathbf{x} \bmod q \text{ for some } \mathbf{x} \in \mathbb{Z}_q^m\},$$

which is the set of all linear combinations of columns of $\mathbf{B}$. The value m is the rank of $\mathbf{B}$.

We consider a discrete Gaussian distribution with respect to a lattice. The Gaussian function centered in a vector $\mathbf{c}$ with parameter $s > 0$ is defined as $\rho_{s,\mathbf{c}}(\mathbf{x}) = e^{-\pi\|(\mathbf{x}-\mathbf{c})/s\|^2}$ and the corresponding probability density function proportional to $\rho_{s,\mathbf{c}}$ is defined as $D_{s,\mathbf{c}}(\mathbf{x}) = \rho_{s,\mathbf{c}}(\mathbf{x})/s^n$ for all $\mathbf{x} \in \mathbb{R}^n$. With respect to a lattice Λ the discrete Gaussian distribution is defined as $D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = D_{s,\mathbf{c}}(\mathbf{x})/D_{s,\mathbf{c}}(\Lambda) = \rho_{s,\mathbf{c}}(\mathbf{x})/\rho_{s,\mathbf{c}}(\Lambda)$ for all $\mathbf{x} \in \Lambda$. Since $\mathbb{Z}^m$ is also a lattice, we can define a discrete Gaussian distribution for $\mathbb{Z}^m$. By $D_{\mathbb{Z}^m,\sigma}$, we denote the discrete Gaussian distribution for $\mathbb{Z}^m$ around the origin with the standard deviation σ.

2.3 Lattice-Related Computational Problems

The security of our scheme relies on the hardness of the two lattice-based problems defined below.

Learning With Errors (LWE)

Definition 1. *Learning With Errors (LWE) [27] is parametrized by integers $n, m \geq 1$, and $q \geq 2$. For a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ and χ, the distribution $\mathrm{A}_{\boldsymbol{s},\chi}$ is obtained by sampling $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random and choosing $\mathrm{e} \leftarrow \chi$, and outputting the pair $(\boldsymbol{a}, \boldsymbol{a}^T \cdot \boldsymbol{s} + \mathrm{e})$.*

There are two LWE problems. They are Search-LWE and Decision-LWE. While Search-LWE is to find the secret $\mathbf{s}$ given LWE samples, Decision-LWE is to distinguish LWE samples and samples chosen according to the uniformly distribution. We use the hardness of Decision-LWE problem.

For a prime power q, $b \geq \sqrt{n}\omega(\log n)$, and distribution χ, solving $LWE_{n,q,\chi}$ problem is at least as hard as solving $SIVP_\gamma$ (*Shortest Independent Vector Problem*), where $\gamma = \tilde{\mathcal{O}}(nq/b)$ [15, 29].

Short Integer Solution ($\boldsymbol{SIS_{n,m,q,\beta}}$) SIS was first discussed in seminal work of Ajtai [2]. SIS problem asks to find a sufficiently short nontrivial integer combination of given uniformly random elements of a certain large finite additive group, which sums to zero [27].

Definition 2. *Short Integer Solution ($SIS_{n,m,q,\beta}$ [27, 29]) is as follows. Given m uniformly random vectors $\boldsymbol{a}_i \in \mathbb{Z}_q^n$, forming the columns of a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, find a nonzero vector $\boldsymbol{x} \in \mathbb{Z}^m$ such that $||\boldsymbol{x}|| \leq \beta$ and $\boldsymbol{A}\boldsymbol{x} = 0 \mod q$.*

Table 1. Parameters of the scheme

Parameter	**Value or Asymptotic bound**
Modulus q	$\omega(n^2 \log n)$
Dimension m	$\geq 2n \log q$
Gaussian parameter σ	$\omega(\sqrt{n \log q \log n})$
Integer norm bound β	$\lceil \sigma \cdot \log m \rceil$
Number of protocol repetitions t	$\omega(\log n)$

For any m, β, and for any $q > \sqrt{n\beta}$, solving $SIS_{n,m,q,\beta}$ problem with non-negligible probability is at least as hard as solving $SIVP_\gamma$ problem, for some $\gamma = \beta \cdot \mathrm{O}(\sqrt{n})$ [15].

2.4 Lattice-Related Algorithms

We use a randomized nearest-plane algorithm SampleD, which is discussed in [15] and [24] in our scheme's construction. The algorithm SampleD samples from a discrete Gaussian $D_{\Lambda,s,\mathbf{c}}$ over any lattice Λ. The version given in [24] is defined below.

– SampleD($\mathbf{R}$, $\mathbf{A}$, $\mathbf{u}$, σ) takes as inputs a vector $\mathbf{u}$ in the image of $\mathbf{A}$, a trapdoor $\mathbf{R}$, and $\sigma = \omega(\sqrt{n \log q \log n})$, and outputs $\mathbf{x} \in \mathbb{Z}^m$ sampled from the distribution $D_{\mathbb{Z}^m,\sigma}$, where $\mathbf{x}$ should satisfy the condition $\mathbf{A} \cdot \mathbf{x} = \mathbf{u} \bmod q$.

Preimage sampleable trapdoor functions (PSTFs) [15] are defined by probabilistic polynomial-time algorithms. We use PSTFs discussed in [1, 15, 24].

– GenTrap(n, m, q) is an efficient randomized algorithm. For any given integers $n \geq 1$, $q \geq 2$, and sufficiently large $m = O(n \log q)$, GenTrap(n, m, q) outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor matrix $\mathbf{R}$. The distribution of the output $\mathbf{A}$ is negl(n)-far from the uniform distribution.

2.5 Other Tools

We denote the security parameter by n, and the maximum number of expected users in a group by $N = 2^\ell$. Depending on the security parameter n, other parameters we used are as in Table 1.

In the construction of our scheme, we use one-time signature scheme $\mathcal{OTS}$ = (OGen, OSign, OVer) [25]. $\mathcal{OTS}$ schemes are based on one-way functions, and they are simpler to implement and are computationally efficient than trapdoor functions. $\mathcal{OTS}$ schemes are digital signature schemes. Since $\mathcal{OTS}$ requires the signer to generate keys for each message to be signed newly, keys formed for each message are unique for the particular messages. OGen is the key generation algorithm, which takes as an input (1^n), and outputs a signing, verification key

pair (**osk**, **ovk**). OSign is the signing algorithm, which uses the key **osk** and a message M as inputs, and outputs a signature Σ. OVer is the verification algorithm, which is a deterministic algorithm that takes as inputs the key **ovk**, the message M, and the signature Σ to validate the signature Σ. Depending on the validation of the signature, OVer outputs $\top$ or $\bot$ [14].

