Forward Secure Efficient Group Signature in Dynamic Setting using Lattices

Meenakshi Kansal, Ratna Dutta and Sourav Mukhopadhyay
Department of Mathematics,
Indian Institute of Technology Kharagpur,
Kharagpur-721302, India
`{kansal,ratna,sourav}@maths.iitkgp.ernet.in`

Abstract: Secret key exposure is at high risk in the computing infrastructure due to the increase in use of harmful devices. As a result, achieving *forward secrecy* is a preferable feature for any cryptosystem where the lifetime of a user is divided into discrete time periods. Forward secrecy preserves the security of past periods even if the secret key is exposed. In this work, we introduce the *first* lattice based *forward secure* dynamic group signature scheme. The existing forward secure group signature schemes are secure in the bilinear setting, and becomes insecure in the quantum computer period. We employ a complete binary tree whose leaves are associated with discrete time periods and label the nodes in a unique way that enables each node of the same depth to have different Hamming weight. This helps the group manager to produce distinct certificates to distinct users. Our scheme withstands framing attack, mis-identification attack and preserves anonymity under the learning with errors (LWE) and short integer solution (SIS) assumptions.

Keywords: Lattice based cryptography, Dynamic group signature, forward security, anonymity, traceability.

1 Introduction

Group signature scheme enables any group member to produce a signature *anonymously* on behalf of the group and in case of misbehavior, the signer can be traced out by a designated authority, called the Opening Authority (OA). Group signature has been introduced by Chaum and Heyst [16] in 1991. Since then various constructions of group signature have been proposed in classical cryptography both in the random oracle model and standard model based on either strong RSA or bilinear map. *Traceability* and *anonymity* are the security attributes of any group signature. Traceability ensures that only the group manager is able to determine which member of the group issued the legitimate signature while anonymity guarantees that no one other than the group manager should be able to determine any information about the signer. The confession page on Facebook can be seen as an example of a group signature scheme. The confession given by individuals is anonymous and can be traced out only by the administrator.

Group signature scheme can be broadly classified into two categories − static and dynamic. In *static* group setting, all the signing keys of the members are fixed during setup and hence the party handling setup phase has a high degree trust which is undesirable for practical real life applications. To overcome this problem, *dynamic* group setting has received considerable attention in the recent research community whereby a member can join the group at anytime and receives his private signing key during the joining period. Consequently, member identities are not fixed at the initial setup phase. Dynamic setting has two different parties called group manager and opening authority for issuing certificates to users and to trace the signature respectively. In case of static setting, group manager acts as the opening authority. The security of static group signatures was formalized by Bellare et al. [7]. In 2004 Bellare, Shi, and Zhang [9] formally defined the dynamic case. Later, in 2010, Gordon et al. [20] presented the first lattice based construction of group signature scheme followed by a number of works based on lattices [[14], [23], [24], [28], [31]]. Lattice based cryptography is a promising tool even in quantum era as there is no quantum algorithm yet to solve hard problems from which lattice based cryptosystems derive their security. Recently, in 2016, Libert et al. [25] proposed the first construction of dynamic group signature based on lattices.

Our contribution: We address the problem of designing efficient group signature scheme in dynamic setting from lattices featuring *forward secrecy*. With the increasing use of damaging tools in computing infrastructure, there is high risk in key exposure. Forward secrecy is achieved by partitioning the lifetime of a user into distinct and discontinuous time periods. It keeps the security of past periods secure even when the secret key of the user is exposed. Introducing forward secrecy in a dynamic group signature scheme without blowing up the storage, communication and computation overheads is a non-trivial task. All the existing works for group signature schemes with forward secrecy are in bilinear setup. There is no group signature based on lattices possessing forward secrecy so far to the best of our knowledge. In this paper, we improvise Libert et al. [25] dynamic group signature scheme into forward secure dynamic group signature scheme using a complete binary tree whose leaf nodes indicate discrete time periods. We label each node of the tree in a suitable manner that enables nodes at the same depth to have different Hamming weights. When a user joins, it is issued a certificate for a time period by the group manager. We skillfully utilize the labeling of this tree to generate distinct matrices for different users in issuing distinct certificates. Similar to [25], we employ Gentry-Peikert-Vaikuntanathan Identity based encryption (GPV-IBE) [18] and zero knowledge argument of knowledge to generate signature. We provide a concrete security analysis of our construction against framing attack, mis-identification attack and anonymity attack following the stronger notion of security for forward secrecy specified by Libert et al. [26]. Our scheme achieves security in the random oracle model under the hardness of learning with errors (LWE) problem and short integer solution (SIS) problem. We briefly summarize the comparison of our scheme with the existing lattice based group signature schemes [14], [20], [23], [25], [28] in Table 1 where N denotes the total number of group members and n is the security parameter. More precisely, we note the

Scheme	Forward secure	Dynamic	Signature size	Public key size	Certificate size
GKV [20]	No	No	$N \cdot \tilde{\mathcal{O}}(n^2)$	$N \cdot \tilde{\mathcal{O}}(n^2)$	-
CNR [14]	No	No	$N \cdot \tilde{\mathcal{O}}(n^2)$	$N \cdot \tilde{\mathcal{O}}(n^2)$	-
LLLS [23]	No	No	$\log N \cdot \tilde{\mathcal{O}}(n)$	$\log N \cdot \tilde{\mathcal{O}}(n^2)$	-
LNW1 [28]	No	No	$\log N \cdot \tilde{\mathcal{O}}(n)$	$\log N \cdot \tilde{\mathcal{O}}(n^2)$	-
LNW2 [28]	No	No	$\log N \cdot \tilde{\mathcal{O}}(n)$	$\log N \cdot \tilde{\mathcal{O}}(n)$	-
LLMNW [25]	No	Yes	$\log N \cdot \tilde{\mathcal{O}}(n)$	$\log N \cdot \tilde{\mathcal{O}}(n^2)$	$\log N \cdot \mathcal{O}(n)$
Ours	Yes	Yes	$\log N \cdot \tilde{\mathcal{O}}(n^3)$	$\log N \cdot \tilde{\mathcal{O}}(n^2)$	$\log N \cdot \tilde{\mathcal{O}}(n^2)$

Table 1: Comparative summary of lattice based group signature schemes

following:

(i) The certificate size in our construction is $\tilde{\mathcal{O}}(n^2)\log N$ while that of [25] is $\mathcal{O}(n)\log N$, where N is the total allowable group members and n is the security parameter. However, public key size in our scheme is same as of [25].

(ii) In contrast to [25], our scheme uses delegation process whereby the certificate once issued at the initial time, gets updated by the group members themselves without any interaction with the group manager.

(iii) The schemes [14], [20], [23], [28] are all static where signing keys of the members are fixed during setup. Although the scheme [25] is dynamic and has the flexibility to issue secret key at the joining time of a user, it does not achieve forward secrecy. On contrary, our scheme is the only forward secure group signature scheme in dynamic setting that derives its security from the hard problems based on lattices.

Paper Organization: The rest of the paper is organized as follows. Section 2, provides preliminaries and background materials. Syntax and security model of dynamic forward secure group signature scheme are presented in Section 3. Our scheme is described in Section 4. The underlying zero knowledge argument of knowledge is included in Section 5 and security of our scheme is discussed in Section 6 respectively. Finally, the paper is concluded in Section 7.

2 Background and Assumptions

Notations: Throughout the paper, we use the following notations unless otherwise stated:

- We use lower bold case letters for vectors and upper bold case letters for matrices. Wherever log is used we assume that base is 2. We refer by "||" the concatenation of strings or matrix columns and by "|" the concatenation of matrices.
- The Euclidean norm of the vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is denoted by $||\mathbf{x}|| = (x_1^2 + x_2^2 + \ldots + x_n^2)^{1/2}$ and infinity norm is denoted by $||\mathbf{x}||_\infty = \max\limits_{1 \le i \le n} x_i$. The Gram-Schmidt orthogonalization of matrix $\mathbf{A}$ is denoted by $\widetilde{\mathbf{A}}$. Transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^t$.

- The notation $\mathbf{A} \hookleftarrow \triangle$ represents that $\mathbf{A}$ is a matrix following distribution $\triangle$. A vector $\mathbf{u} \in \mathbb{Z}^m$ denotes a column vector of size $m \times 1$ with integer entries.
- A function $f(n) \in \widetilde{\mathcal{O}}(g(n))$ indicates $f(n) \in \mathcal{O}(g(n) \log^k g(n))$, for some $k \in \mathbb{N}$. We say that a function f is negligible in λ if $f = \lambda^{-\omega(1)}$.
- Two distributions X and Y, defined over a countable domain D, are said to be statistically closed if their statistical difference, $\frac{1}{2}\sum\limits_{\alpha} |Pr[X = \alpha] - Pr[Y = \alpha]|$ is negligible.
- Two problems are computationally equivalent if one can be reduced to the other. In other words, both the problems are essentially as hard or as easy.
- A vector $\mathbf{v}$ is said to be a "short vector" if given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda(\mathbf{B})$, we have $||\mathbf{v}|| \leq \gamma(n)\lambda(\Lambda)$, where γ is an approximation factor taken to be a function of the lattice dimension n and $\lambda(\Lambda) = \min\limits_{\mathbf{v} \in \Lambda(\mathbf{B}) - \{0\}} ||\mathbf{v}||$.

Definition 1. (Lattice). *Let $\boldsymbol{B} = \{\boldsymbol{b}_i\}_{i \leq n}$ be a linearly independent set of vectors of $\mathbb{R}^n$. A lattice generated by $\boldsymbol{B}$ is defined as $\Lambda(\boldsymbol{B}) = \{\sum\limits_{\boldsymbol{b}_i \in \boldsymbol{B}} c_i \boldsymbol{b}_i : c_i \in \mathbb{Z}\}$, the set of all integer linear combinations of $\boldsymbol{b}_i \in \boldsymbol{B}$. The set $\boldsymbol{B}$ is said to be a basis of the lattice Λ.*

For $q \in \mathbb{N}$, matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define the following three q-ary lattices generated by $\mathbf{A}$:

$$\begin{aligned}
\Lambda_q^{\perp}(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\} \\
\Lambda_q^{\mathbf{u}}(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\} \\
\Lambda_q(\mathbf{A}) &= \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}^t\mathbf{s} = \mathbf{x} \bmod q, \text{ for some } \mathbf{s} \in \mathbb{Z}_q^n\},
\end{aligned}$$

where m, n are integers with $m \geq n \geq 1$.

Definition 2. (Gaussian distribution over a lattice). *For a lattice Λ and a real number $\sigma > 0$, discrete Gaussian distribution over Λ centered at 0, denoted by $D_{\Lambda,\sigma}$, is defined as: $\forall \boldsymbol{y} \in \Lambda$, $D_{\Lambda,\sigma}[\boldsymbol{y}] \sim exp(-\pi||\boldsymbol{y}||^2/\sigma^2)$, meaning that $D_{\Lambda,\sigma}[\boldsymbol{y}]$ is proportional to $exp(-\pi||\boldsymbol{y}||^2/\sigma^2)$.*

Lemma 2.1. *For any n-dimensional lattice Λ, and for $\sigma > 0$,*

(a) $Pr_{\boldsymbol{b} \hookleftarrow D_{\Lambda,\sigma}}[||\boldsymbol{b}|| \leq \sqrt{n}\sigma] \geq 1 - 2^{-\Omega(n)}$ [6], *and*

(b) there exists the following probabilistic polynomial time (PPT) algorithms :

(i) GPVSample$(\boldsymbol{B}, \sigma) \longrightarrow (\boldsymbol{b} \in \Lambda)$ [12]. *On input a basis $\boldsymbol{B}$ of a lattice Λ and a rational number $\sigma \geq ||\widetilde{\boldsymbol{B}}||\Omega(\sqrt{\log n})$, this algorithm outputs a vector $\boldsymbol{b} \in \Lambda$ with distribution $D_{\Lambda,\sigma}$. Here $\widetilde{\boldsymbol{B}}$ is the Gram-Schmidt orthogonalization of $\boldsymbol{B}$.*

(ii) SamplePre$(\boldsymbol{A}, \mathsf{T}_{\boldsymbol{A}}, \boldsymbol{u}, \sigma) \longrightarrow (\boldsymbol{x} \in \Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}))$ [18]. *This algorithm takes as input a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, a short basis $\mathsf{T}_{\boldsymbol{A}}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$, a vector $\boldsymbol{u}$ whose preimage is to be sampled and the standard deviation σ of the distribution from which the preimage is to be sampled. The algorithm returns a short vector $\boldsymbol{x} \in \Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ sampled from a distribution statistically close to $D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma}$ whenever $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ is non empty i.e., $\boldsymbol{x}$ satisfies the relation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \bmod q$.*

(iii) $\mathsf{TrapGen}(1^n, 1^m, q) \longrightarrow (\boldsymbol{A}, \mathsf{T}_{\boldsymbol{A}})$ [4]. *Taking $1^n, 1^m, q$ as input, this algorithm outputs a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and a basis $\mathsf{T}_{\boldsymbol{A}}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\boldsymbol{A}$ is within the statistical distance $2^{-\Omega(n)}$ to $U(\mathbb{Z}_q^{n\times m})$ and $||\widetilde{\mathsf{T}}_{\boldsymbol{A}}|| \leq \mathcal{O}(\sqrt{n\log q})$. Here $U(\mathbb{Z}_q^{n\times m})$ is the uniform distribution of integer matrices of order $n \times m$ and $\widetilde{\mathsf{T}}_{\boldsymbol{A}}$ is the Gram-Schmidt orthogonalization of $\mathsf{T}_{\boldsymbol{A}}$.*

(iv) $\mathsf{ExtBasis}(\boldsymbol{A}, \bar{\boldsymbol{A}}, \mathsf{T}_{\boldsymbol{A}}) \longrightarrow (\mathsf{T}_{\boldsymbol{A}|\bar{\boldsymbol{A}}})$ [15]. *Given a matrix $\boldsymbol{A}$ whose columns span $\mathbb{Z}_q^n$, a basis $\mathsf{T}_{\boldsymbol{A}}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$ and an arbitrary matrix $\bar{\boldsymbol{A}}$, this algorithm outputs a basis $\mathsf{T}_{\boldsymbol{A}|\bar{\boldsymbol{A}}}$ of $\Lambda_q^{\perp}(\boldsymbol{A}|\bar{\boldsymbol{A}})$ such that $||\widetilde{\mathsf{T}}_{\boldsymbol{A}|\bar{\boldsymbol{A}}}|| = ||\widetilde{\mathsf{T}}_{\boldsymbol{A}}||$.*

(v) $\mathsf{SampleRight}(\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{R}, T_{\boldsymbol{C}}, \sigma, \boldsymbol{u}) \longrightarrow (\boldsymbol{b} \in \mathbb{Z}^{2m})$ [1]. *On input two matrices $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{Z}_q^{n\times m}$, a low norm matrix $\boldsymbol{R} \in \mathbb{Z}^{m\times m}$, a short basis $T_{\boldsymbol{C}} \in \mathbb{Z}^{m\times m}$ of $\Lambda_q^{\perp}(\boldsymbol{C})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a parameter $\sigma \geq ||\widetilde{T}_{\boldsymbol{C}}||\Omega(\sqrt{\log n})$, this algorithm outputs a random vector $\boldsymbol{b} \in \mathbb{Z}^{2m}$ satisfying $[\boldsymbol{A}|\boldsymbol{A}\boldsymbol{R} + \boldsymbol{C}]\boldsymbol{b} = \boldsymbol{u} \mod q$ with distribution statistically close to the Gaussian distribution $D_{\Lambda,\sigma}$ where Λ denote the shifted lattice $\{\boldsymbol{x} \in \mathbb{Z}^{2m} : [\boldsymbol{A}|\boldsymbol{A}\boldsymbol{R} + \boldsymbol{C}]\boldsymbol{x} = \boldsymbol{u} \mod q\}$.*

(vi) $\mathsf{BasisDel}(\boldsymbol{A}, \boldsymbol{R}, \mathsf{T}_{\boldsymbol{A}}, \sigma) \longrightarrow (\boldsymbol{B} = \boldsymbol{A}\boldsymbol{R}^{-1}, \mathsf{T}_{\boldsymbol{B}})$[2]. *This algorithm takes as input a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, a $\mathbb{Z}_q$-invertible matrix $\boldsymbol{R} \in \mathbb{Z}^{m\times m}$ sampled from $(D_{\mathbb{Z}^m,\sigma})^m$ (distribution on matrices in $\mathbb{Z}^{m\times m}$), a basis $\mathsf{T}_{\boldsymbol{A}}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$, and a real parameter $\sigma > 0$ and outputs a matrix $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{R}^{-1}$ along with a basis $\mathsf{T}_{\boldsymbol{B}}$ of $\Lambda_q^{\perp}(\boldsymbol{B})$.*

(vii) $\mathsf{SampleRwithBasis}(\boldsymbol{A}) \longrightarrow (\boldsymbol{R}, \mathsf{T}_{\boldsymbol{B}})$[2]. *On input a matrix $\boldsymbol{A}$, this algorithm outputs a random $\mathbb{Z}_q$-invertible low norm matrix $\boldsymbol{R}$ and a basis $\mathsf{T}_{\boldsymbol{B}}$ of $\Lambda_q^{\perp}(\boldsymbol{B})$ as follows:*

```
(1) TrapGen($1^n, 1^m, q$) ⟶ ($\boldsymbol{B}, \mathsf{T}_{\boldsymbol{B}}$).
(2) for $i = 1, 2, \ldots, m$ do
    (a) SamplePre($\boldsymbol{B}, \mathsf{T}_{\boldsymbol{B}}, \boldsymbol{a}_i, \sigma$) ⟶ ($\boldsymbol{r}_i \in \Lambda_q^{\boldsymbol{a}_i}(\boldsymbol{B})$) i.e., $\boldsymbol{B}\boldsymbol{r}_i = \boldsymbol{a}_i \mod q$.
    (b) Repeat step (a) until $\boldsymbol{r}_i \in \mathbb{Z}_q$ is linearly independent of $\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{i-1}$.
(3) Form a matrix $\boldsymbol{R} \in \mathbb{Z}^{m\times m}$ with sampled $\boldsymbol{r}_i$'s as $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_m)$.
end do
```

Thus $\boldsymbol{B}\boldsymbol{R} = \boldsymbol{A} \mod q$ or $\boldsymbol{B} = \boldsymbol{A}\boldsymbol{R}^{-1} \mod q$.