3 Achieving security for VLR schemes with registration

This section first describes the security requirements of group signatures. Then we discuss VLR group signature schemes. Later, we explain the difficulties of achieving full-anonymity for VLR group signature schemes and put forward a new security notion called *dynamical-almost-full anonymity* that manages anonymity of VLR group signature schemes with member registration.

3.1 Security requirements

The group signatures, which were introduced by Chaum and van Heyst [13] provided two main features called *anonymity* and *traceability*. *Anonymity* requires any adversary is not able to discover the signer. *Traceability* requires no one can create a signature that cannot be traced by the group manager. Nevertheless, in last decades, more security requirements have been presented. Bellare et al. [4] (BMW03 model) provided two appropriate security notions called *full-anonymity* and *full-traceability* that formalize the previous security requirements.

Full-anonymity requires no adversary can identify the signer same as in the anonymity suggested in [13]. But, in the full-anonymity, the adversary is stronger since he may corrupt all the members including the one issuing the signature. Moreover, he can view the outcome that group manager sees when tracing a signer.

Full-traceability also much stronger than the traceability in the past and it also acts as a strong version of collusion-resistance. Thus, a group of colluding group members who pool their secret keys cannot create a signature that belongs to none of them; even they know the secret key of the group manager. Thus, in the full traceability game, the adversary can query signatures for any message and index, and he can corrupt any member.

In the scheme in [5], non-frameability is separated from the traceability. In *traceability* game, the tracing manager's secret key is provided to the adversary but, the group manager's secret key is not provided. Thus, the adversary cannot create untraceable dummy members. But in *non-frameability*, the group manager's and the tracing manager's secret keys are given to the adversary.

3.2 VLR group signatures

The functionality of member revocation is a desirable requirement of any group signature since misbehaved members should be removed from the group and restricted them signing on behalf of the group. The simplest method is generating

all the keys newly including public keys and secret keys (except for revoking member) when a member is revoked and broadcasting the new keys to existing members and verifiers. But this method is not appropriate for large groups. Bresson and Stern [8] suggested an approach that requires signers to prove that his member certification is not in the public revocation list at the time of signing. Since the signature size increases with the number of revoked members in this method, it is also not suitable for large groups. Later Camenisch et al. [11] proposed an approach using dynamic accumulators, where the *accumulator* is an algorithm that allows hashing a large set of inputs to one shorter value, and the *dynamic accumulator* allows to add or delete inputs dynamically. However, this approach requires members to keep track of revoked user information, and needs to update their membership. Thus in this approach, workloads of the current group members increase. A different revocation method called *Verifier-local Revocation (VLR)* was proposed by Brickell [9] and formalized by Boneh et al. [6] in their group signature scheme. Other than the scheme in [6], schemes like [21, 28] use VLR to manage member revocation.

Verifier-local Revocation (VLR) uses a token system to manage the status of the members. Each member has a revocation token other than their secret signing keys. When a member is revoked, his revocation token is added to a list called *Revocation list* (*RL*) and passed to the verifiers. Thus, the verifiers can check the validity of the signer using the latest *RL*. Since VLR does not require to generate keys newly or keep track of information for the existing members, it is more convenient than any other approach. It simply asks to update the verifiers who are less than the members in number when a member is revoked. Thus, it is suitable for any size of groups.

In general, group signature schemes consist of four algorithms, KeyGen, Sign, Verify, and Open. VLR group signature schemes consist of former three algorithms, and VLR scheme has an *implicit tracing algorithm* for tracing signers instead of Open.

- KeyGen(n,N): This randomized PPT algorithm takes as inputs n and N. Then it outputs a group public key $\mathbf{gpk}$, a vector of user secret keys $\mathbf{gsk} = (\mathbf{gsk}[0], \mathbf{gsk}[1], ..., \mathbf{gsk}[N-1])$, and a vector of user revocation tokens $\mathbf{grt} = (\mathbf{grt}[0], \mathbf{grt}[1], \ldots, \mathbf{grt}[N-1])$, where $\mathbf{gsk}[i]$ is the i-th user's secret key and $\mathbf{grt}[i]$ is his revocation token.
- Sign($\mathbf{gpk}$, $\mathbf{gsk}[d]$, M): This randomized algorithm takes as inputs the group public key $\mathbf{gpk}$, a secret signing key $\mathbf{gsk}[d]$, and a message $M \in \{0,1\}^*$. Sign generates a group signature Σ on M.
- Verify($\mathbf{gpk}$, RL, Σ, M): This deterministic algorithm verifies whether the given signature Σ is a valid signature on given message M using the given group public key $\mathbf{gpk}$. Moreover, Verify validates the signer is not being revoked using RL.

Implicit Tracing Algorithm: Any VLR group signature scheme has an *implicit tracing algorithm*. The implicit tracing algorithm uses $\mathbf{grt}$ as the tracing key. For a given valid message-signature pair (M, Σ), an authorized person can run

$\mathsf{Verify}(\mathbf{gpk}, RL{=}\mathbf{grt}[i], \Sigma, M)$ for $i = 0, 1, \cdots, N-1$ until Verify returns *invalid*. The index of the signer is the first index $i^* \in \{0, 1, \cdots, N-1\}$ that Verify returns invalid. The implicit tracing algorithm fails if Verify verifies properly for all users on the given signature. Since the implicit tracing algorithm requires to run Verify linear times in N, it is inappropriate for large groups. In comparison to the algorithm Open, its time consumption is high.

Though VLR is the comparably convenient approach for any group signatures, the existing lattice-based group signature schemes with VLR such as [19] relies on a weaker security notion called *selfless-anonymity*. Not like the full-anonymity, the selfless-anonymity has some limitations. According to the BMW03 model, in the full-anonymity game between a challenger and an adversary, all the secret keys of the group members including challenging keys are given to the adversary at the beginning of the game. But, in the selfless-anonymity game, the adversary is not given any secret keys. He can query secret keys at the query phase but not related to the challenging indices. However, the adversary is allowed for the queries; *Signing*, *Corruption*, and *Revocation*.

3.3 Achieving stronger security for VLR schemes with member registration

Our scheme is for managing both member registration and revocation. Thus, the almost-full anonymity suggested in [28] is not sufficient for our scheme. We modify the almost-full anonymity by adding the registration query. Moreover, we add some restrictions to manage the attacks of the adversary. We concern how to secure our scheme (i) when the adversary joins the group as a legal user before the game starts and (ii) when he requests to join the group after the game begins. When any user joins the group, we provide revocation tokens to them. So the adversary can get the revocation tokens by adding new users. In concern (i), the adversary is getting the revocation tokens when joining the group before the game starts (he is a legal user) and he can use those indices in challenge phase after the game starts. Since the adversary has not queried those revocation tokens, this is not tracked by the almost-full anonymity. As a solution for the above concerns, we suggest a new security notion called *dynamical-almost-full anonymity*, which is an extended version of the almost-full anonymity for fully dynamic group signatures with VLR and member registration.

In the dynamical-almost-full anonymity, we allow the adversary to add new members to the group at the anonymity game as same as in previous group signature schemes like [5, 23] and we maintain a global list called **RU**. **RU** is used to track the details of the new members that the adversary adds via the registration query. **RU** only consists of indices of the members that the adversary added. Tracking the new user details is also done in the previous group signature schemes like in [5, 23]. However, we will not provide the revocation tokens of the new users at the registration query, but the adversary can request revocation tokens using the revocation query. At the challenge stage, we check **RU** and only generate the challenging signature for the indices in **RU**, but those are not used for the revocation queries. By creating challenging signature only for the

indices in **RU** we give a solution to the concern (i) and not providing the revocation tokens for the members at the registration query we give a solution to the problem (ii). The dynamical-almost-full anonymity game between a challenger and an adversary is as follows.

- **Initial Phase:** The challenger C runs KeyGen to get a group public key **gpk**, authority secret keys (**ik**,**ok**). Then gives **gpk** and existing group members' secret signing keys **gsk** to the adversary A, and creates a new list **RU**.
- **Query Phase:** A can query any token (**grt**) of any user and can access the opening oracle, which results with $\mathsf{Open}(\mathbf{ok}, M, \Sigma)$. Moreover, A can add new users to the group using registration query. If the new user is valid and not already in the registration table *reg*, then C adds new user to the group. Then C generates token for the new user and updates both *reg* and **RU**. However, C does not return the token of the new user to A.
- **Challenge Phase:** A outputs a message M^* and two distinct identities i_0, i_1. If A already not queried the revocation tokens of i_0, i_1 and if i_0, i_1 are in **RU**, then C selects a bit $b \xleftarrow{\$} \{0,1\}$, generates $\Sigma^* = \mathsf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i_b], M^*)$ and sends Σ^* to A. A still can query the opening oracle except the signature challenged and he is not allowed for revocation queries with challenging indices. A can add users to the group.
- **Guessing Phase:** Finally, A outputs a bit b', the guess of b. If $b' = b$, then A wins.

4 Our scheme

In our scheme, there are two authorities, group manager and, tracing manager. The group manager interacts with new users who want to become group members and issues membership-certifications to the valid users. Moreover, he manages the member revocation. The tracing manager discovers the signers. Each manager has their public and privates keys. We assume the group manager and new users interact through a secure channel. The users generate their secret signing keys, and they can sign messages once the group manager accepted them as group members. The group manager creates group members' tokens. To track the details of the members, we maintain a registration table *reg*.

Every new user has to interact with the group manager by presenting a valid signature. The new user i, who has a personal public and private key pair $(upk[i], usk[i])$ (as in [5]), samples a short vector $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^{4m},\sigma}$ and computes $\mathbf{z}_i$ using $\mathbf{x}_i$ and $\mathbf{F}$, where $\mathbf{F}$ is a public parameter. Then the new user generates a signature Σ_{join} by signing $\mathbf{z}_i$ with his personal private key $usk[i]$. When the group manager receives the massage-signature pair $(\mathbf{z}_i, \Sigma_{join})$, first he checks whether $\mathbf{z}_i$ is used before. If $\mathbf{z}_i$ is not used before, then the group manager verifies Σ_{join} on $\mathbf{z}_i$ using the user's personal public key $upk[i]$. Then he samples the new user's revocation token and updates the registration table *reg* with the new user's details. Finally, the group manager sends revocation token to the user. Now the new user (group member) can sign messages on behalf of the group.

4.1 Supporting zero-knowledge protocol

This section provides a general description of zero-knowledge argument system that we use in our scheme. Many other lattice-based schemes like [19, 20, 22] also use ZKAoK to prove the verifier that the signer is valid in zero-knowledge.

Let COM be the statistically hiding and computationally binding commitment scheme described in [17]. We use matrices $\mathbf{F}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{V}$, $\mathbf{G}$, $\mathbf{H}$ and vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{c}_1$, $\mathbf{c}_2$ as public parameters. The prover's witness consists of vectors $\mathbf{x}$, $\mathsf{bin}(\mathbf{z})$, $\mathbf{r}$, $\mathbf{s}$, $\mathbf{e}_1$, and $\mathbf{e}_2$. The prover's goal is to convince the verifier that $\mathbf{F} \cdot \mathbf{x} = \mathbf{H}_{4n \times 2m} \cdot \mathsf{bin}(\mathbf{z})$ (as discussed in [20]), $\mathbf{V} \cdot (\mathbf{A} \cdot \mathbf{r}) + \mathbf{e}_1 = \mathbf{v} \mod q$ (as discussed in [19]), and $(\mathbf{c}_1 = \mathbf{B}^T\mathbf{s} + \mathbf{e}_1, \mathbf{c}_2 = \mathbf{G}^T\mathbf{s} + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathsf{bin}(\mathbf{z}_i))$ (as discussed in [22]). Here $\mathbf{H}_{n \times n\lceil \log q \rceil} \in \mathbb{Z}^{n \times n\lceil \log q \rceil}$ is a "power-of-2" matrix and $\mathbf{z} = \mathbf{H}_{n \times n\lceil \log q \rceil} \cdot \mathsf{bin}(\mathbf{z})$ for any $\mathbf{z} \in \mathbb{Z}_q^n$.

4.2 Description of Our Scheme

This section describes the algorithms of our scheme. Our scheme consists of six algorithms namely, KeyGen, Join, Sign, Verify, Open, and Revoke. The former five algorithms follow the techniques given in [20]. We adapt algorithms presented in [20] to compatible with member revocation mechanism. We use algorithm Revoke to manage member revocation.

Setup: The randomized algorithm $\mathsf{KeyGen}(1^n, 1^N)$ works as follows.

1. Run PPT algorithm $\mathsf{GenTrap}(n, m, q)$ to get $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor $\mathbf{T_A}$.
2. Sample vector $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^n$.
3. Generate encryption and decryption keys by running $\mathsf{GenTrap}(n, m, q)$ to get $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a trapdoor $\mathbf{T_B}$.
4. Sample matrix $\mathbf{F} \xleftarrow{\$} \mathbb{Z}_q^{4n \times 4m}$.
5. Finally output, the group public key $\mathbf{gpk}$:= ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{F}$, $\mathbf{u}$), the group manager's (issuer's) secret key $\mathbf{ik}$:= $\mathbf{T_A}$ and the opener's secret key $\mathbf{ok}$:= $\mathbf{T_B}$.