2.1 Hardness Assumptions

Definition 3. (Inhomogeneous short integer solution (ISIS)[3]). *Given an integer q, a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ and a real number β, the ISIS problem is to find an integer vector $\boldsymbol{e} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \mod q$ with $||\boldsymbol{e}|| \leq \beta$.*

Remark 1. *If a bound on $\boldsymbol{e}$ is not given, then the equation $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \mod q$ is solvable in polynomial time using Gauss elimination. The ISIS problem is to find $\boldsymbol{e}$ of the equation*

$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \bmod q$ *with* $||\boldsymbol{e}|| \leq \beta$ *which is difficult. However, if a short basis* $\mathsf{T}_{\boldsymbol{A}}$ *of* $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ *is known then* ISIS *problem can be efficiently solved.*

Definition 4. (Short integer solution (SIS)[3]). *Given an integer* q*, a matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$*, and a real number* β*, the* SIS *problem is to find an integer vector* $\boldsymbol{e} \in \mathbb{Z}^m$ *such that* $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{0} \bmod q$ *with* $||\boldsymbol{e}|| \leq \beta$*.*

Definition 5. (Learning with errors (LWE)[32]). *Let* χ *be a distribution on* $\mathbb{Z}$*,* $n \geq 1$ *be any integer and* $p \geq 2$ *be any prime. For any* $\boldsymbol{s} \in \mathbb{Z}_p^n$*, given arbitrarily many samples of the form* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e)$ *with* $\boldsymbol{a}$ *uniform in* $\mathbb{Z}_p^n$ *and* e *is sampled from* χ*, the* search *LWE problem is to find* $\boldsymbol{s}$ *and the* decisional *LWE problem is to distinguish the distribution of* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e)$ *from the uniform distribution* $U(\mathbb{Z}_p^n \times \mathbb{Z}_p)$*. Here* $\langle \boldsymbol{a}, \boldsymbol{s}\rangle = \boldsymbol{a}^t\boldsymbol{s}$*.*

The search LWE and the decisional LWE are computationally equivalent [30].

2.2 Node Select Algorithm [26]

$\mathsf{Nodes}(t_1 + 1, t_2, G) \longrightarrow (\mathsf{SubsetHN})$. Following Libert et al. [26], we describe this procedure in Algorithm 1 which on input a time interval $(t_1 + 1, t_2)$, a complete binary tree G, returns a subset SubsetHN of hanging nodes in the given time interval. Let the height of the binary tree G be l. Then the number of leaves in G is $T = 2^l$. The leaves of G denote the time period which are binary strings. Each node in G is assigned a label in such a way that no two nodes at the same level have the same Hamming weight. One such assignment of labels is shown in Figure 1.

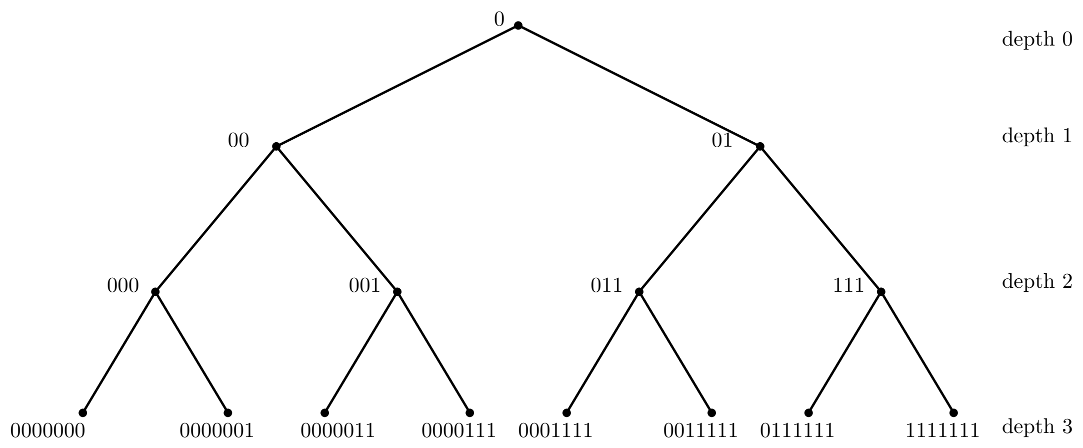

Figure 1: Node Labeling

The root is at depth 0 and is assigned label 0. The two nodes at depth 1 are each assigned label of 2 bits, say, 00 and 01 with different Hamming weights. The $4 = 2^2$ nodes at depth

2 require $2^2 - 1 = 3$-bit each with different Hamming weights and are assigned labels, say, 000, 001, 011, 111. Similarly, all the 2^p nodes at depth p can be assigned $(2^p - 1)$-bit labels with different Hamming weights. For a node w, let w_{dep} denotes its depth and w_{num} denotes its assigned label. Note that w_{num} is a binary string of length $2^{w_{\mathsf{dep}}} - 1$ where $0 \leq w_{\mathsf{dep}} \leq l$. There are total $T = 2^l$ time periods $\{1, 2, ..., T\}$ and the period t corresponds to the t-th node at depth l from left having $(2^l - 1)$-bit label denoted by w^{t-1}_{num}.

Let Path(u) be the set of all nodes on the path from u to the root of the tree G. If a node w is a left or a right child of some node in Path(u) with $w \notin \mathsf{Path}$(u), then w is a hanging node with respect to Path(u). For a node u at depth $\leq l$, its left and right child are denoted by u_0, u_1 respectively.
For instance, let $w^{t_1-1}_{\mathsf{num}} = 0000001, w^{t_2}_{\mathsf{num}} = 0111111$ then $X_1 = \{000, 00, 0\}$ and $X_2 = \{111, 01, 0\}$. For all $w \in X_1$, $\mathsf{SubsetHN} = \{001\}$ and for all $w \in X_2$, $\mathsf{SubsetHN} = \mathsf{SubsetHN} \cup \{011\}$. Hence the resultant ouput is $\mathsf{SubsetHN} = \{001, 011\}$.

Remark 2. *Node Select algorithm outputs a subset of hanging nodes in such a way that the exactly one ancestor of each leaf between the time period t_1 and $t_2 - 1$ are being selected.*

Algorithm 1: $\mathsf{Nodes}(t_1 + 1, t_2, G)$

```
Input   : $t_1 + 1, t_2, G$
Output: SubsetHN
if($t_1 + 1 > t_2$) then
        return 0;
   else
      $X_1, X_2, \mathsf{SubsetHN} \leftarrow \emptyset$;
      $X_1 \leftarrow X_1 \cup \mathsf{Path}(w^{t_1-1}_{\mathsf{num}})$;          // $w^{t_1-1}_{\mathsf{num}}$ is the labeling of the node at period $t_1$
      $X_2 \leftarrow X_2 \cup \mathsf{Path}(w^{t_2}_{\mathsf{num}})$;          // $w^{t_2}_{\mathsf{num}}$ is the labeling of the node at period $t_2 + 1$
        for($w \in X_1$) do
              if($w_1 \notin X_1 \cup X_2$) then
              $\mathsf{SubsetHN} \leftarrow \mathsf{SubsetHN} \cup \{w_1\}$          // Here $w_1$ is the right child of $w$
              end if
        end do
        for($w \in X_2$) do
              if($w_0 \notin X_1 \cup X_2$) then
              $\mathsf{SubsetHN} \leftarrow \mathsf{SubsetHN} \cup \{w_0\}$          // Here $w_0$ is the left child of $w$
              end if
        end do
end if
return  SubsetHN;
```

2.3 GPV-IBE [18]

The identity based encryption (IBE) by Gentry, Peikert and Vaikuntanathan [18] is a 4-tuple $\mathsf{GPV\text{-}IBE}$= (Setup, $\mathsf{Extract}$, Enc, Dec) consisting of 3 PPT algorithms Setup, $\mathsf{Extract}$, Enc and a deterministic algorithm Dec.

- $\mathsf{GPV\text{-}IBE.Setup}(1^n, 1^m, q) \longrightarrow (\mathsf{MPK}, \mathsf{MSK})$. A trusted third party, called a key generation centre (KGC), runs this PPT algorithm by executing $\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, \mathsf{T}_\mathbf{A})$ described in Lemma 1.1$(b)(iii)$ and generates a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ along with a basis $\mathsf{T}_\mathbf{A}$ of $\Lambda_q^\perp(\mathbf{A})$ such that $\mathbf{A}$ is within the statistical distance $2^{-\Omega(n)}$ to $U(\mathbb{Z}_q^{n\times m})$ and $||\widetilde{\mathsf{T}}_\mathbf{A}|| \leq \mathcal{O}(\sqrt{n \log q})$. The KGC also chooses a cryptographically secure hash function $\mathcal{F} : \{0,1\}^* \rightarrow \mathbb{Z}_q^{n\times r}$ for any integer $r \geq 1$. It sets the master public key MPK=$(\mathbf{A}, \mathcal{F}, m, n, q)$ and the master secret key MSK=$\mathsf{T}_\mathbf{A}$. The KGC makes MPK public and keeps MSK secret to itself.
- $\mathsf{GPV\text{-}IBE.Extract}(\mathsf{MPK}, \mathsf{MSK}, \mathsf{id}) \longrightarrow (\mathsf{sk}_\mathsf{id})$. This PPT algorithm is run by the KGC. The user with identity id asks for its secret to the KGC. If the identity $\mathsf{id} \in \{0,1\}^*$ has already been queried and $(\mathsf{id}, \mathbf{E})$ is in the local storage of the KGC for some $\mathbf{E} \in \mathbb{Z}_q^{r\times m}$ then it returns $\mathbf{E}$ to the user. Otherwise, the KGC computes $\mathbf{R} = \mathcal{F}(\mathsf{id}) \in \mathbb{Z}_q^{n\times r}$ and generates $\mathbf{E} \hookleftarrow \mathsf{SamplePre}(\mathbf{A}, \mathsf{T}_\mathbf{A}, \mathbf{R}, \sigma)$ by using multiversion of Lemma 1.1$(b)(ii)$, where $\mathcal{F}, \mathbf{A}$ are extracted from MPK and $T_\mathbf{A}$ from MSK. Here σ is chosen by the KGC and $\mathbf{E} \in \mathbb{Z}_q^{r\times m}$ is an element of the lattice $\Lambda_q^\mathbf{R}(\mathbf{A})$ with a distribution statistically close to $D_{\Lambda_q^\mathbf{R}(\mathbf{A}),\sigma}$. Consequently, $\mathbf{E}$ satisfies the equation $\mathbf{AE} = \mathbf{R} \bmod q$. The KGC sends $\mathsf{sk}_\mathsf{id} = \mathbf{E}$ through a secure channel to the user as its secret key.
- $\mathsf{GPV\text{-}IBE.Enc}(\mathsf{MPK}, \mathsf{id}, \mathbf{b}) \longrightarrow (\mathbf{p}, \mathbf{c})$. This PPT algorithm is executed by an encrptor to encrypt a bit string $\mathbf{b} \in \{0,1\}^r$ using MPK=$(\mathbf{A}, \mathcal{F}, m, n, q)$. The encryptor with identity $\mathsf{id} \in \{0,1\}^*$
  - samples $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$, $\mathbf{x} \hookleftarrow \chi^m$, $\mathbf{y} \hookleftarrow \chi^r$, where χ is the LWE distribution;
  - computes $\mathbf{p} = \mathbf{A}^t\mathbf{s} + \mathbf{x} \in \mathbb{Z}_q^m$;
  - sets $\mathbf{c} = \mathbf{R}^t\mathbf{s} + \mathbf{y} + \mathbf{b}\lfloor q/2 \rfloor \in \mathbb{Z}_q^r$ where $\mathbf{R} = \mathcal{F}(\mathsf{id})$;
  - finally, sends the ciphertext ($\mathbf{p}$,$\mathbf{c}$) over a public channel.
- $\mathsf{GPV\text{-}IBE.Dec}(\mathsf{sk}_\mathsf{id}, (\mathbf{p}, \mathbf{c})) \longrightarrow (0 \vee 1)$. It is a deterministic algorithm run by the decryptor. The decryptor computes $\mathbf{b}' = (b'_1, b'_2, ..., b'_r) = \mathbf{c} - \mathbf{E}^t\mathbf{p} \in \mathbb{Z}_q^r$, using its own secret key sk_id=$\mathbf{E} \in \mathbb{Z}_q^{r\times m}$. For $i = 1, 2, ..., r$, it outputs $b'_i = 0$ if b'_i is closed to 0 mod q; otherwise returns $b'_i = 1$.

We have $b' = \mathbf{g}^t\mathbf{s} + y + b\lfloor q/2 \rfloor - \mathbf{e}^t(\mathbf{A}^t\mathbf{s} + \mathbf{x})$ and also $\mathbf{Ae} = \mathbf{g} \mod q$. So compute $b' = (y - \mathbf{e}^t\mathbf{x}) + b\lfloor q/2 \rfloor$; output 1 if b' is close to $\lfloor q/2 \rfloor$ than to 0 modulo q; otherwise output 0.
Correctness: Let $\delta > 0$ satisfying $\mathsf{Pr}_{\mathbf{x}\in\chi^m, y\in\chi}[|y - \mathbf{e}^t\mathbf{x}| < \frac{q}{5}] > 1 - \delta$. Then the probability of decryption error is atmost δ.
Security: Let $q \geq 5\sigma(m+1)$, $\sigma \geq \omega(\sqrt{\log m})$ and $m \geq 2n \log q$. Then the $\mathsf{GPV\text{-}IBE}$ scheme is CPA-secure and anonymous, assuming that $\mathsf{LWE}_{q,\chi}$ is hard.

2.4 Decomposition-Extension Technique [27]

This section describes a procedure Ext for extending and a procedure Dec-Ext for extending as well as decomposing a vector following Ling et al. [27].
Let m be an arbitrary dimension and B be an infinity norm bound. Let $B^2_m = \{\mathbf{x} \in \{0,1\}^{2m} :$ $\mathbf{x}$ has exactly m co-ordinates equal to j, for every $j \in \{0,1\}\}$ and $B^3_{mp} = \{\mathbf{x} \in \{-1,0,1\}^{3mp} :$ $\mathbf{x}$ has exactly mp co-ordinates equal to j, $\forall j \in \{-1,0,1\}\}$ where $p = \lfloor \log_2 B \rfloor + 1$. Let S_{3mp} be the set of permutations of $3mp$ length vectors. Then for any $\pi \in S_{3mp}$, we have $\widehat{\mathbf{w}} \in B^3_{mp} \Leftrightarrow \pi(\widehat{\mathbf{w}}) \in B^3_{mp}$.

- $\mathsf{Ext}_m(\mathbf{w}) \longrightarrow (\widehat{\mathbf{w}} \in B^2_m)$ [25]. This algorithm takes an input a vector $\mathbf{w} \in \{0,1\}^m$ with λ_0, λ_1 respectively the number of 0's and number of 1's in $\mathbf{w}$. It selects a random vector $\widetilde{\mathbf{w}}$ having exactly $(m-\lambda_0)$ many 0's and $(m-\lambda_1)$ many 1's and outputs the extended vector $\widehat{\mathbf{w}} = (\mathbf{w}||\widetilde{\mathbf{w}}) \in B^2_m$. Then for any permutation $\pi : B^2_m \to B^2_m$, we have $\widehat{\mathbf{w}} \in B^2_m \Leftrightarrow \pi(\widehat{\mathbf{w}}) \in B^2_m$.

- $\mathsf{Dec\text{-}Ext}_{m,p}(\mathbf{w}) \longrightarrow (\widehat{\mathbf{w}} \in B^3_{mp})$ [25]. On input $\mathbf{w} = (w_1, w_2, ..., w_m)$, $w_i \in [-B, B]$, $1 \le i \le m$, , this algorithm works as follows:
  - First, define a finite super-decreasing sequence $\{B_j\}_{j=1}^p$ of integers by setting $B_1 = \lceil \frac{B}{2} \rceil$ and $B_j = \lceil \frac{B-(B_1+B_2+\cdots+B_{j-1})}{2} \rceil$ for $2 \le j \le p$;
  - As $\{B_j\}$ is super-decreasing, for any $v \in [-B, B]$, one can efficiently compute $v^{(1)}, v^{(2)}, \ldots, v^{(p)} \in \{-1,0,1\}$ such that $\sum_{j=1}^{p} B_j v^{(j)} = v$. Note that $v^{(1)}, v^{(2)}, \ldots, v^{(p)} \in \{-1,0,1\}$ is a solution of the subset sum problem with weights $\{B_j\}_{j=1}^p$ and capacity v and is solvable in polynomial time when the sequence $\{B_j\}_{j=1}^p$ is super decreasing.
  - Next, define an $m \times mp$ matrix $\mathbf{K}_{m,B}$ as
$$\mathbf{K}_{m,B} = I_m \otimes [B_1|B_2|...|B_p] = \begin{bmatrix} B_1\ B_2 \ldots B_p & & & \\ & B_1\ B_2 \ldots B_p & & \\ & & \cdots & \\ & & & B_1\ B_2 \ldots B_p \end{bmatrix}$$
and set $\widehat{\mathbf{K}}_{m,B} = [\mathbf{K}_{m,B} | 0^{m \times 2mp}] \in \mathbb{Z}^{m \times 3mp}$.
  - For the given vector $\mathbf{w} = (w_1, w_2, \ldots, w_m)^t$, one can efficiently compute $w_i^{(1)}, w_i^{(2)}$, $\ldots, w_i^{(p)} \in \{-1,0,1\}$ corresponding to each $w_i \in [-B, B], 1 \le i \le m$, satisfying $\sum_{j=1}^{p} B_j w_i^{(j)} = w_i$. Then $\mathbf{K}_{m,B}\mathbf{w}' = \mathbf{w}$ for the vector $\mathbf{w}' = (w_1^{(1)}, \ldots, w_1^{(p)}, w_2^{(1)}, \ldots, w_2^{(p)},$ $\ldots, w_m^{(1)}, \ldots, w_m^{(p)}) \in \{-1,0,1\}^{mp}$.
  - Append a $2mp$ length vector $\widetilde{\mathbf{w}}$ to $\mathbf{w}'$, following the procedure Ext to obtain $\widehat{\mathbf{w}} = (\mathbf{w}||\widetilde{\mathbf{w}}) \in B^3_{mp}$. As the last $2mp$ columns of $\widetilde{\mathbf{K}}_{m,B}$ are all 0, the vector $\widehat{\mathbf{w}}$ satisfies $\widehat{\mathbf{K}}_{m,B}\widehat{\mathbf{w}} = \mathbf{K}_{m,B}\mathbf{w}' = \mathbf{w}$.