Join: A new user i, who has a personal public key and private key pair $(\mathbf{upk}[i], \mathbf{usk}[i])$ can interact with the group manager (issuer) to join the group as follow.

1. User i samples a discrete Gaussian vector $\mathbf{x}_i \leftarrow D_{\mathbb{Z}^{4m},\sigma}$, and computes $\mathbf{z}_i \leftarrow \mathbf{F} \cdot \mathbf{x}_i \in \mathbb{Z}_q^{4n}$. Then he generates a signature $\Sigma_{join} \leftarrow \mathsf{Sig}(\mathbf{usk}[i], \mathbf{z}_i)$ and sends both $\mathbf{z}_i$, and Σ_{join} to the group manager.
2. The group manager GM verifies that $\mathbf{z}_i$ was not used by any user previously, by checking the registration table *reg*. Then he verifies Σ_{join} is a valid signature on $\mathbf{z}_i$, using $\mathsf{Vf}(\mathbf{upk}[i], \mathbf{z}_i, \Sigma_{join})$. He aborts if any condition fails. Otherwise he will sign the user's index $d = \mathsf{bin}(\mathbf{z}_i)$, the binary representation of $\mathbf{z}_i$, using group manager's private key and generates the certificate for the index $\mathit{cert\text{-}index}_i = \mathsf{Sign}(\mathbf{ik}, \mathsf{bin}(\mathbf{z}_i))$.

The group manager selects $\mathbf{R}_i \xleftarrow{\$} \mathbb{Z}_q^{n\times 4n}$ and computes $\mathbf{w}_i = \mathbf{R}_i \cdot \mathbf{z}_i$. Then he samples a vector $\mathbf{r}_i \in \mathbb{Z}^m \leftarrow \mathsf{SampleD}(\mathbf{T_A}, \mathbf{A}, \mathbf{u} - \mathbf{w}_i, \sigma)$, and generates the certificate for the token $\textit{cert-token}_i = \mathsf{Sign}(\mathbf{ik}, (\mathbf{A} \cdot \mathbf{r}_i))$ $(\mathbf{ik} = \mathbf{T_A})$.
Then he saves the details of the new member (user) i in the registration table $reg[i] \leftarrow (i, d, \mathbf{upk}[i], \mathbf{z}_i, \Sigma_{join}, \mathbf{R}_i, \mathbf{w}_i, \mathbf{r}_i, 1)$ and makes the record active (1). Finally, GM sends the $\textit{cert}_i = (\textit{cert-index}_i, \textit{cert-token}_i, \mathbf{R}_i, (\mathbf{A} \cdot \mathbf{r}_i))$ as the new member's member-certificate.

Sign: $\mathsf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i], \textit{cert}_i, M)$ is a randomized algorithm, that generates a signature Σ on a given message M using $\mathbf{gsk}[i] = \mathbf{x}_i$ as follows.

1. Let $\mathcal{H}_1$: $\{0,1\}^* \to \mathbb{Z}_q^{n\times \ell}$, $\mathcal{H}_2$: $\{0,1\}^* \to \{1,2,3\}^t$ and $\mathcal{G}$: $\{0,1\}^* \to \mathbb{Z}_q^{n\times m}$ be hash functions, modeled as a random oracle.
2. Parse **gpk** as (**A, B, F, u**) and $\textit{cert}_i$ as $(\textit{cert-index}_i, \textit{cert-token}_i, \mathbf{R}_i, (\mathbf{A}\cdot\mathbf{r}_i))$.
3. Run $\mathsf{OGen}(1^n) \to (\mathbf{ovk}, \mathbf{osk})$.
4. Encrypt the index $d = \mathsf{bin}(\mathbf{z}_i)$, where $\mathbf{z}_i = \mathbf{F} \cdot \mathbf{x}_i$.
   (a) Let $\mathbf{G} = \mathcal{H}_1(\mathbf{ovk}) \in \mathbb{Z}_q^{n\times 2m}$.
   (b) Sample $\mathbf{s} \leftarrow \chi^n$, $\mathbf{e}_1 \leftarrow \chi^m$ and $\mathbf{e}_2 \leftarrow \chi^\ell$.
   (c) Compute the ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ pair
   $(\mathbf{c}_1 = \mathbf{B}^T\mathbf{s} + \mathbf{e}_1,\ \mathbf{c}_2 = \mathbf{G}^T\mathbf{s} + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathsf{bin}(\mathbf{z}_i))$.
5. Sample $\rho \xleftarrow{\$} \{0,1\}^n$, let $\mathbf{V} = \mathcal{G}(\mathbf{A}, \mathbf{u}, M, \rho) \in \mathbb{Z}_q^{n\times m}$.
6. Compute $\mathbf{v} = \mathbf{V} \cdot (\mathbf{A} \cdot \mathbf{r}_i) + \mathbf{e}_1 \mod q$ ($||\mathbf{e}_1||_\infty \leq \beta$ with overwhelming probability).
7. Execute $\mathsf{Verify}(\mathbf{A}, \mathsf{bin}(\mathbf{z}_i), \textit{cert-index}_i)$ to prove $\textit{cert-index}_i$ is generated on $\mathsf{bin}(\mathbf{z}_i)$ and $\mathsf{Verify}(\mathbf{A}, (\mathbf{A} \cdot \mathbf{r}_i), \textit{cert-token}_i)$ to prove $\textit{cert-token}_i$ is generated on $(\mathbf{A} \cdot \mathbf{r}_i)$. Then generate a proof as in Section 4.1, that the user is valid, honestly computed above $\mathbf{v}$, and index is correctly encrypted. By repeating the basic protocol of KTX commitment scheme in Section 4.1 $t = \omega(\log n)$ times to make the soundness error negligible. Then we make it non-interactive using the Fiat-Shamir heuristic as a triple, $\Pi = (\{CMT^{(k)}\}_{k=1}^t, CH, \{RSP^{(k)}\}_{k=1}^t)$, where CH $= (\{Ch^{(k)}\}_{k=1}^t) = \mathcal{H}_2(M, \{CMT^{(k)}\}_{k=1}^t, \mathbf{c}_1, \mathbf{c}_2)$.
8. Compute $\mathcal{OTS}$; $sig = \mathsf{OSig}(\mathbf{osk}, (\mathbf{c}_1, \mathbf{c}_2, \Pi))$.
9. Output signature $\Sigma = (\mathbf{ovk}, (\mathbf{c}_1, \mathbf{c}_2), \rho, \Pi, sig, \mathbf{v})$.