Example 1. *Let $B = 3$ and $m = 2$. Consider the vector $\boldsymbol{w} = (w_1, w_2)^t = (-2, 1)^t$ where $w_1, w_2 \in [-B, B]$. We generate the super-decreasing sequence $\{B_j\}_{j=1}^p$ of integers where $p = \lfloor \log B \rfloor + 1 = 2$ by setting $B_1 = \lceil \frac{3}{2} \rceil = 2, B_2 = \lceil \frac{3-B_1}{2} \rceil = 1$.*
Now,

$$w_1 = -2 = 2.w_1^{(1)} + 1.w_1^{(2)} \text{ gives } w_1^{(1)} = -1, w_1^{(2)} = 0$$

.
Similarly,

$$w_2 = 1 = 2.w_2^{(1)} + 1.w_2^{(2)} \text{ gives } w_2^{(1)} = 0, w_2^{(2)} = 1$$

Therefore, $\boldsymbol{w}' = (w_1^{(1)}, w_1^{(2)}, w_2^{(1)}, w_2^{(2)}) = (-1, 0, 0, 1) \in \{-1, 0, 1\}^{mp}$ satisfies $K_{m,B}\boldsymbol{w}' = \boldsymbol{w}$, where $K_{m,B} = K_{2,3}$ is an $m \times mp$ matrix given by $K_{2,3} = I_2 \otimes [B_1, B_2] = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \end{bmatrix}$. Next we set,

$$\widehat{K}_{2,3} = \left[\begin{array}{cccc|cccc|cccc} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right]$$

Note that the number of -1's, 0's, 1's in $\boldsymbol{w}'$ are respectively $\lambda_{-1} = 1, \lambda_0 = 2, \lambda_1 = 1$. We append to $\boldsymbol{w}'$ a vector $\widetilde{\boldsymbol{w}}$ of length $2mp = 2 \times 2 \times 2 = 8$ having $mp - \lambda_{-1} = 3$ many -1's, $mp - \lambda_0 = 2$ many 0's, $mp - \lambda_1 = 3$ many 1's following the procedure Ext *and obtain an extended vector $\widehat{\boldsymbol{w}} = (\boldsymbol{w}'||\widetilde{\boldsymbol{w}})$ that satisfies $\widehat{K}_{2,3}\widehat{\boldsymbol{w}} = \boldsymbol{w}$. For instance, we can choose $\widetilde{\boldsymbol{w}} = (0, 1, -1, 1| -1, -1, 1, 0)$ and hence $\widehat{\boldsymbol{w}} = (\boldsymbol{w}'||\widetilde{\boldsymbol{w}}) = (-1, 0, 0, 1|0, 1, -1, 1| -1, -1, 1, 0)$. Then*

$$\begin{aligned} \widehat{K}_{2,3}\widehat{\boldsymbol{w}} = & \left[\begin{array}{cccc|cccc|cccc} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \\ & \times [-1, 0, 0, 1, 0, 1, -1, 1, -1, -1, 1, 0]^t \\ = & (-2, 1)^t = \boldsymbol{w} \end{aligned}$$

2.5 Zero Knowledge Argument Systems

In this section, we describe zero knowledge argument of knowledge (ZKAoK) for the relation $\mathbf{Px} = \mathbf{v}$ given in [25]. Here $\mathbf{P}$ is any matrix and $\mathbf{v}$ is a vector (or matrix), both publicly available and $\mathbf{x}$ is a secret vector (or matrix) with some conditions to be proven in zero-knowledge. Utilizing permutation, Stern designed in [34] a protocol that proves the Hamming weight of a binary vector $\mathbf{x}$ in zero-knowledge. Ling et al. [27] extended Stern's protocol to prove the possession of any vector (or matrix) $\mathbf{x}$, with a bound on the norm of $\mathbf{x}$ using procedure Dec-Ext described in Section 2.4.
Let $q \geq 2$ and D, L be two positive integers. We consider a subset VALID of all L-length strings over $\{-1, 0, 1\}$. Let S be any finite set of permutations such that for any $\pi \in S$, we can associate a permutation T_π of L elements satisfying

$$\left.\begin{array}{c}\mathbf{x} \in \mathsf{VALID} \Leftrightarrow T_\pi(\mathbf{x}) \in \mathsf{VALID} \\ \text{If } \mathbf{x} \in \mathsf{VALID} \text{ and } \pi \text{ is uniform in } S \text{ then } T_\pi(\mathbf{x}) \text{ is uniform in } \mathsf{VALID}.\end{array}\right\} \quad (1)$$

A zero knowledge argument protocol should satisfy following three properties:

- Correctness: Honest prover will convince the verifier on proving a true statement. That is the probability of verifier accepting a true statement is 1.
- Soundness: A cheating prover cannot convince the verifier on proving a false statement. Thus the probability of verifier accepting a false statement is negligible. In case of proof of knowledge, soundness is replaced by a stronger notion called "Knowledge Extractor". For every cheating prover there exists an algorithm "Knowledge Extractor" which extracts the witness involved in the interaction between the prover and the verifier.
- Zero knowledge: If the statement proven by the prover is true then the cheating verifier learns only the fact that the statement is true.

A ZKAoK for the relation $\mathcal{R} = \{(\mathbf{P}, \mathbf{v}) \in \mathbb{Z}_q^{D\times L} \times \mathbb{Z}_q^D, \mathbf{x} \in \mathsf{VALID} : \mathbf{Px} = \mathbf{v} \bmod q\}$ is a 3-round protocol ZKAoK=(Commitment, Challenge, Response,Verification) between a prover and a verifier, both having access to $\mathbf{P}$ and $\mathbf{v}$, and works as follows:

1. $\mathsf{ZKAoK.Commitment}(\mathbf{P}, \mathbf{v}, \mathbf{x}) \longrightarrow (\mathsf{COM} = (C_1, C_2, C_3))$. The prover
   (a) samples randomness ρ_1, ρ_2, ρ_3 for generating commitments and $\mathbf{r} \hookleftarrow U(\mathbb{Z}_q^L), \pi \hookleftarrow U(S)$ where S is a finite set of permutation.
   (b) computes the commitment, $\mathsf{COM} = (C_1, C_2, C_3)$ where $C_1 = \mathsf{CMT}_1(\pi, \mathbf{Pr}; \rho_1)$ uses randomness ρ_1, $C_2 = \mathsf{CMT}_2(T_\pi(\mathbf{r}); \rho_2)$ uses randomness ρ_2, $C_3 = \mathsf{CMT}_3(\mathbf{T}_\pi(\mathbf{x} + \mathbf{r}); \rho_3)$ uses randomness ρ_3 and CMT_i, $i = 1, 2, 3$, is a statistically hiding and computationally binding commitment scheme where the hiding property holds even against all-powerful receivers, while the binding property holds only for polynomially bounded senders.
   (c) sends COM to the verifier.
2. $\mathsf{ZKAoK.Challenge}(\mathbf{P}, \mathbf{v}) \longrightarrow (\mathsf{Ch} \hookleftarrow U(\{1, 2, 3\}))$. The verifier sends a challenge $\mathsf{Ch} \hookleftarrow U(\{1, 2, 3\})$ to the prover.
3. $\mathsf{ZKAoK.Response}(\mathsf{Ch}, \rho_1, \rho_2, \rho_3, \pi, T_\pi, \mathbf{r}, \mathbf{x}) \longrightarrow (\mathsf{RSP})$. The prover sends a response RSP computed as follows:
   (a) if Ch= 1, then the prover sets $\mathbf{t_x} = T_\pi(\mathbf{x}), \mathbf{t_r} = T_\pi(\mathbf{r})$ and RSP= $(\mathbf{t_x}, \mathbf{t_r}, \rho_2, \rho_3)$.
   (b) if Ch= 2, then the prover sets $\pi_2 = \pi, \mathbf{y} = \mathbf{x} + \mathbf{r}$ and RSP= $(\pi_2, \mathbf{y}, \rho_1, \rho_3)$.
   (c) if Ch= 3, then the prover sets $\pi_3 = \pi, \mathbf{r}_3 = \mathbf{r}$ and RSP= $(\pi_3, \mathbf{r}_3, \rho_1, \rho_2)$.

   Here ρ_1, ρ_2, ρ_3 are the randomness picked by the prover in generating the commitment CMT using the procedure ZKAoK.Commitment.
4. $\mathsf{ZKAoK.Verification}(\mathbf{P}, \mathbf{v}, \mathsf{RSP}, \mathsf{COM}, \mathsf{CMT}_1, \mathsf{CMT}_2, \mathsf{CMT}_3) \longrightarrow (\mathsf{VRF})$. On receiving the response RSP from the prover, the verifier proceeds as follows:

(a) if $\mathsf{Ch}=1$, then the verifier checks whether $\mathbf{t_x} \in \mathsf{VALID}$ and $C_2 = \mathsf{CMT}_2(\mathbf{t_r}; \rho_2), C_3 = \mathsf{CMT}_3(\mathbf{t_x} + \mathbf{t_r}; \rho_3)$, where C_2, C_3 are extracted from COM and $\mathbf{t_x}, \mathbf{t_r}, \rho_2, \rho_3$ from RSP.

(b) if $\mathsf{Ch}=2$, then the verifier checks whether $C_1 = \mathsf{CMT}_1(\pi_2, \mathbf{Py} - \mathbf{v}; \rho_1), C_3 = \mathsf{CMT}_3(\mathbf{T}_{\pi_2}(\mathbf{y}); \rho_3)$, using C_1, C_3 from COM and $\pi_2, \mathbf{y}, \rho_1, \rho_3$ from RSP.

(c) if $\mathsf{Ch}=3$, then the verifier checks whether $C_1 = \mathsf{CMT}_1(\pi_3, \mathbf{Pr}_3; \rho_1)$, $C_2 = \mathsf{CMT}_2(\mathbf{T}_{\pi_3}(\mathbf{r}_3); \rho_2)$, where C_1, C_2 are obtained from COM and $\pi_3, \mathbf{r}_3, \rho_1, \rho_3$ are from RSP.

In each case, the verifier outputs $\mathsf{VRF} = 1$ if the verification succeeds, otherwise outputs $\mathsf{VRF} = 0$.

Remark 3. *The above protocol is repeated $s = \omega(\log n)$ times to achieve negligible soundness error and can be made non-interactive using Fiat-Shamir heuristic [17] as a triple $\Pi = (\{\mathsf{COM}_\gamma\}_{\gamma=1}^s, \mathsf{Ch}, \{\mathsf{RSP}\}_{\gamma=1}^s)$ where $\mathsf{Ch} = H(M, \{\mathsf{COM}_\gamma\}_{\gamma=1}^s, \mathsf{aux}) \in \{1,2,3\}^s$ where $H : \{0,1\}^* \to \{1,2,3\}^s$ is a hash function and aux is some auxilliary information given as input.*

Theorem 2.2. [25] *The protocol in Figure 2 is a statistical ZKAoK for the relation $\mathcal{R}$ with perfect completeness, soundness error $2/3$, and the communication cost $\mathcal{O}(L \log q)$.*
Theorem 2.3. *(Improved Forking Lemma)*[13] *Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) Turing machine called "Attacker" and a PPT simulator $\mathcal{B}$. Let l, q be real numbers such that $l \leq \frac{\sqrt{q}}{4}$. If $\mathcal{A}$ can find a valid signature (M, σ, h) with non-negligible probability $\epsilon > \frac{4}{q}$ in less than Q_H number of hash queries where $h = H((M, \sigma))$ then with non-negligible constant probability $\frac{1}{96}$ with $\frac{(1+24Q_H l \log(2l))}{\epsilon}$ replays of $\mathcal{A}$ and $\mathcal{B}$ with different random oracles, $\mathcal{A}$ will output $(l+1)$ pair wise distinct valid signatures.*

3 Dynamic Forward Secure Group Signature Scheme (FS-GS)[26]

3.1 Syntax

We describe below the syntax and notion of *dynamic fully forward secure group signature* ($\mathsf{FS\text{-}GS}$) following Kiayias-Yung [22].
The concept of forward secure signature was demonstrated by Anderson in [5] and was characterized formally by Bellare and Miner [8]. A $\mathsf{FS\text{-}GS}$=(Setup, Join, Update, Sign, Verify, Open) scheme is run among a group manager (GM), users ($\mathcal{U}_i$) and an opening authority (OA) where $\mathsf{FS\text{-}GS.Verify}$ and $\mathsf{FS\text{-}GS.Open}$ are determinstic algorithms while $\mathsf{FS\text{-}GS.Setup}$, $\mathsf{FS\text{-}GS.Join}$, $\mathsf{FS\text{-}GS.Update}$, and $\mathsf{FS\text{-}GS.Sign}$ are probabilistic algorithms.

Figure 2: Zero Knowledge Argument of Knowledge

Prover $(\mathbf{P}, \mathbf{v})$ Verifier $(\mathbf{P}, \mathbf{v})$

Secret: $\mathbf{x}$ satisfying $\mathbf{Px} = \mathbf{v}$

Samples: Randomness ρ_1, ρ_2, ρ_3 for COM and
$\mathbf{r} \hookleftarrow U(\mathbb{Z}_q^L)$, $\pi \hookleftarrow U(S)$

Computes: $C_1 = \mathsf{CMT}_1(\pi, \mathbf{Pr}; \rho_1)$
$C_2 = \mathsf{CMT}_2(T_\pi(\mathbf{r}); \rho_2)$
$C_3 = \mathsf{CMT}_3(T_\pi(\mathbf{x} + \mathbf{r}); \rho_3)$
$\mathsf{COM} = (C_1, C_2, C_3)$

$$\xrightarrow{\mathsf{CMT}}$$

$\mathsf{Ch} \hookleftarrow U(\{1, 2, 3\})$

$$\xleftarrow{\mathsf{Ch}}$$

if Ch $= 1$ then RSP $= (t_\mathbf{x} = T_\pi(\mathbf{x}), t_\mathbf{r} = T_\pi(\mathbf{r}), \rho_2, \rho_3)$

if Ch $= 2$ then RSP $= (\pi_2 = \pi, \mathbf{y} = \mathbf{x} + \mathbf{r}, \rho_1, \rho_3)$

if Ch $= 3$ then RSP $= (\pi_3 = \pi, \mathbf{r}_3 = \mathbf{r}, \rho_1, \rho_2)$

$$\xrightarrow{\mathsf{RSP}}$$

if Ch $= 1$ then check whether $t_\mathbf{x} \in$ VALID,
$C_2 = \mathsf{CMT}_2(t_\mathbf{r}; \rho_2), C_3 = \mathsf{CMT}_3(t_\mathbf{x} + t_\mathbf{r}; \rho_3)$

if $\mathsf{Ch} = 2$ then check whether
$C_1 = \mathsf{CMT}_1(\pi_2, \mathbf{Py} - \mathbf{v}; \rho_2), C_3 = \mathsf{CMT}_3(T_{\pi_2}(\mathbf{y}); \rho_3)$

if $\mathsf{Ch} = 3$ then check whether
$C_1 = \mathsf{CMT}_1(\pi_3, \mathbf{Pr}_3; \rho_1), C_2 = \mathsf{CMT}_2(T_{\pi_3}(\mathbf{r}_3); \rho_2)$

In each case, the verifier outputs $\mathsf{VRF} = 1$ if the verification succeeds, otherwise outputs $\mathsf{VRF} = 0$.

(i) $\mathsf{FS\text{-}GS.Setup}(\lambda, T) \longrightarrow (\mathcal{Y}, \mathsf{S_{GM}}, \mathsf{S_{OA}}, \mathsf{St})$. It is run by a trusted third party, called the key generation center (KGC), the KGC given a security parameter $\lambda \in \mathbb{N}$, maximum allowable time period T, outputs the public parameter $\mathcal{Y}$, group manager's secret key $\mathsf{S_{GM}}$ and opening authority's private key $\mathsf{S_{OA}}$. While $\mathsf{S_{GM}}, \mathsf{S_{OA}}$ are given to the respective authorities through secure channels, $\mathcal{Y}$ is publicized. The KGC also initializes a public state, St=($\mathsf{St_{users}}, \mathsf{St_{trans}}$), consisting of a set data structure $\mathsf{St_{users}}$ and a string data structure $\mathsf{St_{trans}}$, initially set as $\mathsf{St_{users}} = \emptyset$ and $\mathsf{St_{trans}} = \epsilon$.