Verify: The deterministic algorithm $\mathsf{Verify}(\mathbf{gpk}, M, \Sigma, RL)$ works as follows, where $RL = \{\{\mathbf{u}_i\}_i\}$.

1. Parse the signature Σ as $(\mathbf{ovk}, (\mathbf{c}_1, \mathbf{c}_2), \rho, \Pi, sig, \mathbf{v})$.
2. Get $\mathbf{V} = \mathcal{G}(\mathbf{A}, \mathbf{u}, M, \rho) \in \mathbb{Z}_q^{n\times m}$.
3. If $\mathsf{OVer}(\mathbf{ovk}, ((\mathbf{c}_1, \mathbf{c}_2), \Pi), sig) = 0$ then return 0.
4. Parse Π as $(\{CMT^{(k)}\}_{k=1}^t, \{Ch^{(k)}\}_{k=1}^t, \{RSP^{(k)}\}_{k=1}^t)$.
5. If $(Ch^{(1)}, \ldots, Ch^{(t)}) \neq \mathcal{H}_2(M, \{CMT^{(k)}\}_{k=1}^t, \mathbf{c}_1, \mathbf{c}_2)$ return 0 else proceed.
6. For $k = 1$ to t run the verification steps of the commitment scheme to validate $RSP^{(k)}$ with respect to $CMT^{(k)}$ and $Ch^{(k)}$. If any of the conditions fails then output invalid.

7. For each $\mathbf{u}_i \in RL$ compute $\mathbf{e}_{i}^{'} = \mathbf{v} - \mathbf{V} \cdot \mathbf{u}_i \mod q$ to check whether there exists an index i such that $||\mathbf{e}_{i}^{'}||_\infty \leq \beta$. If so return invalid.
8. Return valid.

Open: $\mathsf{Open}(\mathbf{gpk}, \mathbf{ok}, reg, M, \Sigma)$ functions as follows, where $\mathbf{ok} = \mathbf{T_B}$.

1. Let $\mathbf{G} = \mathcal{H}_1(\mathbf{ovk})$.
2. Then using $\mathbf{T_B}$ compute a small norm matrix $\mathbf{Y} \in \mathbb{Z}^{m \times 2m}$, where $\mathbf{B} \cdot \mathbf{Y} = \mathbf{G} \mod q$.
3. Compute $\mathsf{bin}(\mathbf{z}_i) = \lfloor (\mathbf{c}_2 - \mathbf{Y}^T \cdot \mathbf{c}_1)/(q/2) \rceil$, determine the signer using *reg* corresponds to $\mathsf{bin}(\mathbf{z}_i)$, and output the index.

Revoke: The algorithm $\mathsf{Revoke}(\mathbf{gpk}, \mathbf{ik}, i, reg, \mathrm{RL})$ functions as follows.

1. Query *reg* for i and obtain revoking member's revocation token $(\mathbf{A} \cdot \mathbf{r}_i)$.
2. Add $(\mathbf{A} \cdot \mathbf{r}_i)$ to RL and update $reg[i]$ to inactive (0).
3. Return RL.

5 Correctness and Security Analysis of the Scheme

To define correctness and the security requirements we use a set of experiments consisted of a set of oracles which can be executed by the adversary. We maintain a set of global lists, which are used by the oracles and performed by the challenger C. When the adversary A adds a new user to the group (registration query), and if the new user is excepted as a new member, his index is added to a list called **RU**. When A corrupts any user, then that user's index is added to **CUL**. **SL** contains the signatures that obtained from Sign oracle. When A requests a signature, the generated signature is added to **SL** with the index and the message. When A accesses $\mathsf{Challenge}$ oracle, the generated signature is added to **CL** with the message sent. When A reveals any user-revocation token, the challenger adds that user index to **TU**. When A reveals any user-secret signing key then that user index is added to **BU**. We use a set S to maintain a set of revoked users.

The oracles that we use in the experiments are as follows.

- $\mathsf{AddU}(i)$: The adversary A can add a user $i \in \mathbb{N}$ to the group as an honest user. The oracle adds i to **RU**. But the new user's revocation token is not returned to the adversary.
- $\mathsf{CrptU}(i, upk)$: A can corrupt user i by setting its personal public key $\mathbf{upk}[i]$ to *upk*. The oracle adds i to **CUL**, and initializes the issuer's state in *group-joining protocol*.
- $\mathsf{SendToGM}(i, M_{in})$: A corrupts user i, and engages in *group-joining protocol* with Issue-executing issuer. The adversary provides i and M_{in} to the oracle. The oracle which maintains the Issue state, returns the outgoing message, and adds a record to *reg*.

- $\mathsf{SendToUser}(i,\ M_{in})$: A corrupts the issuer and engages in *group-joining protocol* with Join-executing user. The adversary passes i and M_{in} to the oracle. The oracle which maintains the user i state, returns the outgoing message, and sets the private signing key of i to the final state of Join.
- $\mathsf{RevealSk}(i)$: A can retrieve the secret signing key of the user i. The oracle updates $\mathbf{BU}$ and returns $\mathbf{gsk}[i]$.
- $\mathsf{RevealRt}(i)$: A can retrieve the revocation token of user i, and the oracle returns revocation token and adds i to $\mathbf{TU}$.
- $\mathsf{ReadReg}(i)$: A reads the information of i in *reg*.
- $\mathsf{ModifyReg}(i,\ val)$: A modifies $reg[i]$ by setting *val*.
- $\mathsf{Sign}(i,\ M)$: A obtains a signature Σ for a given message M and user i who is an honest user and has private signing key.
- $\mathsf{Chal}_b(i_0, i_1, M)$: This oracle is for defining anonymity and provides a group signature for the given message M under the private signing key of i_b, as long as both i_0, i_1 are active and honest users having private signing keys (in $\mathbf{RU}$). Moreover, those indices should not being used to reveal revocation tokens (not in $\mathbf{BU}$).
- $\mathsf{Revoke}(i)$: A can request to revoke user i. The oracle updates the record to 0 for i in *reg* and adds revocation token of i to the set S.
- $\mathsf{Open}(M,\ \Sigma)$: A can access the *opening* oracle with a message M and a signature Σ to obtain the identity of the user, who generated the signature Σ. If Σ is generated at Chal_b, then oracle will abort.

In addition, we use the following simple polynomial-time algorithm for ease.

- $\mathsf{IsActive}(i,reg)$: This algorithm determines whether the member i is active by querying the registration table *reg* and outputs either 0 or 1.