(ii) $\mathsf{FS\text{-}GS.Join}^{(\mathsf{GM},\, \mathcal{U}_i)}(\mathcal{Y}, T) \longrightarrow (\mathsf{cert}_{i,t_1 \to t_2},\ \mathsf{sec}_{i,t_1 \to t_2},\ \mathsf{St})$. It is an interactive protocol between the GM and the user $\mathcal{U}_i$ who wants to be a group member. This protocol manages two Turing machines $\mathsf{J_{user}}$ and the $\mathsf{J_{GM}}$ controlled respectively by $\mathcal{U}_i$ and GM. These Turing machines, on input $\mathcal{Y}, T$, generates a secret key $\mathsf{sec}_{i,t_1 \to t_2}$ and a certificate $\mathsf{cert}_{i,t_1 \to t_2}$ in some time interval $[t_1, t_2] \subset [1, T]$ selected by the GM for each user $\mathcal{U}_i$. Let $[\mathsf{J_{user}}(\mathcal{Y}, T), \mathsf{J_{GM}}(\mathcal{Y}, t_1, t_2, T, \mathsf{S_{GM}}, \mathsf{St})]$ denotes the execution between $\mathcal{U}_i$ and GM after which $\mathcal{U}_i$ receives its certificate $\mathsf{cert}_{i,t_1 \to t_2}$ generated by the GM through a secure communication channel. The secret key $\mathsf{sec}_{i,t_1 \to t_2}$ of $\mathcal{U}_i$ is generated by $\mathcal{U}_i$. On successful completion of the protocol, the user $\mathcal{U}_i$ sets its secret key $\mathsf{sec}_{i,t_1 \to t_2}$ and the GM updates the public state St=($\mathsf{St_{users}}, \mathsf{St_{trans}}$) as $\mathsf{St_{users}} = \mathsf{St_{users}} \cup \{i\}$ and $\mathsf{St_{trans}} = \mathsf{St_{trans}} | \langle i, t_1, t_2, \mathsf{transcript}_i \rangle$,.

(iii) $\mathsf{FS\text{-}GS.Update}(\mathcal{Y}, t, t_2, \mathsf{sec}_{i,t \to t_2}, \mathsf{cert}_{i,t \to t_2}) \longrightarrow (\mathsf{sec}_{i,t+1 \to t_2}, \mathsf{cert}_{i,t+1 \to t_2})$. It is a randomized algorithm executed by the user $\mathcal{U}_i$. On input of $\mathcal{Y}, [t, t_2] \subseteq [1, T]$, a valid membership certificate $\mathsf{cert}_{i,t \to t_2}$ and corresponding membership secret $\mathsf{sec}_{i,t \to t_2}$, this algorithm allows $\mathcal{U}_i$ to obtain its updated membership certificate $\mathsf{cert}_{i,t+1 \to t_2}$ and membership secret $\mathsf{sec}_{i,t+1 \to t_2}$ for the subsequent time period $t + 1$.

(iv) $\mathsf{FS\text{-}GS.Sign}\ (\mathsf{sec}_{i,t_1 \to t_2}, \mathsf{cert}_{i,t_1 \to t_2}, \mathcal{Y}, M) \longrightarrow (\sigma)$. Given $\mathsf{sec}_{i,t_1 \to t_2}, \mathsf{cert}_{i,t_1 \to t_2}, \mathcal{Y}$ and the message M, the user $\mathcal{U}_i$ runs this algorithm and outputs a signature σ on the message M.

(v) $\mathsf{FS\text{-}GS.Verify}(\sigma, t, M, \mathcal{Y}) \longrightarrow (1 \vee 0)$. The verifier, on receiving the message M, signature σ and a time period t, invokes this algorithm using $\mathcal{Y}$ to check whether the produced signature σ is a valid signature on M or not during the time period t. If σ is a valid signature, the verifier returns 1; otherwise outputs 0.

(vi) $\mathsf{FS\text{-}GS.Open}(\sigma, t, M, \mathcal{Y}, \mathsf{S_{OA}}, \mathsf{St}) \longrightarrow (i \vee \perp)$. The opening authority OA with its secret key $\mathsf{S_{OA}}$ along with $\mathcal{Y}, t, \sigma, M$ and St runs this algorithm and returns either the identity i of the group member that has produced the signature σ on M during the time period t or an opening failure $\perp$.

A state St is said to be valid if it can be attained from St=$(\emptyset, \epsilon)$ by a Turing machine having access to the oracle $\mathsf{J_{GM}}$. Every membership certificate includes a unique tag (identity) specific to user generated by the GM .

As in Kiayias-Yung [22], we use notation, $\mathsf{cert}_{i,t_1 \to t_2} \rightleftharpoons_{\mathcal{Y}} \mathsf{sec}_{i,t_1 \to t_2}$ to indicate that there exists some coin tosses ϖ for $\mathsf{J_{GM}}$ run by the GM and $\mathsf{J_{user}}$ run by the user $\mathcal{U}_i$ such that for some valid public state St, the execution of $[\mathsf{J_{user}}(\mathcal{Y}, T), \mathsf{J_{GM}}(\mathcal{Y}, t_1, t_2, T, \mathsf{S_{GM}}, \mathsf{St})](\varpi)$ issues

$\langle i, \mathsf{sec}_{i,t_1\to t_2}, \mathsf{cert}_{i,t_1\to t_2}\rangle$ to $\mathsf{J}_{\mathsf{user}}$ (i.e., to $\mathcal{U}_i$).

Correctness: The correctness of $\mathsf{FS\text{-}GS}$ is as follows:

(a) No two entries of $\mathsf{St}_{\mathsf{trans}}$ should share the same tag and $|\mathsf{St}_{\mathsf{users}}| = |\mathsf{St}_{\mathsf{trans}}|$.

(b) If $\langle i, \mathsf{sec}_{i,t_1\to t_2}, \mathsf{cert}_{i,t_1\to t_2}\rangle$ is obtained by $\mathsf{J}_{\mathsf{user}}$ (i.e., $\mathcal{U}_i$) on honestly running $[\mathsf{J}_{\mathsf{user}}(\mathcal{Y}, T)$, $\mathsf{J}_{\mathsf{GM}}(\mathcal{Y}, t_1, t_2, T, \mathsf{S}_{GM}, \mathsf{St})]$, then $\mathsf{cert}_{i,t_1\to t_2} \rightleftharpoons_{\mathcal{Y}} \mathsf{sec}_{i,t_1\to t_2}$. Here $\mathcal{Y}, \mathsf{S}_{\mathsf{GM}}, \mathsf{S}_{\mathsf{OA}}, \mathsf{St}$ are generated by executing $\mathsf{FS\text{-}GS.Setup}(\lambda, T)$.

(c) Let $\mathsf{cert}_{i,t_1\to t_2} \rightleftharpoons_{\mathcal{Y}} \mathsf{sec}_{i,t_1\to t_2}$ be as in (b) and $\langle i, \mathsf{sec}_{i,t\to t_2}, \mathsf{cert}_{i,t\to t_2}\rangle$ for $t_1 \le t \le t_2$ is obtained by running $\mathsf{FS\text{-}GS.Update}(\mathcal{Y}, k, t_2, \mathsf{sec}_{i,k\to t_2}, \mathsf{cert}_{i,k\to t_2}) \to (\mathsf{sec}_{i,k+1\to t_2}, \mathsf{cert}_{i,k+1\to t_2})$, successively for $k = t_1, t_1+1, ..., t-1$. Then $\mathsf{cert}_{i,t\to t_2} \rightleftharpoons_{\mathcal{Y}} \mathsf{sec}_{i,t\to t_2}$ and $\mathsf{FS\text{-}GS.Verify}(\mathsf{FS\text{-}GS.Sign}(\mathsf{sec}_{i,t_1\to t_2}, \mathsf{cert}_{i,t\to t_2}, \mathcal{Y}, M), t, M, \mathcal{Y}) = 1$.

(d) For any $\langle i, \mathsf{cert}_{i,t_1\to t_2}, \mathsf{sec}_{i,t_1\to t_2}\rangle$ obtained by $\mathcal{U}_i$ on execution of $[\mathsf{J}_{\mathsf{user}}(\mathcal{Y}, t_1, t_2, T)$, $\mathsf{J}_{\mathsf{GM}}(\mathcal{Y}, t_1, t_2, T, \mathsf{S}_{\mathsf{GM}}, \mathsf{St})]$ for some valid state St, let $\langle i, \mathsf{cert}_{i,t\to t_2}, \mathsf{sec}_{i,t\to t_2}\rangle$ be derived by $\mathsf{FS\text{-}GS.Update}$ algorithm as in (c). Now if $\sigma = \mathsf{FS\text{-}GS.Sign}(\mathsf{sec}_{i,t\to t_2}, \mathsf{cert}_{i,t\to t_2}, \mathcal{Y}, M)$, then $\mathsf{FS\text{-}GS.Open}(\sigma, t, M, \mathcal{Y}, \mathsf{S}_{\mathsf{OA}}, \mathsf{St}) = i$.

3.2 Security Model

Security attributes of $\mathsf{FS\text{-}GS}$ scheme are formalized by three experiments (i) mis-identification attack (ii) framing attack and (iii) anonymity attack. An adversary $\mathcal{A}$ invokes several oracles accessible in the attack games and interacts with a stateful interface I that maintains the following variables:

- State_I: a data structure representing the state of the interface I;
- $U = |\mathsf{St}_{\mathsf{users}}|$: the number of currently involved members in the group;
- Sigs: a database of signatures generated by the signing oracle $\mathsf{FS\text{-}GS.Sign}$;
- U^a : collection of adversarially controlled users since their admission;
- U^b : collection of honest users with adversary as the dishonest group manager.

The state of the interface I is $\mathsf{State}_I = (\mathsf{State}_{\mathsf{pri}}, \mathsf{State}_{\mathsf{pub}})$ where $\mathsf{State}_{\mathsf{pri}} = (\mathsf{cert}_{i,t_1\to t_2}, \mathsf{S}_{\mathsf{GM}}, \mathsf{S}_{\mathsf{OA}})$ and $\mathsf{State}_{\mathsf{pub}} = (\mathsf{St}_{\mathsf{users}}, \mathsf{St}_{\mathsf{trans}}, \mathcal{Y}, \mathsf{St}_{\mathsf{corr}})$ where $\mathcal{Y}, \mathsf{S}_{\mathsf{GM}}, \mathsf{S}_{\mathsf{OA}}$ and $\mathsf{St} = (\mathsf{St}_{\mathsf{users}}, \mathsf{St}_{\mathsf{trans}})$ are generated by I on invoking $\mathsf{FS\text{-}GS.Setup}(\lambda, T)$ and $\mathsf{St}_{\mathsf{corr}}$ store pairs of the form (i, t) with $i \in \mathsf{St}_{\mathsf{users}}$ and $t \in \{1, 2, ..., T\}$ representing i is corrupted (i.e., $i \in \mathcal{U}^b$) at time period t. Each record stored in Sigs is of the form (i, t, M, σ) indicates the message M is signed by $\mathcal{U}_i$ during the period t. The adversary has access to the transcript $\mathsf{St}_{\mathsf{trans}}$ of the $\mathsf{FS\text{-}GS.Join}$ protocol but not to user's membership secret for users in U^b.

The adversary $\mathcal{A}$ is given access to the following oracles while mounting attacks.

- $\mathsf{Q}_{\mathsf{pub}}, \mathsf{Q}_{\mathsf{keyGM}}, \mathsf{Q}_{\mathsf{keyOA}}$. When adversary $\mathcal{A}$ invokes the oracles $\mathsf{Q}_{\mathsf{pub}}, \mathsf{Q}_{\mathsf{keyGM}}, \mathsf{Q}_{\mathsf{keyOA}}$, the interface I with State_I returns $\mathcal{Y}, \mathsf{S}_{\mathsf{GM}}, \mathsf{S}_{\mathsf{OA}}$ respectively to $\mathcal{A}$ by extracting them from State_I.

- $\mathsf{Q}_{\mathsf{a-join}}$. The adversary $\mathcal{A}$ can introduce users under its control (users in U^a) by querying the oracle $\mathsf{Q}_{\mathsf{a-join}}$. On input $t_1, t_2 \in \{1, 2, .., T\}$, the interface I executes $\mathsf{FS\text{-}GS.Join}$ acting as the GM that controls the Turing machine J_{GM} and interacts with a malicious user $\mathcal{U}_i$ that controls the Turing machine $\mathsf{J}_{\mathsf{user}}$. If the protocol is successful, then the interface I increments U by 1, and updates St=$(\mathsf{St}_{\mathsf{users}}, \mathsf{St}_{\mathsf{trans}})$ adding index i of the malicious user $\mathcal{U}_i$ to both U^a and $\mathsf{St}_{\mathsf{users}}$, and updating $\mathsf{St}_{\mathsf{trans}}$ as $\mathsf{St}_{\mathsf{trans}} = \mathsf{St}_{\mathsf{trans}} | \langle i, t_1, t_2, \mathsf{transcript}_i \rangle$.

- $\mathsf{Q}_{\mathsf{b-join}}$. This query enables the adversary $\mathcal{A}$ to introduce honest users while acting as the dishonest GM. The adversary $\mathcal{A}$ acts as the GM and handles the Turing machine J_{GM} in executing algorithm $\mathsf{FS\text{-}GS.Join}$ while the Turing machine $\mathsf{J}_{\mathsf{user}}$ is controlled by an honest user $\mathcal{U}_i$, who wants to be a member. If the protocol successfully terminates, then the interface I increments U by 1, adds user index i to both U^b and $\mathsf{St}_{\mathsf{users}}$, and updates $\mathsf{St}_{\mathsf{trans}}$ as $\mathsf{St}_{\mathsf{trans}} = \mathsf{St}_{\mathsf{trans}} | \langle i, t_1, t_2, \mathsf{transcript}_i \rangle$. The Interface I finally stores the membership certificate $\mathsf{cert}_{i,t_1 \to t_2}$ in a private part of State_I.

- $\mathsf{Q}_{\mathsf{sig}}(M, i, t) \longrightarrow (\sigma \vee \perp)$. On querying a tuple (M, i, t) consisting of a message M, an index i, and a time period t, the interface I first checks if $\mathsf{State}_{\mathsf{pri}}$ contains $\mathsf{sec}_{i,t_1 \to t_2}, \mathsf{cert}_{i,t_1 \to t_2}$ for some $t_1, t_2 \in \{1, 2, ..., T\}$ with $t_1 \leq t \leq t_2$ and $\mathcal{U}_i \in U^b$. If so, I calls $\mathsf{FS\text{-}GS.Update}(\mathcal{Y}, k, t_2, \mathsf{sec}_{i,k \to t_2}, \mathsf{cert}_{i,k \to t_2})$ for $k = t_1, t_1{+}1, ..., t$ to generate the pair $(\mathsf{cert}_{i,t \to t_2}, \mathsf{sec}_{i,t \to t_2})$, returns a signature $\sigma \longleftarrow \mathsf{FS\text{-}GS.Sign}(\mathsf{sec}_{i,t \to t_2}, \mathsf{cert}_{i,t \to t_2}, \mathcal{Y}, M)$ to $\mathcal{A}$ on behalf of user $\mathcal{U}_i$ for the period t and updates Sigs=$\mathsf{Sigs} | \langle i, t, M, \sigma \rangle$. Otherwise, the interface I returns $\perp$ to $\mathcal{A}$.

- $\mathsf{Q}_{\mathsf{open}}(M, \sigma, t) \longrightarrow (i \vee \perp)$. Given a valid tuple (M, σ, t) from the adversary, the interface I runs the opening algorithm $\mathsf{FS\text{-}GS.Open}(\sigma, t, M, \mathcal{Y}, \mathsf{S}_{\mathsf{OA}}, \mathsf{St}) \longrightarrow (i \vee \perp)$ with the current state St=$(\mathsf{St}_{\mathsf{users}}, \mathsf{St}_{\mathsf{trans}})$. Let $S = \{(M, \sigma, t) \mid \mathsf{FS\text{-}GS.Verify}(\sigma, t, M, \mathcal{Y}) = 1\}$ and $\mathsf{Q}_{\mathsf{open}}^{\neg \mathsf{S}}$ denotes a restricted oracle that applies opening algorithm $\mathsf{FS\text{-}GS.Open}$ only for the valid tuples (M, σ, t) which are not in S.

- $\mathsf{Q}_{\mathsf{read}}$ and $\mathsf{Q}_{\mathsf{write}}$. These queries permit the adversary $\mathcal{A}$ to read and write the contents in $\mathsf{State}_I = (\mathsf{State}_{\mathsf{pri}}, \mathsf{State}_{\mathsf{pub}})$. By $\mathsf{Q}_{\mathsf{read}}$ queries, the adversary $\mathcal{A}$ can read State_I but cannot read $\mathsf{State}_{\mathsf{pri}}$ of State_I where membership secrets are stored after $\mathsf{Q}_{\mathsf{b-join}}$ queries. On the other hand, $\mathsf{Q}_{\mathsf{write}}$ query allows the adversary $\mathcal{A}$ to update State_I by introducing users in U^b without altering, removing or reusing the already existing certificates.

- $\mathsf{Q}_{\mathsf{corrupt}}(i, t) \longrightarrow ((\mathsf{cert}_{i,t \to t_2}, \mathsf{sec}_{i,t \to t_2}) \vee \perp)$. On receiving query (i, t), where $i \in \mathsf{St}_{\mathsf{users}}$ and $t \in \{1, 2, ..., T\}$ from $\mathcal{A}$, the interface I checks if $i \in U^b$ and $\mathsf{St}_{\mathsf{trans}}$ has a record of the form $\langle i, t_1, t_2, \mathsf{transcript}_i \rangle$ for some $t_1, t_2 \in \{1, 2, ..., T\}$ with $t_1 \leq t \leq t_2$. If not, returns $\perp$. Else, I extracts $\mathsf{cert}_{i,t_1 \to t_2}$, $\mathsf{sec}_{i,t_1 \to t_2}$ from $\mathsf{State}_{\mathsf{pri}}$ and iteratively call algorithm $\mathsf{FS\text{-}GS.Update}(\mathcal{Y}, k, t_2, \mathsf{sec}_{i,k \to t_2}, \mathsf{cert}_{i,k \to t_2})$ for $k = t_1, t_1 + 1, ..., t - 1$ to generate $\mathsf{cert}_{i,t \to t_2}$ and $\mathsf{sec}_{i,t \to t_2}$ for $t > t_1$. It provides all these information to the adversary and stores (i, t) in $\mathsf{St}_{\mathsf{corr}}$. Once a user gets corrupted, it remains corrupted thereafter.

The security against mis-identification, framing and anonymity attacks are formally described below:

1. Mis-identification Attack: This attack is modeled by the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{mis-id}}(\lambda, \mathsf{T})$ described in Algorithm 2 between an adversary $\mathcal{A}$ and the interface I. In this attack, $\mathcal{A}$ gets the public parameter $\mathcal{Y}$ by $\mathsf{Q}_{\mathsf{pub}}$ query and $\mathsf{State}_{\mathsf{pub}}$ by $\mathsf{Q}_{\mathsf{read}}$ query. Besides, $\mathcal{A}$ has the power to manipulate the opening authority given access to S_{OA} by querying $\mathsf{Q}_{\mathsf{keyOA}}$ oracle. It can also launch new dishonest members in the group by making $\mathsf{Q}_{\mathsf{a-join}}$ queries whereby St gets updated and $\mathcal{A}$ receives all the secret information of the dishonest members. The aim of the adversary $\mathcal{A}$ is to produce a valid signature σ^* on a message M^* during the time period t^* that does not belong to any adversarially controlled member (i.e., of U^a) endowed with the ability to sign during the period t^*.

   Definition 6. *A* FS-GS *scheme over T periods is secure against mis-identification attack if* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mis-id}}(\lambda, T) = Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{mis-id}}(\lambda, T) = 1] \in \mathsf{negl}(\lambda)$*, where* negl *is a negligible function in λ.*

Algorithm 2: $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{mis-id}}(\lambda, T)$

```
$\mathsf{State}_I = (\mathsf{St}, \mathcal{Y}, \mathsf{S}_{\mathsf{GM}}, \mathsf{S}_{\mathsf{OA}}) \leftarrow \mathsf{FS\text{-}GS.Setup}(\lambda, T)$;
$(M^*, t^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{Q}_{\mathsf{pub}}, \mathsf{Q}_{\mathsf{a-join}}, \mathsf{Q}_{\mathsf{read}}, \mathsf{Q}_{\mathsf{keyOA}})$;
if $(\mathsf{FS\text{-}GS.Verify}(\sigma^*, t^*, M, \mathcal{Y}) = 0)$ then
    return $0$;
  else
      $i = \mathsf{FS\text{-}GS.Open}(\sigma^*, t^*, M^*, \mathcal{Y}, \mathsf{S}_{\mathsf{OA}}, \mathsf{St})$;
        if $(i \notin U^a)$ then
          return $1$;
        else
            if $([\langle i, t_1, t_2, \mathsf{transcript}_i\rangle \in \mathsf{St}_{\mathsf{trans}}] \wedge [t_1 \le t^* \le t_2])$ then
              return $0$;
             else return $1$;
            end if
        end if
end if
```

2. Framing Attack: This attack game is formally described by the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{fra}}(\lambda)$ in Algorithm 3 which is played between an adversary $\mathcal{A}$ and the interface I. The adversary $\mathcal{A}$ has given the power to make the whole system including the group manager and the opening authority both colludes against some honest user. In this attack, the adversary $\mathcal{A}$ is capable to access the secret keys of both the GM and the OA. Acting as a dishonest group manager, $\mathcal{A}$ can introduce honest members using $\mathsf{Q}_{\mathsf{join}}$ queries. At the same moment, $\mathcal{A}$ can corrupt the honest members of U^b by invoking $\mathsf{Q}_{\mathsf{corrupt}}$ queries. It also has the power to observe the system when user produce signatures with $\mathsf{Q}_{\mathsf{sig}}$ queries, and can create dummy users with $\mathsf{Q}_{\mathsf{write}}$ queries. The adversary's goal is to (a) either frame an uncorrupted group member or (b) generate a signature that opens to some corrupt member for a period proceeding the one where that member was not

a group member.

Definition 7. *A* FS-GS *scheme over T periods is secure against framing attack if* $\mathsf{Adv}^{\mathsf{fra}}_{\mathcal{A}}(\lambda) = Pr[\mathsf{Exp}^{\mathsf{fra}}_{\mathcal{A}}(\lambda) = 1] \in \mathsf{negl}(\lambda)$*, where* negl *is a negligible function in* λ.

Algorithm 3: $\mathsf{Exp}^{\mathsf{fra}}_{\mathcal{A}}(\lambda)$

```
$\mathsf{State_I} = (\mathsf{St}, \mathcal{Y}, \mathsf{S_{GM}}, \mathsf{S_{OA}}) \leftarrow \mathsf{FS\text{-}GS.Setup}(\lambda, \mathsf{T})$;
$(M^*, t^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{Q_{pub}}, \mathsf{Q_{keyGM}}, \mathsf{Q_{b-join}}, \mathsf{Q_{sig}}, \mathsf{Q_{corrupt}}, \mathsf{Q_{write}}, \mathsf{Q_{read}}, \mathsf{Q_{keyOA}})$;
if $(\mathsf{FS - GS.Verify}(\sigma^*, t^*, M, \mathcal{Y}) = 0)$ then
    return 0;
else
    $i = \mathsf{FS\text{-}GS.Open}(\sigma^*, t^*, M^*, \mathcal{Y}, \mathsf{S_{OA}}, \mathsf{St})$;
    if($[\nexists\langle i, t_1, t_2, \mathsf{transcript}_i\rangle \in \mathsf{St_{trans}}$ such that $(t_1 \leq t^* \leq t_2)]$
            $\vee\ [\exists(i, t') \in \mathsf{St_{corr}}$ such that $t' \leq t^*])$ then
        return 0;
    else
        if($\wedge_{(j \in U^b \text{ such that } j=i)}(j, t^*, M^*, *) \notin \mathsf{Sigs}$) then
            return 1;
        else return 0;
        end if
    end if
end if
```

3. Anonymity Attack: The formal notion of this attack game is modeled by the experiment $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}(\lambda, T)$ outlined in Algorithm 4 between an adversary $\mathcal{A}$ and the interface I. It consists of 2 stages. The first stage is the play stage, where the adversary $\mathcal{A}$ is allowed to make $\mathsf{Q_{write}}$ queries and has access to $\mathsf{Q_{open}}$ oracle. At the end of this stage, $\mathcal{A}$ chooses (M^*, t^*) and two pairs $(\mathsf{sec}_{i_0,t^*\to t^*_0}, \mathsf{cert}_{i_0,t^*\to t^*_0}), (\mathsf{sec}_{i_1,t^*\to t^*_1}, \mathsf{cert}_{i_1,t^*\to t^*_1})$, consisting of a well formed membership secret and a membership certificate for periods $[t^*, t^*_{b,2}]$ for $b \in \{0, 1\}$, and auxiliary information aux. The interface generates a signature σ^* using $(\mathsf{sec}_{i_d,t^*\to t^*_d}, \mathsf{cert}_{i_d,t^*\to t^*_d})$ by flipping a fair coin $d \in \{0, 1\}$. The aim of the adversary $\mathcal{A}$ is to output a guess d' for the guess stage.

Definition 8. *A* FS-GS *scheme is fully anonymous for any PPT adversary* $\mathcal{A}$*, if* $\mathsf{Adv}^{\mathsf{anon}}(\mathcal{A}) := |Pr[\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}(\lambda) = 1] - 1/2| \in \mathsf{negl}(\lambda)$*, where* negl *is a negligible function in* λ.

4 Description of Our FSGS Scheme

- $\mathsf{FSGS.Setup}(\lambda, T) \longrightarrow (\mathcal{Y}, \mathsf{S_{GM}}, \mathsf{S_{OA}}, \mathsf{St})$. Given a security parameter $\lambda > 0$ and maximum allowable time period $T = 2^l$ for some integer l, the KGC chooses an integer n of size $\mathcal{O}(\lambda)$, a prime modulus q of size $\widetilde{\mathcal{O}}(ln^3)$ such that $T \leq q$ and an integer

Algorithm 4: $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A}}(\lambda)$

```
$\mathsf{State}_I = (\mathsf{St}, \mathcal{Y}, \mathsf{S_{GM}}, \mathsf{S_{OA}}) \leftarrow \mathsf{FS\text{-}GS.Setup}(\lambda, \mathsf{T})$;
$\big(\mathsf{aux}, M^*, t^*, (\mathsf{sec}_{i_0,t^*\to t_0^*}, \mathsf{cert}_{i_0,t^*\to t_0^*}), (\mathsf{sec}_{i_1,t^*\to t_1^*}, \mathsf{cert}_{i_1,t^*\to t_1^*})\big)$
                    $\leftarrow \mathcal{A}(\mathsf{Play}\text{: } \mathsf{Q_{pub}}, \mathsf{Q_{keyGM}}, \mathsf{Q_{open}}, \mathsf{Q_{read}}, \mathsf{Q_{write}})$;
if $([\neg(\mathsf{cert}_{i_b,t^*\to t_b^*} \rightleftharpoons_{\mathcal{Y}} \mathsf{sec}_{i_b,t^*\to t_b^*})$ where $b \in \{0,1\}] \vee [\mathsf{cert}_{i_0,t^*\to t_0^*} = \mathsf{cert}_{i_1,t^*\to t_1^*}])$ then
    return $0$;
else
    $d \leftarrow U\{0,1\}$;
    $\sigma^* \leftarrow \mathsf{FS\text{-}GS.Sign}(\mathsf{sec}_{i_d,t^*\to t_d^*}, \mathsf{cert}^*_{i_d,t^*\to t_d^*}, \mathcal{Y}, M^*)$;
    $d' \leftarrow \mathcal{A}(\mathsf{guess}, \sigma^*, \mathsf{aux} : \mathsf{Q_{pub}}, \mathsf{Q_{keyGM}}, \mathsf{Q_{open}}^{\neg\{(\mathsf{M}^*,\sigma^*,\mathsf{t}^*)\}}, \mathsf{Q_{read}}, \mathsf{Q_{write}})$;
    if $(d' = d)$ then
        return $1$;
    else  return $0$;
    end if
end if
```

μ with $\mu \geq \log l$. The system supports atmost $N = 2^\mu$ members. The KGC sets $m = 2n\lceil \log q \rceil$, selects real numbers σ of size $\Omega(\sqrt{n \log q} \log n)$, β of size $\sigma\omega(\log m)$ and γ of size $\sqrt{n}\omega(\log n)$. Here $\mathcal{O}, \Omega, \omega$ are the standard asymptotic notations and $\tilde{\mathcal{O}}$ is as defined in Section 2. The KGC executes the following steps to generate the public key $\mathcal{Y}$, and the private keys $\mathsf{S_{OA}}$ of the opening authority (OA) and $\mathsf{S_{GM}}$ of the group manager (GM) respectively and initializes the state St.

(i) It chooses random matrices $\mathbf{M}_0, \mathbf{M}_1, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{D} \hookleftarrow U(\mathbb{Z}_q^{n\times m})$, $\mathbf{D}_0, \mathbf{D}_1 \hookleftarrow U(\mathbb{Z}_q^{2n\times 2m})$, $\mathbf{F} \hookleftarrow U(\mathbb{Z}_q^{4n\times 4m})$, and a vector $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^n)$. It also selects three quantum secure hash functions which will be modeled as random oracle in the security analysis $H : \{0,1\}^* \to \{1,2,3\}^s$, $H_0 : \{0,1\}^* \to \mathbb{Z}_q^{n\times 2m}$ and $H_2 : \{0,1\}^* \to \mathcal{I}$, where s is an integer of size $\omega(\log n)$, $\mathcal{I}$ is a set of $m \times m$ invertible matrices over $\mathbb{Z}_q$ with columns having low norm and also picks a one time signature $\mathsf{OTS}{=}(\mathcal{G}, \mathcal{S}, \mathcal{V})$, where $\mathcal{G}, \mathcal{S}, \mathcal{V}$ are the key generation, signing and verification algorithms respectively.

(ii) The KGC runs the key generation algorithm $\mathsf{DSig.KeyGen}$ of a digital signature scheme $\mathsf{DSig}{=}(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ and generates signing-verification key pair $(\mathsf{usk}[i], \mathsf{uvk}[i])$ for each user $\mathcal{U}_i$, $i \in [N]$. We assume that uvk table is public and anyone can access the genuine copy of the verification key of any user.

(iii) The KGC runs $\mathsf{TrapGen}(1^n, 1^m, q) \longrightarrow (\mathbf{A}, \mathsf{T}_\mathbf{A})$ and $\mathsf{TrapGen}(1^n, 1^m, q) \longrightarrow (\mathbf{B}, \mathsf{T}_\mathbf{B})$ as described in Lemma 1.1$(b)(iii)$ in Section 2 and outputs matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n\times m}$ with short bases $\mathsf{T}_\mathbf{A}$ of $\Lambda_q^\perp(\mathbf{A})$ and $\mathsf{T}_\mathbf{B}$ of $\Lambda_q^\perp(\mathbf{B})$. The short basis $\mathsf{T}_\mathbf{A}$ is the secret key $\mathsf{S_{GM}}$ of the GM utilized to issue certificates to the new users while $\mathsf{T}_\mathbf{B}$ is the master secret key $\mathsf{S_{OA}}$ of the OA employed to trace the signer.

(iv) Initializes the public state $\mathsf{St} = (\mathsf{St_{users}}, \mathsf{St_{trans}}) = (\emptyset, \varepsilon)$.

(v) The KGC publishes the group public key

$$\mathcal{Y} = (\mathbf{M}_0, \mathbf{M}_1, \mathbf{A}, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{B}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}, \mathsf{OTS}, \mathsf{DSig}, H, H_0, \beta, \gamma, \sigma)$$

and sends $\mathsf{S_{OA}} = \mathsf{T_{\mathbf{B}}}$ to the OA and $\mathsf{S_{GM}} = \mathsf{T_{\mathbf{A}}}$ to the GM through a secure communication channel.

```
chooses $\lambda, l, n, \mu, q$
sets $m, \sigma, \beta, \gamma, N, T$
chooses $\mathbf{M}_0, \mathbf{M}_1, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}$
selects $H, H_0, \mathsf{OTS}, \mathsf{DSig}$
generates $(\mathbf{A}, \mathsf{T_{\mathbf{A}}}) \longleftarrow \mathsf{TrapGen}(1^n, 1^m, q)$
generates $(\mathbf{B}, \mathsf{T_{\mathbf{B}}}) \longleftarrow \mathsf{TrapGen}(1^n, 1^m, q)$
sets $\mathsf{S_{GM}} = \mathsf{T_{\mathbf{A}}}$ , $\mathsf{S_{OA}} = \mathsf{T_{\mathbf{B}}}$
sends $\mathsf{S_{GM}} = \mathsf{T_{\mathbf{A}}}$ to the GM and $\mathsf{S_{OA}} = \mathsf{T_{\mathbf{B}}}$ to the OA
sets $\mathsf{St} = (\mathsf{St_{users}}, \mathsf{St_{trans}}) = (\emptyset, \varepsilon)$

publishes $\mathcal{Y} = (\mathbf{M}_0, \mathbf{M}_1, \mathbf{A}, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{B}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}, \mathsf{OTS}, \mathsf{DSig}, H, H_0, \beta, \gamma, \sigma)$
```

- $\mathsf{FSGS.Join}$ $(\mathcal{Y}, T)^{(\mathsf{GM}, \mathcal{U}_i)} \longrightarrow (\mathsf{cert}_{i,t_1 \to t_2},\ \mathsf{sec}_{i,t_1 \to t_2},\ \mathsf{St})$. It is an interactive protocol run between the GM and the user $\mathcal{U}_i$. The GM and the user $\mathcal{U}_i$ runs the Turing machines $\mathsf{J_{GM}}$ and $\mathsf{J_{user}}$ respectively. On completion of the protocol, $\mathcal{U}_i$ receives its membership certificate $\mathsf{cert}_{i,t_1 \to t_2}$ from the GM through a secure communication channel for the time interval $[t_1, t_2]$ and fixes its secret key $\mathsf{sec}_{i,t_1 \to t_2}$ while the GM updates the public state St=$(\mathsf{St_{users}}, \mathsf{St_{trans}})$ (initially set to be $(\emptyset, \varepsilon)$ by the KGC) as follows.

  (i) The user $\mathcal{U}_i$ samples $\mathbf{z}_i^{(0)} \hookleftarrow D_{\mathbb{Z}^{4m},\sigma}$ meaning $\mathbf{z}_i^{(0)}$ is an integer column vector of length $4m$ chosen from discrete Gaussian with standard deviation σ. The user $\mathcal{U}_i$ computes $\mathbf{v}_i^{(0)} = \mathbf{F} \cdot \mathbf{z}_i^{(0)} \mod q$, $\mathsf{sig}_i = \mathsf{DSig.Sign}(\mathsf{usk}[i], \mathbf{v}_i^{(0)})$ and sends $(\mathbf{v}_i^{(0)}, \mathsf{sig}_i)$ to the GM.

  (ii) The GM runs $\mathsf{DSig.Verify}(\mathsf{uvk}[i], \mathsf{sig}_i, \mathbf{v}_i^{(0)})$. If the verification fails then the GM aborts. Otherwise, if $\mathbf{v}_i^{(0)}$ is not previously used by a registered member, then the GM chooses a μ-bit identity $\mathsf{id}_i = (\mathsf{id}_i[1], \mathsf{id}_i[2], ..., \mathsf{id}_i[\mu]) \in \{0,1\}^{\mu}$ and a time interval $[t_1, t_2] \subset [1, T]$ for user $\mathcal{U}_i$ and uses the secret key $\mathsf{S_{GM}} = \mathsf{T_{\mathbf{A}}}$ to issue a new certificate to $\mathcal{U}_i$ as follows:

    – The GM computes the matrix

    $$\mathbf{A}_{\mathsf{id}_i} = [\mathbf{A}|\bar{\mathbf{A}}] \in \mathbb{Z}_q^{n \times 2m}, \text{ with } \bar{\mathbf{A}} = \mathbf{A}_0 + \sum_{k=1}^{\mu} \mathsf{id}_i[k]\mathbf{A}_k$$

    where $\mathbf{A}$, $\{\mathbf{A}_k\}_{k=0}^{\mu}$ are extracted from the public key $\mathcal{Y}$, and $\mathsf{id}_i[k]$ indicates the kth bit of the identity id_i of user $\mathcal{U}_i$.