5.1 Correctness

$\underline{\mathbf{Exp}^{corr}_{FDGS,A}(\lambda)}$
$(\mathbf{gpk}, \mathbf{ok}, \mathbf{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathbf{RU} \leftarrow \emptyset$;
$(i, M) \leftarrow A(\mathbf{gpk}; \mathsf{AddU}, \mathsf{ReadReg}, \mathsf{Revoke}, \mathsf{RevealRt})$;
If $i \notin \mathbf{RU}$ or $\mathbf{gsk}[i] = \varepsilon$ or $cert_i = \varepsilon$ or $\mathsf{IsActive}(i, reg) = 0$ then return 0.
$\Sigma \leftarrow \mathsf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i], cert_i, M)$;
If $\mathsf{Verify}(\mathbf{gpk},\ M,\ \Sigma,\ S) = 0$ then return 1.
$(i') \leftarrow \mathsf{Open}(\mathbf{gpk}, \mathbf{ok}, reg, M, \Sigma)$;
If $i \neq i'$ then return 1.
Return 0.

First note Verify accepts signatures, which are only generated by active and honest users. If signer's revocation token is in RL, then his signature is not accepted. Steps 6 and 7 in both Sign and Verify guarantee this condition. Completeness of the underlying argument system guarantees that the valid signatures are always accepted and soundness of the underlying argument system guarantees that a revoked signer cannot pass the test. Open outputs the index of the signer with overwhelming probability. It computes $\mathsf{bin}(\mathbf{z}_i)$ and extracts the details of the signer from *reg*.

5.2 Anonymity

Theorem 1: *In the random oracle model, our scheme is dynamical-almost-full anonymous if $LWE_{n,q,\chi}$ problem is hard to solve.*

$\underline{\mathbf{Exp}^{anon\text{-}b}_{FDGS,A}(\lambda)}$

$(\mathbf{gpk}, \mathbf{ok}, \mathbf{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathbf{RU}, \mathbf{CUL}, \mathbf{SL}, \mathbf{CL}, \mathbf{BU}, \mathbf{TU} \leftarrow \emptyset$;

$b^* \leftarrow A(\mathbf{gpk}, \mathbf{gsk}$;

$\mathsf{AddU}, \mathsf{CrptU}, \mathsf{SendToUser}, \mathsf{RevealSk}, \mathsf{RevealRt}, \mathsf{Open}, \mathsf{ModifyReg}, \mathsf{Revoke}, \mathsf{Chal}_b)$;

Return b^*;

We prove that our scheme is dynamical-almost-full anonymous via a sequence of games.

Game 0: This is the above-defined experiment. The challenger C runs $\mathsf{KeyGen}(1^n, 1^N)$ to obtain group public keys and authority keys. Next, C gives the group public key $\mathbf{gpk}$ and all the existing group members' secret keys $\mathbf{gsk}$ to the adversary A. In the query phase, A can request for revocation tokens of any member, and A can access opening for any signature. Moreover, A can add new members to the group. C validates the new members and adds records to the registration table *reg* and $\mathbf{RU}$. But C will not provide the revocation tokens of the new members without a request of A. Thus, the member certification will not be provided at the registration query and the challenger returns a success message only. When A corrupts the users, those users' indices are added to $\mathbf{CUL}$ and when he revokes users, those users' indices are added to S with tokens of them. Moreover, when A reveals any user token, C adds those users' indices to $\mathbf{TU}$ and returns the member certificate *cert*. In the challenge phase, A sends two indices (i_0, i_1) with a message M^*. If (i_0, i_1) are newly added as per $\mathbf{RU}$ and are not used for querying revocation tokens (not in $\mathbf{TU}$), then C generates and sends back a signature $\Sigma^* = (\mathbf{ovk}^*, (\mathbf{c}_1^*, \mathbf{c}_2^*), \rho^*, \Pi^*, sig^*, \mathbf{v}^*)$ $\leftarrow \mathsf{Sign}(\mathbf{gpk}, \mathbf{gsk}[i_b]^*, cert_{i_b}, M^*)$ for a random $b \leftarrow \{0,1\}$. A returns $b' \in \{0,1\}$ the guess of b. If $b' = b$ then returns 1 or 0 otherwise.

The following games are same as Game 0 with slight modifications. Thus, still A can access the oracles, and C maintains the global lists according to A's requests through the oracles. In any game, A's requests are almost the same up to some slight changes in inputs. Thus, C manages those queries as following games explained and updates the global lists according to A's requests.

Game 1: In this game, the challenger C makes a slight modification comparing to **Game** 0. C generates the one-time key pair $(\mathbf{ovk}^*, \mathbf{osk}^*)$ at the beginning of the game. If A accesses the opening oracle with a valid signature $\Sigma = (\mathbf{ovk}, (\mathbf{c}_1, \mathbf{c}_2), \rho, \Pi, sig, \mathbf{v})$, where $\mathbf{ovk}$=$\mathbf{ovk}^*$, C returns a random bit and aborts. However, $\mathbf{ovk}$=$\mathbf{ovk}^*$ contradicts the strong unforgeability of $\mathcal{OTS}$. Moreover, since the $\mathbf{ovk}^*$ is independent of the adversary's view, probability of $\mathbf{ovk}$=$\mathbf{ovk}^*$ is negligible. Besides, if A comes up with a valid signature Σ, where $\mathbf{ovk}$=$\mathbf{ovk}^*$, then *sig* is a forged signature. We assume that A does not request for opening of a valid signature with $\mathbf{ovk}^*$.

Game 2: In this game, C programs the random oracle $\mathcal{H}_1$. At the beginning of the game, C replaces the encrypting matrices $\mathbf{B}$ and $\mathbf{G}$. C chooses uniformly random $\mathbf{B}^* \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{G}^* \in \mathbb{Z}_q^{n \times \ell}$. Then sets $\mathcal{H}_1(\mathbf{ovk}^*)$=$\mathbf{G}^*$. To answer

the opening oracle requests with $\Sigma = (\mathbf{ovk}, (\mathbf{c}_1, \mathbf{c}_2), \rho, \Pi, sig, \mathbf{v})$, C samples $\mathbf{Y} \leftarrow (D_{z^m,\sigma})^\ell$, and computes $\mathbf{G} = \mathbf{B}^*\mathbf{Y} \in \mathbb{Z}_q^{n\times\ell}$. This $\mathbf{G}$ is used to answer the opening and keep track of ($\mathbf{ovk}$, $\mathbf{Y}$, $\mathbf{G}$) to be reused if A repeats the same requests for $\mathcal{H}_1(\mathbf{ovk})$. The distributions of $\mathbf{G}$ is statistically close to the uniform over $\mathbb{Z}_q^{n\times\ell}$ [15]. Thus, this game is indistinguishable from Game 1.