    – It runs $\mathsf{ExtBasis}(\mathbf{A}, \bar{\mathbf{A}}, \mathsf{T_{\mathbf{A}}}) \longrightarrow (\mathsf{T}_{\mathbf{A}|\bar{\mathbf{A}}} = \mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}})$ as in Lemma 1.1(b)(iv) of Section 2 to generate a short basis $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ of $\Lambda_q^{\perp}(\mathbf{A}_{\mathsf{id}_i})$.

– The GM samples a short vector $\mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$. Note that by Lemma 1.1(a) in Section 2, $||\mathbf{s}_i|| \leq \sqrt{2m}\sigma$. As $||\mathbf{x}||_\infty \leq ||\mathbf{x}|| \leq \sqrt{n}||\mathbf{x}||_\infty$ for any $\mathbf{x} \in \mathbb{R}^n$, we have $||\mathbf{s}_i||_\infty \leq \sigma\omega(\log m) = \beta$.

– By using the short basis $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ of $\Lambda_q^\perp(\mathbf{A}_{\mathsf{id}_i})$ as mentioned in Remark 1 in Section 2, the GM computes a short vector $\mathbf{d}_i = [\mathbf{d}_{i,1}|\mathbf{d}_{i,2}]^t \in \mathbb{Z}^{2m}$ with $||\mathbf{d}_i|| \leq \beta$, $\mathbf{d}_{i,1}, \mathbf{d}_{i,2} \in \mathbb{Z}^m$, satisfying

$$\mathbf{A}_{\mathsf{id}_i}\mathbf{d}_i = \mathbf{u} + \mathbf{D} \cdot \mathbf{r}_i \mod q \tag{2}$$

where

$$\mathbf{r}_i = \mathsf{bin}(\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_i^{(0)}) + \mathbf{D}_1 \cdot \mathbf{s}_i) \mod q \tag{3}$$

Here $\mathsf{bin}(\mathbf{v}_i^{(0)})$ stands for the binary representation of the vector $\mathbf{v}_i^{(0)}$. Observe that $\mathbf{D}_1 \cdot \mathbf{s}_i \in \mathbb{Z}^{2n\times 1}$ and $\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_i^{(0)}) \in \mathbb{Z}_q^{2n\times 1}$ as $\mathsf{bin}(\mathbf{v}_i^{(0)})$ is of length $4n\lceil \log q \rceil = 2m$. We also point out that $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ is computable only by the GM, as $\mathsf{T}_{\mathbf{A}}$ is known only to the GM and the μ-bit identifier id_i for user $\mathcal{U}_i$ is selected by the GM itself. Thus, the GM can compute the short vector $\mathbf{d}_i$ with $||\mathbf{d}_i|| \leq \beta$.

– The GM then runs the algorithm $\mathsf{Nodes}(t_1+1, t_2, G)$ to identify the collection of subset of hanging nodes $\mathsf{SubsetHN}$ as explained in Section 2.2 where G is the complete binary tree of height l with $T = 2^l$ leaves each indicating a time period. For each node $w = (w_{\mathsf{num}}, w_{\mathsf{dep}}) \in \mathsf{SubsetHN}$ where $w_{\mathsf{num}} \in \{0,1\}^*$ is the label of the node w and $w_{\mathsf{dep}} \in \mathbb{Z}$, $0 \leq w_{\mathsf{dep}} \leq l$, is its depth in the tree G of height l, the GM does the following:

(a) computes the matrix

$$\mathbf{A}_{i,w} = [\mathbf{A}_{\mathsf{id}_i}|\bar{\mathbf{A}}_w] \text{ with } \bar{\mathbf{A}}_w = [\mathbf{M}_0 + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1|\mathbf{A}_0 + \sum_{k=1}^{\mu}\alpha_k\mathbf{A}_k]$$

where $\alpha_1\alpha_2...\alpha_\mu = 0^{\mu-\delta}||\mathsf{bin}(w_{\mathsf{dep}})$, $\alpha_k \in \{0,1\}$, $\delta = \mathsf{len}(\mathsf{bin}(w_{\mathsf{dep}}))$ denotes the length of $\mathsf{bin}(w_{\mathsf{dep}})$ and $\mathsf{wt}(w_{\mathsf{num}})$ represents the Hamming weight of w_{num};
Note that $\mathsf{wt}(w_{\mathsf{num}})$ and w_{dep} helps to generate different matrices $\mathbf{A}_w$ for each $w \in \mathsf{SubsetHN}$. By our choice of labeling, Hamming weights are different at the same level so in this case w_{num} ensures for different matrices whereas w_{dep} guarantees to get different matrices each time when Hamming weights are same at different depth.

(b) executes $\mathsf{ExtBasis}(\mathbf{A}_{\mathsf{id}_i}, \bar{\mathbf{A}}_w, \mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}) \longrightarrow (\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}|\bar{\mathbf{A}}_{i,w}} = \mathsf{T}_{\mathbf{A}_w})$ to generate a short basis $\mathsf{T}_{\mathbf{A}_{i,w}}$ of $\Lambda_q^\perp(\mathbf{A}_{i,w})$;

(c) samples short vector $\mathbf{s}_{i,w}^{(0)} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$, i.e., $||\mathbf{s}_{i,w}^{(0)}|| \leq \sqrt{2m}\sigma$ with high probability whereby $||\mathbf{s}_{i,w}^{(0)}||_\infty \leq \sigma\omega(\log m) = \beta$ as guaranteed by Lemma 1.1(a).

(d) computes a short vector $\mathbf{x}_{i,w}^{(0)} = [\mathbf{x}_{i,w,1}^{(0)}|\mathbf{x}_{i,w,2}^{(0)}|\mathbf{x}_{i,w,3}^{(0)}]^t$ with $||\mathbf{x}_{i,w}^{(0)}||_\infty \leq \beta$ satisfying

$$\mathbf{A}_{i,w}\mathbf{x}_{i,w}^{(0)} = \mathbf{u} + \mathbf{D} \cdot \mathsf{bin}(\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_i^{(0)}) + \mathbf{D}_1 \cdot \mathbf{s}_{i,w}^{(0)}) \mod q \tag{4}$$

by using the short basis $\mathsf{T}_{\mathbf{A}_w}$ of $\Lambda_q^\perp(\mathbf{A}_w)$ as stated in Remark 1 of Section 2 where $\mathbf{x}_{i,w,1}^{(0)} \in \mathbb{Z}^{2m}$ and $\mathbf{x}_{i,w,2}^{(0)}, \mathbf{x}_{i,w,3}^{(0)} \in \mathbb{Z}^m$. Note that the knowledge of $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ is required to compute $\mathsf{T}_{\mathbf{A}_w}$ and knowledge of $\mathsf{T}_{\mathbf{A}}$ is required to compute $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ using the algorithm $\mathsf{ExtBasis}$. As the GM knows the short basis $\mathsf{T}_{\mathbf{A}}$, it can generate a short vector $\mathbf{x}_{i,w}^{(0)}$ with $||\mathbf{x}_{i,w}^{(0)}||_\infty \leq \beta$.

– The GM computes $H_2(\mathsf{id}_i||0) = \mathbf{R}_i^{(0)}$, where $\mathbf{R}_i^{(0)}$ is a $\mathbb{Z}_q$- invertible matrix of size $m \times m$ whose columns have low norm, sets $\mathbf{C}_i^{(0)} = \mathbf{A}(\mathbf{R}_i^{(0)})^{-1}$, and runs the delegation algorithm $\mathsf{BasisDel}(\mathbf{A}, \mathbf{R}_i^{(0)}, \mathsf{T}_{\mathbf{A}}, \sigma) \longrightarrow (\mathbf{C}_i^{(0)}, \mathsf{T}_{\mathbf{C}_i^{(0)}})$ as in Lemma 1.1$(b)(vi)$ in Section 2 to generate a short basis $\mathsf{T}_{\mathbf{C}_i^{(0)}}$ of $\Lambda_q^\perp(\mathbf{C}_i^{(0)})$.

(iii) The GM finally issues the membership certificate

$$\mathsf{cert}_{i,t_1\to t_2} = \Big(\mathsf{id}_i, \mathbf{d}_i, \mathbf{s}_i, \{\mathbf{x}_{i,w}^{(0)}, \mathbf{s}_{i,w}^{(0)}\}_{w \in \mathsf{SubsetHN} \leftarrow \mathsf{Nodes}(t_1+1,t_2,G)}, \mathbf{C}_i^{(0)}, \mathsf{T}_{\mathbf{C}_i^{(0)}}, [t_1, t_2]\Big)$$

to the user $\mathcal{U}_i$ through a secure communication channel. The GM updates the public state St by storing i in $\mathsf{St}_{\mathsf{users}}$ and $\mathsf{transcript}_i = (\mathbf{v}_i^{(0)}, i, \mathsf{uvk}[i], \mathsf{sig}_i, [t_1, t_2])$ in $\mathsf{St}_{\mathsf{trans}}$.

(iv) The user $\mathcal{U}_i$ verifies $||\mathbf{d}_i||_\infty \leq \beta$, $||\mathbf{s}_i||_\infty \leq \beta$ satisfying Eq 2 and for all $w \in \mathsf{SubsetHN} \longleftarrow \mathsf{Nodes}(t_1+1, t_2, G)$, $||\mathbf{x}_{i,w}^{(0)}||_\infty \leq \beta$ and $||\mathbf{s}_{i,w}^{(0)}||_\infty \leq \beta$ satisfying Eq 4. Also it checks $\mathbf{C}_i^{(0)} = \mathbf{A}(H_2(\mathsf{id}_i||0))^{-1}$. If any of these verification fails, then $\mathcal{U}_i$ aborts. Otherwise $\mathcal{U}_i$ sets its secret key $\mathsf{sec}_{i,t_1\to t_2} = \mathbf{z}_i^{(0)}$ and defines $\mathsf{cert}_{i,t_1\to t_2}$ as its membership certificate corresponding to its secret key $\mathsf{sec}_{i,t_1\to t_2} = \mathbf{z}_i^{(0)}$ for the time interval $[t_1, t_2]$.

Figure 3: Join Algorithm

```
User $\mathcal{U}_i(\mathsf{usk}[i], T, \mathcal{Y})$                                GM $(\mathsf{T}_{\mathbf{A}}, \mathsf{uvk}[i], \mathcal{Y})$

Chooses: $\mathbf{z}_i^{(0)}$
Computes: $\mathbf{v}_i^{(0)} = \mathbf{F}.\mathbf{z}_i^{(0)} \mod q$
          $\mathsf{sig}_i = \mathsf{DSig.Sign}(\mathsf{usk}[i], \mathbf{v}_i^{(0)})$

                    $\xrightarrow{\left(\mathbf{v}_i^{(0)},\ \mathsf{sig}_i,\ \mathsf{uvk}[i]\right)}$

                              Runs: $\mathsf{DSig}.\mathrm{Verify}(\mathsf{usk}[\mathrm{i}], \mathsf{sig}_i, \mathbf{v}_i^{(0)})$
                              if (verification succeeds) then
                                Selects: $[t_1, t_2] \subset [1, T]$ and
                                    $\mathsf{id}_i = \mathsf{id}_i[1], \ldots, \mathsf{id}_i[\mu] \in \{0,1\}^{\mu}$
                                Computes: $\mathbf{A}_{\mathsf{id}_i} = [\mathbf{A}|\bar{\mathbf{A}}]$
                                Generates: $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}} \longleftarrow \mathsf{ExtBasis}(\mathbf{A}, \bar{\mathbf{A}}, \mathsf{T}_{\mathbf{A}})$
                                Selects: $\mathbf{s}_i$
                                Computes: $\mathbf{d}_i$ satisfying Eq (2) using $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$
                                Generates: $\mathsf{SubsetHN} \leftarrow \mathsf{Nodes}(t_1 + 1, t_2, G)$
                                for (each $w \in \mathsf{SubsetHN}$) do
                                  Computes: $\mathbf{A}_{i,w} = [\mathbf{A}_{\mathsf{id}_i} || \bar{\mathbf{A}}_w]$,
                                  Generates:
                                      $\mathsf{T}_{\mathbf{A}_w} \longleftarrow \mathsf{ExtBasis}(\mathbf{A}_{\mathsf{id}_i}, \bar{\mathbf{A}}_w, \mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}})$,
                                  Selects: $\{\mathbf{s}_{i,w}^{(0)}\}$,
                                  Computes: $\{\mathbf{x}_{i,w}^{(0)}\}$ using $\mathsf{T}_{\mathbf{A}_{i,w}}$.
                                end do
                                Computes: $\mathbf{R}_i^{(0)}$ and Sets: $\mathbf{C}_i^{(0)} = \mathbf{A}(\mathbf{R}_i^{(0)})^{-1}$
                                Generates:
                                  $(\mathbf{C}_i^{(0)}, \mathsf{T}_{\mathbf{C}_i^{(0)}}) \longleftarrow \mathsf{BasisDel}(\mathbf{A}, \mathbf{R}_i^{(0)}, \mathsf{T}_{\mathbf{A}}, \sigma)$
                                Stores:
                                $\mathsf{cert}_{i,t_1 \to t_2} = (\mathsf{id}_i, \mathbf{d}_i, \mathbf{s}_i, \{\mathbf{x}_{i,w}^{(0)}, \mathbf{s}_{i,w}^{(0)}\}_{w \in \mathsf{SubsetHN}},$
                                              $\mathsf{T}_{\mathbf{C}_i^{(0)}}, [t_1, t_2])$
                              Stores: $\mathsf{transcript}_i = (\mathbf{v}_i^{(0)}, i, \mathsf{uvk}[i], \mathsf{sig}_i, [t_1, t_2])$

                    $\xleftarrow{\mathsf{cert}_{i,t_1 \to t_2}}$

Sets: $\mathsf{sec}_{i,t_1 \to t_2} = \mathbf{z}_i^{(0)}$
```

- $\mathsf{FSGS.Update}(\mathcal{Y}, t_1, t_2, \mathsf{sec}_{i,t_1+(j-1)\to t_2}, \mathsf{cert}_{i,t_1+(j-1)\to t_2}) \longrightarrow (\mathsf{sec}_{i,t_1+j\to t_2}, \mathsf{cert}_{i,t_1+j\to t_2})$.
A user $\mathcal{U}_i$ with a valid certificate $\mathsf{cert}_{i,t_1+(j-1)\to t_2}$ and the corresponding secret key $\mathsf{sec}_{i,t_1+(j-1)\to t_2}$ for the time interval $[t_1+(j-1), t_2]$, computes $\mathbf{R}_i^{(j)} = H_2(\mathsf{id}_i||j)$ and executes the following steps to output the updated certificate $\mathsf{cert}_{i,t^{(j)}\to t_2}$ and secret key $\mathsf{sec}_{i,t^{(j)}\to t_2}$ for the time interval $[t^{(j)}, t_2]$, where $t^{(j)} = t_1 + j$ for $j = 1, 2, \ldots, (t_2 - t_1)$.

  (i) The user $\mathcal{U}_i$ runs $\mathsf{BasisDel}(\mathbf{C}_i^{(j-1)}, \mathbf{R}_i^{(j)}, \mathsf{T}_{\mathbf{C}_i^{(j-1)}}, \sigma) \longrightarrow (\mathbf{C}_i^{(j)}, \mathsf{T}_{\mathbf{C}_i^{(j)}})$ as stated in Lemma 1.1$(b)(vi)$ of Section 2 to generate

$$\mathbf{C}_i^{(j)} = \mathbf{A}(\mathbf{R}_i^{(j)})^{-1}$$

  and a short basis $\mathsf{T}_{\mathbf{C}_i^{(j)}}$ of $\Lambda_q^{\perp}(\mathbf{C}_i^{(j)})$.

  (ii) The user $\mathcal{U}_i$ randomly chooses a vector $\mathbf{z}_i^{(j)}$ from $D_{\mathbb{Z}^{4m},\sigma}$, computes $\mathbf{v}_i^{(j)} = \mathbf{F} \cdot \mathbf{z}_i^{(j)} \bmod q$ and sets user's secret for the time period $[t^{(j)}, t_2]$ as $\mathsf{sec}_{i,t^{(j)}\to t_2} = \mathbf{z}_i^{(j)}$.

  (iii) For all $w = (w_{\mathsf{num}}, w_{\mathsf{dep}}) \in \mathsf{SubsetHN} \leftarrow \mathsf{Nodes}(t^{(j)}+1, t_2, G)$ user $\mathcal{U}_i$ does the following:

   (a) sets

$$\mathbf{C}_{i,w}^{(j)} = [\mathbf{C}_i^{(j)} | \mathbf{C}_{\mathsf{id}_{i,w}}^{(j)}] \tag{5}$$

   with $\mathbf{C}_{\mathsf{id}_{i,w}}^{(j)} = [\mathbf{A}_0 + \sum\limits_{k=1}^{\mu} \mathsf{id}_i[k]\mathbf{A}_k | \mathbf{M}_0 + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1 | \mathbf{A}_0 + \sum\limits_{k=1}^{\mu} \alpha_k \mathbf{A}_k]$
   where $\alpha_k \in \{0,1\}$ is the kth bit of the μ-bit string $(0^{\mu-\delta}||\mathsf{bin}(w_{\mathsf{dep}}))$.

   Note that $\mathbf{C}_{i,w}^{(0)} = \mathbf{A}_{i,w}$.

   (b) runs $\mathsf{ExtBasis}\ (\mathbf{C}_i^{(j)}, \mathbf{C}_{\mathsf{id}_{i,w}}^{(j)}, \mathsf{T}_{\mathbf{C}_i^{(j)}}) \longrightarrow (\mathsf{T}_{\mathbf{C}_i^{(j)}|\mathbf{C}_{\mathsf{id}_{i,w}}^{(j)}} = \mathsf{T}_{\mathbf{C}_{i,w}^{(j)}})$ to get a short basis $\mathsf{T}_{\mathbf{C}_{i,w}^{(j)}}$ of $\Lambda_q^{\perp}(\mathbf{C}_{i,w}^{(j)})$.