Game 3: In this game, instead of honestly generating the legitimate non-interactive proof Π, the challenger C simulates the proof without using the witness. This is done by invoking the simulator for each $k \in [t]$ and then program the random oracle $\mathcal{H}_1$ accordingly. The challenged signature Σ^* is statistically close to the signature in the previous games since the argument system is statistically zero-knowledge. Thus, Game 3 is indistinguishable from Game 2.

Game 4: In this game, the challenger C replaces the original revocation token. We have $\mathbf{v} = \mathbf{V} \cdot \mathbf{grt}[i_b] + \mathbf{e}_1 \mod q$. C samples $\mathbf{t} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly and computes $\mathbf{v} = \mathbf{V}\cdot\mathbf{t}+\mathbf{e}_1 \mod q$. $\mathbf{V}$ is uniformly random over $\mathbb{Z}_q^{m\times n}$, $\mathbf{e}_1$ is sampled from the error distribution χ, and C replaces only $\mathbf{grt}[i_b]$ with $\mathbf{t}$. The rest of the game is same as Game 3. Thus, the two games are statistically indistinguishable.

Game 5: In this game, the challenger C obtains $\mathbf{v}$ uniformly. Thus, C makes details of revocation token totally independent of the bit b. C samples $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^m$ and sets $\mathbf{v} = \mathbf{y}$. In the previous game, the pair ($\mathbf{V}$, $\mathbf{v}$) is a proper $LWE_{n,q,\chi}$ instance and in this game C replaces $\mathbf{v}$ with truly uniformly sampled $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^m$. Under the assumption of $LWE_{n,q,\chi}$ problem is hard, Game 4 and Game 5 are indistinguishable.

Game 6: In this game the challenger C modifies the generation of ciphertext $(\mathbf{c}_1^*, \mathbf{c}_2^*)$ uniformly. Let $\mathbf{c}_1^* = x_1$ and $\mathbf{c}_2^* = x_2 + \lfloor q/2 \rfloor d_b$, where $x_1 \in \mathbb{Z}^m$ and $x_2 \in \mathbb{Z}^\ell$ are uniformly random and d_b is the index of the challenger's bit. The rest of the game is same as Game 5. Game 5 and Game 6 are indistinguishable under the assumption of the hardness of $LWE_{n,q,\chi}$.

Game 7: Finally, we make Σ^* totally independent of the bit b. The challenger C samples $x_1' \in \mathbb{Z}_q^m$ and $x_2' \in \mathbb{Z}_q^\ell$ uniformly random and assigns $\mathbf{c}_1^* = x_1'$ and $\mathbf{c}_2^* = x_2'$. Thus, Game 6 and Game 7 are statistically indistinguishable. Since Game 7 is totally independent from the challenger's bit b, the advantage of the adversary in this game is 0.

Hence, these games prove that proposed scheme is secure with the dynamical-almost-full anonymity.

5.3 Traceability

Theorem 2: *In the random oracle model, our scheme is traceable if* ***SIS*** *problem is hard.*

$\underline{\mathbf{Exp}^{trace}_{FDGS,A}(\lambda)}$
$(\mathbf{gpk}, \mathbf{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathbf{RU}, \mathbf{CUL}, \mathbf{SL}, \mathbf{BU}, \mathbf{TU} \leftarrow \emptyset$;
$(M, \Sigma) \leftarrow A(\mathbf{gpk}, \mathbf{ok}$;
$\mathsf{AddU}, \mathsf{CrptU}, \mathsf{SendToIssuer}, \mathsf{RevealSk}, \mathsf{RevealRt}, \mathsf{Sign}, \mathsf{Revoke})$;
If $\mathsf{Verify}(\mathbf{gpk}, M, \Sigma, S) = 0$ then return 0.

$i \leftarrow \mathsf{Open}(\mathbf{gpk}, \mathbf{ok}, reg, M, \Sigma)$;
If $i = 0$ or $\mathsf{IsActive}(i, reg) = 0$ then return 1 else return 0.

Suppose there is an algorithm B that solves SIS problem with non-negligible probability. The adversary A who has **gpk** and **ok** outputs (M, Σ) in the traceability game. He can add new users and replace members' personal public keys. Moreover, he can query for secret signing keys and revocation tokens of any member. For the queries of A, B answers as in [22] and [20] by using oracles.

Finally, A outputs forgery signature $\Sigma^*=(\mathbf{ovk}^*, (\mathbf{c}_1^*, \mathbf{c}_2^*), \rho^*, \Pi^*, sig^*, \mathbf{v}^*)$ on message M^*. B opens Σ^* and obtains the index. As same as in [22] and [20], the improved Forking Lemma [10] guarantees that, with probability at least $1/2$, B can obtain 3-fork involving tuple $(M, \{CMT^{(k)}\}_{k=1}^{t}, \mathbf{c}_1, \mathbf{c}_2)$ running A up to $32 \cdot Q_H/(\varepsilon - 3^{-t})$ times with the same tape. Rest of the proof flows as in [22] and [20] and finally we can say, if A has non-negligible success probability and runs in polynomial time, then so does B. This concludes our proof of traceability.

5.4 Non-frameability

Theorem 3: *In the random oracle model, our scheme is non-frameable if **SIS** problem is hard.*

We use the proof discussed in [20] to prove our scheme's non-frameability.

$\underline{\mathbf{Exp}^{non\text{-}fram}_{FDGS,A}(\lambda)}$
$(\mathbf{gpk}, \mathbf{ok}, \mathbf{ik}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathbf{RU}, \mathbf{CUL}, \mathbf{SL}, \mathbf{BUTU} \leftarrow \emptyset$;
$(M, \Sigma, i) \leftarrow A(\mathbf{gpk}, \mathbf{ik}, \mathbf{ok}$;
$\mathsf{CrptU}, \mathsf{SendToUser}, \mathsf{RevealSk}, \mathsf{RevealRt}, \mathsf{Sign}, \mathsf{ModifyReg})$;
If $\mathsf{Verify}(\mathbf{gpk}, M, \Sigma, S) = 0$ then return 0.
If $i \notin \mathbf{RU}$ or $i \in \mathbf{BU}$ or $(i, M, \Sigma) \in \mathbf{SL}$ then return 0 else 1.

Suppose there is a frameable adversary A with advantage ϵ. We construct a **PPT** algorithm B that solves *SIS* problem. B is given a matrix $\mathbf{F}$. B generates all the public keys and authority keys. Then B interacts with A by sending **gpk** and authority keys $(\mathbf{T_A}, \mathbf{T_B})$.

As discussed in [20], B responses to A's queries. A can act as a corrupted group manager and add a new user i to the group. When A requests user i to generate a signature on a message M, B generates and returns the signature $\Sigma=(\mathbf{ovk}, (\mathbf{c}_1, \mathbf{c}_2), \rho, \Pi, sig, \mathbf{v})$.

Finally, on a message M^*, A outputs $\Sigma^*=(\mathbf{ovk}^*, (\mathbf{c}_1^*, \mathbf{c}_2^*), \Pi^*, sig^*, \mathbf{v}^*)$, which opens to i^* who did not sign the message. Thus, (M^*, Σ^*) should frame user i^*. B has a short vector $\mathbf{z}_{i^*} = \mathbf{F} \cdot \mathbf{x}_{i^*} \mod q$. To solve SIS instance B should have another short vector $\mathbf{z}_{i\prime} = \mathbf{F} \cdot \mathbf{x}_{i\prime} \mod q$. To compute such a vector, B proceeds by replaying A sufficient times and applying Improved Forking Lemma [10].

As discussed in [20], B can extract a short vector $\mathbf{x\prime}$, where $\mathbf{z}_{i^*} = \mathbf{F} \cdot \mathbf{x\prime} \mod q$. According to Stern-like proof of knowledge, with overwhelming probability, we say $\mathbf{x\prime} \neq \mathbf{x}_{i^*}$. A nonzero vector $\mathbf{h} = \mathbf{x}_{i^*} - \mathbf{x\prime}$ is a solution for SIS problem.

This proves the non-frameability of our scheme.

6 Conclusion

This paper presented a simple lattice-based group signature scheme which satisfies both member registration and revocation with VLR. We have discussed VLR group signatures and difficulties of achieving full-anonymity for VLR group signatures. Moreover, we proved our scheme's security by suggesting a new security notion called dynamical-almost-full anonymity. However, achieving full-anonymity for VLR group signature schemes still remains as a problem.

Acknowledgments. This work is supported in part by JSPS Grant-in-Aids for Scientic Research (A) JP16H01705 and for Scientic Research (B) JP17H01695.

References

1. Agrawal, S., Boyen, X., Vaikuntanathan, V., Voulgaris, P., Wee, H.: Functional encryption for threshold functions (or fuzzy ibe) from lattices. In: PKC 2012, LNCS. vol. 7293, pp. 280–297. Springer Berlin Heidelberg (2012)
2. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 99–108. ACM (1996)
3. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: CRYPTO 2000, LNCS. vol. 1880, pp. 255–270. Springer Berlin Heidelberg (2000)
4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: EUROCRYPT 2003, LNCS. vol. 2656, pp. 614–629. Springer Berlin Heidelberg (2003)
5. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: CT-RSA 2005. vol. 3376, pp. 136–153. LNCS (2005)
6. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM-CCS 2004. pp. 168–177. ACM (2004)
7. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of fully dynamic group signatures. In: ACNS 2016, LNCS. vol. 9696, pp. 117–136. Springer, Cham (2016)
8. Bresson, E., Stern, J.: Efficient revocation in group signatures. In: PKC 2001, LNCS. vol. 1992, pp. 190–206. Springer Berlin Heidelberg (2001)
9. Brickell, E.: An efficient protocol for anonymously providing assurance of the container of the private key. *Submitted to the Trusted Comp. Group (April 2003)* (2003)
10. Brickell, E., Pointcheval, D., Vaudenay, S., Yung, M.: Design validations for discrete logarithm based signature schemes. In: PKC 2000, LNCS. vol. 1751, pp. 276–292. Springer Berlin Heidelberg (2000)
11. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: CRYPTO 2002, LNCS. vol. 2442, pp. 61–76. Springer Berlin Heidelberg (2002)
12. Camenisch, J., Neven, G., Rückert, M.: Fully anonymous attribute tokens from lattices. In: SCN 2012, LNCS. vol. 12, pp. 57–75. Springer Berlin Heidelberg (2012)

13. Chaum, D., Van Heyst, E.: Group signatures. In: EUROCRYPT 1991, LNCS. vol. 547, pp. 257–265. Springer Berlin Heidelberg (1991)
14. Chow, S.S., Wong, D.S.: Anonymous identification and designated-verifiers signatures from insecure batch verification. In: EuroPKI 2007, LNCS. vol. 4582, pp. 203–219. Springer Berlin Heidelberg (2007)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: ACM 2008. pp. 197–206. ACM (2008)
16. Gordon, S.D., Katz, J., Vaikuntanathan, V.: A group signature scheme from lattice assumptions. In: ASIACRYPT 2010, LNCS. vol. 6477, pp. 395–412. Springer Berlin Heidelberg (2010)
17. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: ASIACRYPT 2008, LNCS. vol. 5350, pp. 372–389. Springer Berlin Heidelberg (2008)
18. Laguillaumie, F., Langlois, A., Libert, B., Stehlé, D.: Lattice-based group signatures with logarithmic signature size. In: ASIACRYPT 2013, LNCS. vol. 8270, pp. 41–61. Springer Berlin Heidelberg (2013)
19. Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-based group signature scheme with verifier-local revocation. In: PKC 2014, LNCS. vol. 8383, pp. 345–361. Springer Berlin Heidelberg (2014)
20. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In: ASIACRYPT 2016, LNCS. vol. 10032, pp. 373–403. Springer Berlin Heidelberg (2016)
21. Libert, B., Vergnaud, D.: Group signatures with verifier-local revocation and backward unlinkability in the standard model. In: CANS 2009, LNCS. vol. 5888, pp. 498–517. Springer Berlin Heidelberg (2009)
22. Ling, S., Nguyen, K., Wang, H.: Group signatures from lattices: simpler, tighter, shorter, ring-based. In: PKC 2015, LNCS. vol. 9020, pp. 427–449. Springer Berlin Heidelberg (2015)
23. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Lattice-based group signatures: Achieving full dynamicity with ease. In: ACNS 2017, LNCS. vol. 10355, pp. 293–312. Springer International Publishing, Cham (2017)
24. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT 2012. vol. 7237, pp. 700–718. Springer Berlin Heidelberg (2012)
25. Naor, D., Shenhav, A., Wool, A.: One-time signatures revisited: Have they become practical? IACR Cryptology ePrint Archive 2005, 442 (2005)
26. Nguyen, P.Q., Zhang, J., Zhang, Z.: Simpler efficient group signatures from lattices. In: PKC 2015, LNCS. vol. 9020, pp. 401–426. Springer Berlin Heidelberg (2015)
27. Peikert, C.: A decade of lattice cryptography. Foundations and Trends in Theoretical Computer Science 10(4), 283–424 (2016), https://doi.org/10.1561/0400000074
28. Perera, M.N.S., Koshiba, T.: Fully dynamic group signature scheme with member registration and verifier-local revocation. In: ICMC 2018, Mathematics and Computing (to appear)
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: In STOC. pp. 84–93. ACM Press (2005)