   (c) samples short vectors $\mathbf{s}_{i,w}^{(j)} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and using the short basis $\mathsf{T}_{\mathbf{C}_{i,w}^{(j)}}$ as in Remark 1 in Section 2, computes a short vector $\mathbf{x}_{i,w}^{(j)} = [\mathbf{x}_{i,w,1}^{(j)} | \mathbf{x}_{i,w,2}^{(j)} | \mathbf{x}_{i,w,3}^{(j)}]^t$ with $||\mathbf{x}_{i,w}^{(j)}||_\infty \le \beta$ and $\mathbf{x}_{i,w,1}^{(j)} = [\mathbf{x}_{i,w,1,1}^{(j)} | \mathbf{x}_{i,w,1,2}^{(j)}] \in \mathbb{Z}^{2m}$ satisfying

$$\mathbf{C}_i^{(j)}\mathbf{x}_{i,w,1,1}^{(j)} + \mathbf{C}_{\mathsf{id}_{i,w}}^{(j)} \left[\mathbf{x}_{i,w,1,2}^{(j)} | \mathbf{x}_{i,w,2}^{(j)} | \mathbf{x}_{i,w,3}^{(j)}\right]^t = \mathbf{u} + \mathbf{D} \cdot \mathbf{m}_{i,w}^{(j)} \bmod q \tag{6}$$

$$\text{and} \quad \mathbf{C}_i^{(j)}\mathbf{x}_{i,w,1,1}^{(j)} = \mathbf{0} \bmod q$$

$$\text{where} \quad \mathbf{m}_{i,w}^{(j)} = \mathsf{bin}(\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_i^{(j)}) + \mathbf{D}_1 \cdot \mathbf{s}_{i,w}^{(j)}) \bmod q \tag{7}$$

   where $\mathbf{x}_{i,w,1}^{(j)} \in \mathbb{Z}^{2m}$ and $\mathbf{x}_{i,w,2}^{(j)}, \mathbf{x}_{i,w,3}^{(j)} \in \mathbb{Z}^m$.

(v) The updated certificate is
$\mathsf{cert}_{i,t_1^{(j)}\to t_2} = \big(\mathsf{id}_i, \mathbf{d}_i, \mathbf{s}_i, \{\mathbf{x}_{i,w}^{(j)}, \mathbf{s}_{i,w}^{(j)}\}_{w\in\mathsf{SubsetHN}\leftarrow\mathsf{Nodes}(t^{(j)}+1,t_2,G)}, \mathbf{C}_i^{(j)}, \mathsf{T}_{\mathbf{C}_i^{(j)}}, [t^{(j)}, t_2]\big)$
and the associated updated secret is $\mathsf{sec}_{i,t^{(j)}\to t_2} = \{\mathbf{v}_i^{(0)}, \mathbf{z}_i^{(j)}\}$, for $j = 1, 2, ..., (t_2 - t_1)$.

(vi) The user $\mathcal{U}_i$ also updates the public state St=($\mathsf{St}_{users}, \mathsf{St}_{trans}$).

(vii) After updating the keys for the period $[t^{(j)}, t_2]$, the algorithm erases $\mathsf{cert}_{i,t^{(j-1)}\to t_2}$ and $\mathsf{sec}_{i,t^{(j-1)}\to t_2}$ for the time period $[t^{(j-1)}, t_2]$.

- $\mathsf{FSGS.Sign}(\mathsf{sec}_{i,t^{(j)}\to t_2}, \mathsf{cert}_{i,t^{(j)}\to t_2}, \mathcal{Y}, M) \longrightarrow (\sigma)$: To sign a message $M \in \{0,1\}^*$, the user $\mathcal{U}_i$ generates a one time signature signing-verifying key-pair (VK,SK) by running $\mathsf{OTS}\cdot\mathcal{G}$ and performs the following steps:

(i) Computes $\mathbf{G}_0 = H_0(\mathsf{VK}) \in \mathbb{Z}_q^{n\times 2m}$, randomly chooses $\mathbf{e}_0 \hookleftarrow \chi^n, \mathbf{e}_1 \hookleftarrow \chi^m, \mathbf{e}_2 \hookleftarrow \chi^{2m}$, where χ is a γ bounded distribution, extracts $\mathbf{F}$, $\mathbf{B}$ from $\mathcal{Y}$, $\mathbf{z}_i^{(j)}$ from $\mathsf{sec}_{i,t^{(j)}\to t_2}$, calculates

$$\mathbf{v}_i^{(j)} = \mathbf{F}\cdot\mathbf{z}_i^{(j)} \tag{8}$$

and encrypts

$$\mathbf{y}_i^{(j)} = \mathsf{bin}(\mathbf{v}_i^{(j)}) \in \{0,1\}^{2m} \tag{9}$$

to generate the ciphertext $\mathsf{GPV\text{-}IBE\cdot Enc}(\mathbf{B}, \mathsf{VK}, \mathbf{y}_i^{(j)}) \to \mathbf{c}_{\mathbf{v}_i^{(j)}}$.

$$\mathbf{c}_{\mathbf{v}_i^{(j)}} = (\mathbf{c}_1, \mathbf{c}_2) = (\mathbf{B}^t\mathbf{e}_0 + \mathbf{e}_1, \mathbf{G}_0^t\mathbf{e}_0 + \mathbf{e}_2 + \mathbf{y}_i^{(j)}\lfloor q/2\rfloor) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{2m} \tag{10}$$

(ii) For each signature, the ith signer chooses a new random $\mathbf{R}_i \in \mathbb{Z}_q^{n\times n}$ and computes

$$\mathbf{C}_{i,R}^{(j)} = \mathbf{R}_i\mathbf{C}_i^{(j)}.$$

Note that

$$\mathbf{C}_i^{(j)}\mathbf{x}_{i,w,1,1}^{(j)} = \mathbf{0} \bmod q \Rightarrow \mathbf{R}_i\mathbf{C}_i^{(j)}\mathbf{x}_{i,w,1,1}^{(j)} = \mathbf{0} \bmod q$$

and thus basis $\mathsf{T}_{\mathbf{C}_i^{(j)}}$ of $\Lambda_q^\perp(\mathbf{C}_i^{(j)})$ serves as a basis $\mathsf{T}_{\mathbf{C}_{i,R}^{(j)}}$ of $\Lambda_q^\perp(\mathbf{C}_{i,R}^{(j)})$. Random matrix $\mathbf{R}_i$ helps in maintaining anonymity.

(iii) Rewrite Eq 5 as

$$\mathbf{C}_{i,w}^{(j)} = [\mathbf{C}_{\mathsf{id}_i}^{(j)}|\mathbf{M}_0 + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1|\mathbf{A}_0 + \sum_{k=1}^{\mu}\alpha_k\mathbf{A}_k] \tag{11}$$

where

$$\mathbf{C}_{\mathsf{id}_i}^{(j)} = [\mathbf{C}_{i,R}^{(j)}|\mathbf{A}_0 + \sum_{k=1}^{\mu}\mathsf{id}_i[k]\mathbf{A}_k].$$

Let $\mathbf{H}$ and $\widetilde{\mathbf{H}}$ be the following $(2n\times m)$ and $(4n\times 2m)$ matrices.

$$\mathbf{H} = \begin{bmatrix} 1\ 2\ 4 \cdots\ 2^{\lceil \log q \rceil -1} & & & \\ & 1\ 2\ 4 \cdots\ 2^{\lceil \log q \rceil -1} & & \\ & & \vdots & \\ & & & 1\ 2\ 4 \cdots\ 2^{\lceil \log q \rceil -1} \end{bmatrix}_{4n \times 2n\lceil \log q \rceil}$$

$$\widetilde{\mathbf{H}} = \begin{bmatrix} 1\ 2\ 4 \cdots\ 2^{\lceil \log q \rceil -1} & & & \\ & 1\ 2\ 4 \cdots\ 2^{\lceil \log q \rceil -1} & & \\ & & \vdots & \\ & & & 1\ 2\ 4 \cdots\ 2^{\lceil \log q \rceil -1} \end{bmatrix}_{4n \times 4n\lceil \log q \rceil}$$

In order to prove the knowledge of

- $\mathbf{s}_i, \mathbf{d}_i, \mathbf{z}_i^{(j)}$ with infinity norm bound β
- $\mathbf{x}_{i,w}^{(j)}, \mathbf{s}_{i,w}^{(j)}$, for all $w \in \mathsf{SubsetHN} \leftarrow \mathsf{Nodes}(t^{(j)}+1, t_2, G)$ with infinity norm bound β
- $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2$ with infinity norm bound B
- $\mathbf{r}_i, \mathbf{m}_{i,w}^{(j)}$ satisfying Eq 3, Eq 7 respectively
- $\mathsf{id}_i \in \{0,1\}^\mu$, $\mathbf{y}_i^{(j)} = \mathsf{bin}(\mathbf{v}_i^{(j)})$ satisfying the following system of equations:

$$\left\{ \begin{array}{rl} \text{from Eq 2}: & \mathbf{A}\mathbf{d}_{i,1} + \mathbf{A}_0\mathbf{d}_{i,2} + \sum_{k=1}^{\mu} \mathbf{A}_k \mathsf{id}_i[k]\mathbf{d}_{i,2} - \mathbf{D}\mathbf{r}_i = \mathbf{u} \bmod q \\ \text{from Eq 3}: & \mathbf{H}\cdot\mathbf{r}_i = \mathbf{D}_0\mathbf{y}_i^{(0)} + \mathbf{D}_1\mathbf{s}_i \bmod q \\ \text{from Eq 8 and 9}: & \mathbf{F}\cdot\mathbf{z}_i^{(j)} = \widetilde{\mathbf{H}}\mathbf{y}_i^{(j)} \bmod q \\ \text{from Eq 10}: & \mathbf{c}_1 = \mathbf{B}^t\mathbf{e}_0 + \mathbf{e}_1 \bmod q \\ \text{from Eq 10}: & \mathbf{c}_2 = \mathbf{G}_0^t\mathbf{e}_0 + \mathbf{e}_2 + \lfloor q/2 \rfloor \mathbf{y}_i^{(j)} \bmod q \\ \text{from Eq 6 and 11}: & \mathbf{C}_{\mathsf{id}_i}^{(j)}\mathbf{x}_{i,w,1}^{(j)} + \mathbf{M}_0\mathbf{x}_{i,w,2}^{(j)} + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1\mathbf{x}_{i,w,2}^{(j)} + \mathbf{A}_0\mathbf{x}_{i,w,3}^{(j)} \\ & \qquad + \sum_{k=1}^{\mu} \alpha_k \mathbf{A}_k \mathbf{x}_{i,w,3}^{(j)} - \mathbf{D}\mathbf{m}_{i,w}^{(j)} = \mathbf{u} \bmod q \\ \text{from Eq 7}: & \mathbf{H}\cdot\mathbf{m}_{i,w}^{(j)} = \mathbf{D}_0\mathbf{y}_i^{(j)} + \mathbf{D}_1\cdot\mathbf{s}_{i,w}^{(j)} \bmod q \end{array} \right\} \quad (12)$$

We write the above system of equations in the form $\mathbf{P}\mathbf{x} = \mathbf{v}$ where $\mathbf{P}, \mathbf{v}$ are public and $\mathbf{x}$ is secret, as explained in the subsequent Section 5 and employ the non-interactive ZKAoK protocol (see Remark 3 in Section 2.5) for zero knowledge proof of argument described in Section 2.5 to generate a proof $\Pi_i = (\{\mathsf{COM}_{i,k}\}_{k=1}^{s}, \mathsf{Ch}_i,$ $\{\mathsf{RSP}_{i,k}\}_{k=1}^{s})$ where $\mathsf{Ch}_i = H(M, \mathsf{VK}, \mathbf{c}_{\mathbf{v}_i^{(j)}}, \{\mathsf{COM}_{i,k}\}_{k=1}^{s}) \in \{1,2,3\}^s$ to prove the knowledge of $\mathbf{x} \in \mathcal{R} = \{(\mathbf{P}, \mathbf{v}) \in \mathbb{Z}_q^{D\times L} \times \mathbb{Z}_q^D, \mathbf{x} \in \mathsf{VALID} : \mathbf{P}\mathbf{x} = \mathbf{v} \bmod q\}$ satisfying $\mathbf{P}\mathbf{x} = \mathbf{v}$.

(iv) Computes a one time signature, $\mathsf{osig}_i = \mathsf{OTS}.\mathcal{S}(\mathsf{SK}, \mathbf{c}_{\mathbf{v}_i^{(j)}}, \Pi_i)$ and outputs the message signature pair (M, σ) where

$$\sigma = (\mathsf{VK}, \mathbf{c}_{\mathbf{v}_i^{(j)}}, \Pi_i, \mathsf{osig}_i, \mathbf{C}_{i,R}^{(j)}, [t^{(j)}, t_2]) \tag{13}$$

- $\mathsf{FSGS.Verify}(\sigma, t, M, \mathcal{Y}) \longrightarrow (1 \ \vee \ 0)$: The verifier parses σ as in Eq (13) and returns 1 if and only if the following holds:

  (i) $t \in [t^{(j)}, t_2]$.

  (ii) $\mathsf{OTS}.\mathcal{V}(\mathsf{VK}, (\mathbf{c}_{\mathbf{v}_i^{(j)}}, \Pi_i), \mathsf{osig}_i) = 1$.

  (iii) The proof of knowledge Π_i properly verifies following $\mathsf{ZKAoK.Verification}$ in section 2.5.

- $\mathsf{FSGS.Open}(\sigma, t, M, \mathcal{Y}, \mathsf{S_{OA}}, \mathsf{St}) \longrightarrow (i \ \vee \perp)$: The opening authority parses σ as in Eq 13 and executes the following steps using its secret key $\mathsf{S_{OA}} = \mathsf{T}_{\mathbf{B}}$.

  (i) Computes $\mathbf{G}_0 = H_0(\mathsf{VK}) \in \mathbb{Z}_q^{n\times 2m}$ and use $\mathsf{T}_{\mathbf{B}}$ to compute a small norm matrix $\mathbf{K}_{\mathsf{VK}} \in \mathbb{Z}^{m\times 2m}$ using the algorithm $\mathsf{SamplePre}$ as in Lemma 1.1$(b)(ii)$ such that $\mathbf{B} \cdot \mathbf{K}_{\mathsf{VK}} = \mathbf{G}_0 \bmod q$. Note that this equation gives $2m$ number of equations, each corresponds to inhomogeneous short integer solution (ISIS) problem. As the OA has the short basis $\mathsf{T}_{\mathbf{B}}$, it can generate the short matrix $\mathbf{K}_{\mathsf{VK}}$ by Remark 1 in Section 2.1.

  (ii) Calls $\mathsf{GPV\text{-}IBE.Dec}(\mathbf{K}_{\mathsf{VK}}, \mathbf{c}_{\mathbf{v}_i^{(j)}}) \to \mathsf{bin}(\mathbf{v}) \in \{0,1\}^{2m}$ and checks if $\mathbf{v} = \widetilde{\mathbf{H}} \cdot \mathsf{bin}(\mathbf{v}) \bmod q$ appears in a record $\mathsf{transcript}_i = (\mathbf{v}, i, \mathsf{uvk}[i], \mathsf{sig}_i, [t_1, t_2])$ of the database $\mathsf{St_{trans}}$ for some i. If so, output i indicating that the user $\mathcal{U}_i$ is the signer. Otherwise, outputs $\perp$.
  Note that if the obtained $\mathbf{v}$ is available as one of the records in the transcript, then the OA will output the corresponding index i appears in the corresponding tuple.

Remark 4: Note that in our construction, we do not expose $[\mathbf{A}_0 + \sum_{i=1}^{\mu} \mathsf{id}_i[k]\mathbf{A}_k]$ in public while executing zero-knowledge between the prover the verifier which preserves the anonymity of our scheme. If $[\mathbf{A}_0 + \sum_{i=1}^{\mu} \mathsf{id}_i[k]\mathbf{A}_k]$ gets exposed then we can construct an adversary breaking the anonymity as shown below:
1) By making two Join queries for two users U_0 and U_1, the adversary obtains two pairs $(\mathsf{sec}_0^*, \mathsf{cert}_0^*)$ and $(\mathsf{sec}_1^*, \mathsf{cert}_1^*)$. Note that adversary knows id_0 and id_1 for U_0 and U_1 respectively.
2) The challenger returns σ^* signed by U_d where $d \in \{0,1\}$.
3) Given σ^*, the adversary checks the matrix $\mathbf{C}_{\mathsf{id}}^{(j)}$ in σ^*. Adversary can determine d by checking whether the right block of the matrix $\mathbf{C}_{\mathsf{id}}^{(j)}$ is equivalent to $[\mathbf{A}_0 + \sum_{i=1}^{\mu} \mathsf{id}_i[k]\mathbf{A}_k]$ or not.

Remark 5: Our scheme withstands mis-identification attack because after a user's secret key and certificate gets updated, the update algorithm updates the $\mathsf{transcript}$ in $\mathsf{St}_{\mathsf{trans}}$. The updated transcript helps the open algorithm to run and enables the OA to find the record available in the $\mathsf{transcript}$. If the update algorithm does not update the $\mathsf{transcript}$ then we can construct an adversary against mis-identification attack as follows:
1) The aversary queries to $\mathsf{Q}_{\mathsf{a-join}}$ with $[t_1, t_2] = [1, T]$ and obtains user's secret key and certificate. The database contains $\mathsf{transcript}_1 = (\mathbf{v}_1^{(0)}, 1, \mathsf{uvk}[1], sig, [1, T])$.
2) The adversary updates the certificate and secret key. In this procedure, the adversary randomly chooses $\mathbf{z}_1^{(1)}$ from a Gaussian distribution and computes $\mathbf{v}_1^{(1)} = \mathbf{F} \cdot \mathbf{z}_1^{(1)} \bmod q$. As the transcript is not updated by the update algorithm, $\mathbf{v}_1^{(1)}$ is not available in the transcript.
3) The adversary generates the signature σ^* using the updated certificate and secret key. Consequently, σ^* contains the updated value $\mathbf{v}_1^{(1)}$ instead of $\mathbf{v}_1^{(0)}$.
4) The Adversary outputs M^*, $t^* = 1$, and σ^*.
5) The OA will output nothing since there are no $\mathsf{transcript} = (\mathbf{v}, i, \mathsf{uvk}[i], sig_i, [t_1, t_2])$ such that $\mathbf{v} = \mathbf{v}_1^{(1)}$.

4.1 Efficiency

The forward secure dynamic group signature has public key size $\mathcal{O}(\mu mn \log q) = \tilde{\mathcal{O}}(mn) \log N = \tilde{\mathcal{O}}(n^2) \log N$. The secret key size of a group member is $\mathcal{O}(m) = \tilde{\mathcal{O}}(n)$. The signature size depends upon the non-interactive argument Π. The communication cost of Π is $s \cdot \mathcal{O}(\mu m \log q) = s \cdot \tilde{\mathcal{O}}(n) \log N$. Thus the total size of the signature is $s \cdot \tilde{\mathcal{O}}(n^3) \log N = \tilde{\mathcal{O}}(n^3) \log N$.

5 The Underlying Zero Knowledge for Our Scheme

It is the argument system for the zero knowledge Π_i. The argument system upon which our group signature scheme is built can be summarized as follows:

1. **Commom Input**: $\mathbf{C}_{\mathsf{id}_i}^{(j)} \in \mathbb{Z}_q^{n \times 2m}, \mathbf{M}_0, \mathbf{M}_1 \in \mathbb{Z}_q^{n \times m}, \mathbf{A}_{\mathsf{id}_i} \in \mathbb{Z}_q^{n \times 2m}, \mathbf{A}, \mathbf{A}_0, ..., \mathbf{A}_\mu, \mathbf{D} \in \mathbb{Z}_q^{n \times m}, \mathbf{D}_0, \mathbf{D}_1 \in \mathbb{Z}_q^{2n \times 2m}, \mathbf{F} \in \mathbb{Z}_q^{4n \times 4m}$, $\mathbf{H} \in \mathbb{Z}_q^{2n \times m}, \widetilde{\mathbf{H}} \in \mathbb{Z}_q^{4n \times 2m}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}, \mathbf{G}_0 \in \mathbb{Z}_q^{n \times 2m}, \mathbf{u} \in \mathbb{Z}_q^n, \mathbf{c}_1 \in \mathbb{Z}_q^m, \mathbf{c}_2 \in \mathbb{Z}_q^{2m}$.

2. **Prover's Input**: $\mathbf{x}_{i,w,1} \in [-\beta, \beta]^{2m}, \mathbf{x}_{i,w,2} \in [-\beta, \beta]^m, \mathbf{x}_{i,w,3} \in [-\beta, \beta]^m, \mathbf{y}_i^{(0)} \in \{0,1\}^{2m}$, $\mathbf{y}_i^{(j)} \in \{0,1\}^{2m}, \mathbf{s}_{i,w}^{(j)} \in [-\beta, \beta]^{2m}, \mathbf{m}_{i,w}^{(j)} \in \{0,1\}^m, \mathbf{r}_i \in \{0,1\}^m, \mathbf{s}_i \in [-\beta, \beta]^{2m}, \mathbf{d}_{i,1}, \mathbf{d}_{i,2} \in [-\beta, \beta]^m, \mathbf{z}_i^{(j)} \in [-\beta, \beta]^{4m}, \mathsf{id}_i = (\mathsf{id}_i[1], \mathsf{id}_i[2], ..., \mathsf{id}_i[\mu])^t \in \{0,1\}^\mu, \mathbf{e}_0 \in [-\gamma, \gamma]^n, \mathbf{e}_1 \in [-\gamma, \gamma]^m, \mathbf{e}_2 \in [-\gamma, \gamma]^{2m}$. Note that $x \in [-\beta, \beta]^m$ means $\mathbf{x} \in \mathbb{Z}_q^m$ with $||\mathbf{x}||_\infty \leq \beta$.

3. **Prover's Goal**: Convince the verifier in zero knowledge of the system of equations given in Eq 12. To achieve this, we rewrite Eq 12 as

$$\mathbf{N}_0\mathbf{x}_0 + +\mathbf{N}_1\mathbf{x}_1 + \mathbf{N}_2\mathbf{x}_2 + \mathbf{N}_3\mathbf{x}_3 + \mathbf{N}_4\mathbf{x}_4 = \mathbf{v} \bmod q \qquad (14)$$

$$\text{where } \mathbf{N}_0 = \begin{bmatrix} \mathbf{A}_{n\times m} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\ \mathbf{D}_1)_{2n\times 2m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_{4n\times 4m} \\ \mathbf{0}_{m\times m} & \mathbf{0}_{m\times 2m} & \mathbf{0}_{m\times 4m} \\ \mathbf{0}_{2m\times m} & \mathbf{0}_{2m\times 2m} & \mathbf{0}_{2m\times 4m} \\ \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} & \mathbf{0}_{2n\times 4m} \\ \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} & \mathbf{0}_{2n\times 4m} \end{bmatrix}_{(3m+11n)\times 7m},$$

$$\mathbf{N}_1 = \left[\mathbf{A}'_0|\mathbf{A}'_1|\mathbf{A}'_2|\cdots|\mathbf{A}'_\mu\right]_{(3m+11n)\times(\mu+1)m}, \text{ with } \mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{0} \end{bmatrix}_{(3m+11n)\times m}$$

$$\mathbf{N}_2 = \begin{bmatrix} (-\mathbf{D})_{n\times m} & \mathbf{0}_{2n\times 2m} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ -\mathbf{H}_{2n\times m} & (\mathbf{D}_0)_{2n\times 2m} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ \mathbf{0}_{4n\times m} & \mathbf{0}_{2n\times 2m} & \mathbf{0}_{2n\times m} & -\widetilde{\mathbf{H}}_{2n\times 2m} \\ \mathbf{0}_{m\times m} & \mathbf{0}_{m\times 2m} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ \mathbf{0}_{2m\times m} & \mathbf{0}_{2n\times 2m} & \mathbf{0}_{2n\times m} & \lfloor q/2 \rfloor 1_{2n\times 2m} \\ \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} & [-\mathbf{D}||\mathbf{0}]^t_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} & -\mathbf{H}_{2n\times m} & (\mathbf{D}_0)_{2n\times 2m} \end{bmatrix},$$

$$\mathbf{N}_3 = \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{0}_{n\times m} & \mathbf{0}_{n\times 2m} \\ \mathbf{0}_{2n\times n} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ \mathbf{0}_{4n\times n} & \mathbf{0}_{4n\times m} & \mathbf{0}_{4n\times 2m} \\ (\mathbf{B}^t)_{m\times n} & \mathbf{1}_{m\times m} & \mathbf{0}_{m\times 2m} \\ (\mathbf{G}_0^t)_{2m\times n} & \mathbf{0}_{2m\times m} & \mathbf{1}_{2m\times 2m} \\ \mathbf{0}_{2n\times n} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ \mathbf{0}_{2n\times n} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} \end{bmatrix}_{(3m+11n)\times(3m+n)},$$

$$\mathbf{N}_4 = \begin{bmatrix} \mathbf{0}_{n\times 2m} & \mathbf{0}_{n\times m} & \mathbf{0}_{n\times m} & \mathbf{0}_{n\times 2m} \\ \mathbf{0}_{2n\times 2m} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ \mathbf{0}_{4n\times 2m} & \mathbf{0}_{4n\times m} & \mathbf{0}_{4n\times m} & \mathbf{0}_{4n\times 2m} \\ \mathbf{0}_{m\times 2m} & \mathbf{0}_{m\times m} & \mathbf{0}_{m\times m} & \mathbf{0}_{m\times 2m} \\ \mathbf{0}_{2m\times 2m} & \mathbf{0}_{2m\times m} & \mathbf{0}_{2m\times m} & \mathbf{0}_{2m\times 2m} \\ (\mathbf{Z}||0)^t_{2n\times(\mu+2)m} & (\mathbf{M}_0 + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1||0)^t_{2n\times m} & (\mathbf{A}_0 + \sum_{k=1}^{\mu} \alpha_k \mathbf{A}_k||0)^t_{2n\times m} & \mathbf{0}_{2n\times 2m} \\ \mathbf{0}_{2n\times 2m} & \mathbf{0}_{2n\times m} & \mathbf{0}_{2n\times m} & (\mathbf{D}_1)_{2n\times 2m} \end{bmatrix}$$

where $\mathbf{Z} = \left[\mathbf{C}^{(j)}_{i,R}|\mathbf{A}_0|\mathbf{A}_1|\cdots|\mathbf{A}_\mu\right]_{n\times(\mu+2)m}$,

$$\mathbf{x}_0 = \begin{bmatrix} \mathbf{d}_{i,1} \\ \mathbf{s}_i \\ \mathbf{z}_i^{(j)} \end{bmatrix}, \ \mathbf{x}_1 = \left[\mathbf{d}_{i,2}||\mathsf{id}_i[1]\mathbf{d}_{i,2}||\cdots||\mathsf{id}_i[\mu]\mathbf{d}_{i,2}\right]^t,$$

$$\mathbf{x}_2 = \begin{bmatrix} \mathbf{r}_i \\ \mathbf{y}_i^{(0)} \\ \mathbf{m}^{(j)}_{i,w} \\ \mathbf{y}_i^{(j)} \end{bmatrix}, \ \mathbf{x}_3 = \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix}, \ \mathbf{x}_4 = \begin{bmatrix} \mathbf{x}' & \mathbf{x}^{(j)}_{i,w,2} & \mathbf{x}^{(j)}_{i,w,3} & \mathbf{s}^{(j)}_{i,w} \end{bmatrix}^t \text{ where}$$

$$\mathbf{x}' = \left[\mathbf{x}^{(j)}_{i,w,1,1}||\mathbf{x}^{(j)}_{i,w,1,2}||\mathsf{id}_i[1]\mathbf{x}^{(j)}_{i,w,1,2}||\cdots||\mathsf{id}_i[\mu]\mathbf{x}^{(j)}_{i,w,1,2}\right]^t, \ \mathbf{v} = [\mathbf{u} \quad \mathbf{0} \quad \mathbf{0} \quad \mathbf{c}_1 \quad \mathbf{c}_2 \quad \mathbf{u} \quad \mathbf{0}]^t$$

- Let $\widetilde{p} = \lfloor \log \gamma \rfloor + 1$ and $p = \lfloor \log \beta \rfloor + 1$

- B_l^2 is the collection of $2l$ length strings over the alphabet $\{0, 1\}$ having equal number of $0's,\ 1's$ and

- B^3_l is the collection of $3l$ length strings over the alphabet $\{-1, 0, 1\}$ having equal number of $0's,\ 1's\ \ -1's$.

Let $\mathbf{N}'_i$ and $\widehat{\mathbf{x}}_i$ for $i = 0, 1, \dots, 4$ are obtained as follows using the algorithm Dec.Ext described in Section 2.4 and $\widehat{K}_{m,\beta}$ described in the same section with the property that $\widehat{K}_{m,\beta}\widehat{\mathbf{x}} = \mathbf{x}$.

$\mathsf{Dec\text{-}Ext}_{7m,p}(\mathbf{x}_0) \to (\widehat{\mathbf{x}}_0 \in B^3_{7mp})$

$\mathbf{N}_0.\widehat{K}_{7m,\beta} \to (\mathbf{N}'_0 \in \mathbb{Z}_q^{(3m+11n)\times 3(7m)p})$

$\mathsf{Dec\text{-}Ext}_{(\mu+1)m,p}(\mathbf{x}_1) \to (\widehat{\mathbf{x}}_1 \in B^3_{(\mu+1)mp})$

$\mathbf{N}_1.\widehat{K}_{(\mu+1)m,\beta} \to (\mathbf{N}'_1 \in \mathbb{Z}_q^{(3m+11n)\times 3(\mu+1)mp})$

$\mathsf{Dec\text{-}Ext}_{6m,p}(\mathbf{x}_2) \to (\widehat{\mathbf{x}}_2 \in B^2_{6mp})$

$[\mathbf{N}_2||0^{(3m+11n)\times 6mp}] = \mathbf{N}'_2 \in \mathbb{Z}_q^{(3m+11n)\times 2(6m)p})$

$\mathsf{Dec\text{-}Ext}_{(3m+n),\widetilde{p}}(\mathbf{x}_3) \to (\widehat{\mathbf{x}}_3 \in B^3_{(3m+n)\widetilde{p}})$

$\mathbf{N}_3.\widehat{K}_{(3m+n),\gamma} \to (\mathbf{N}'_3 \in \mathbb{Z}_q^{(3m+11n)\times 3(3m+n)\widetilde{p}})$

$\mathsf{Dec\text{-}Ext}_{6m,p}(\mathbf{x}_4) \to (\widehat{\mathbf{x}}_4 \in B^3_{(\mu+6)mp})$

$\mathbf{N}_4.\widehat{K}_{6m,\beta} \to (\mathbf{N}'_4 \in \mathbb{Z}_q^{(3m+11n)\times 3(\mu+6)mp})$

Next we set

- $\mathbf{P} = [\mathbf{N}'_0||\mathbf{N}'_1||\mathbf{N}'_2||\mathbf{N}'_3||\mathbf{N}'_4] \in \mathbb{Z}_q^{5(3m+11n)\times((52+4\mu)mp+(9m+3n)\widetilde{p})}$
- $\mathbf{x} = [\widehat{\mathbf{x}}_0||\widehat{\mathbf{x}}_1||\widehat{\mathbf{x}}_2||\widehat{\mathbf{x}}_3||\widehat{\mathbf{x}}_4] \in \mathbb{Z}_q^{(52+4\mu)mp+(9m+3n)\widetilde{p}}$
- $L = (52 + 4\mu)mp + (9m + 3n)\widetilde{p}$, $D = 5(3m + 11n)$
- $\mathsf{VALID}=\{\mathbf{w} \in \{-1, 0, 1\}^L :\ \mathbf{w} = (\mathbf{w}^t_0||\mathbf{w}^t_1||\mathbf{w}^t_2||\mathbf{w}^t_3||\mathbf{w}^t_4)$ for some $\mathbf{w}_0 \in B^3_{7mp}, \mathbf{w}_1 \in B^3_{(\mu+1)mp}, \mathbf{w}_2 \in B^2_{6mp}, \mathbf{w}_3 \in B^3_{(3m+n)\widetilde{p}}, \mathbf{w}_4 \in B^3_{(\mu+6)mp}\}$. Then $\mathbf{x} \in \mathsf{VALID}$.

Note that

$$\begin{aligned}&\mathbf{N}_0\widehat{K}_{m,\beta}\widehat{\mathbf{x}}_0 + \mathbf{N}_1\widehat{K}_{m,\beta}\widehat{\mathbf{x}}_1 + [\mathbf{N}_2||0^{(3m+11n)\times 6mp}]\widehat{\mathbf{x}}_2 + \mathbf{N}_3\widehat{K}_{m,\gamma}\widehat{\mathbf{x}}_3 + \mathbf{N}_4\widehat{K}_{m,\beta}\widehat{\mathbf{x}}_4\\ &= \mathbf{N}_0\mathbf{x}_0 + \mathbf{N}_1\mathbf{x}_1 + \mathbf{N}_2\mathbf{x}_2 + \mathbf{N}_3\mathbf{x}_3 + \mathbf{N}_4\mathbf{x}_4 = \mathbf{v} \bmod q.\end{aligned}$$

Let us now define the set S and permutations of L elements $\mathsf{T}_\pi : \pi \in S$ satisfying the following conditions:

(i) $\mathbf{z} \in \mathsf{VALID} \Leftrightarrow \mathsf{T}_\pi(\mathbf{z}) \in \mathsf{VALID}$

(ii) $\mathbf{z} \in \mathsf{VALID} \Leftrightarrow \mathsf{T}_\pi(\mathbf{z})$ is uniform in VALID whenever π is uniform in S.

Let us define $S = S_{21mp} \times S_{3(\mu+1)mp} \times S_{12mp} \times S_{(9m+3n)\widetilde{p}} \times S_{(3\mu+18)mp}$.
Then for any randomly selected $\pi = (\pi_0, \pi_1, \pi_2, \pi_3, \pi_4)$ and $\mathbf{z} = (\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4) \in$

VALID, we have $\mathsf{T}_\pi(\mathbf{z}) = (\pi_0(\mathbf{z}_0), \pi_1(\mathbf{z}_1), \pi_2(\mathbf{z}_2), \pi_3(\mathbf{z}_3), \pi_4(\mathbf{z}_4))$ satisfying above conditions (i) and (ii). Finally, we invoke the algorithm ZKAoK described in 2.5 for the relation $\mathcal{R} = \{(\mathbf{P}, \mathbf{v}) \in \mathbb{Z}_q^{D\times L} \times \mathbb{Z}_q^D, \mathbf{x} \in \mathsf{VALID} : \mathbf{P}\mathbf{x} = \mathbf{v} \bmod q\}$ for statistical zero knowledge argument of knowledge.

6 Security

Theorem 6.1. *The group signature scheme* FSGS *described in Section 4 is secure against mis-identification attack as per the security framework given in Algorithm 2 of Section 3.2 under the* SIS *assumption in the random oracle model.*

Proof: Let $\mathcal{A}$ be a PPT adversary that breaks the security of our group signature scheme FSGS with non-negligible advantage ϵ. We will construct a simulator $\mathcal{B}$ that solves the SIS problem using $\mathcal{A}$ as a subroutine i.e., given $\bar{\mathbf{A}} = [\bar{\mathbf{A}}_1|\bar{\mathbf{A}}_2] \in \mathbb{Z}_q^{n\times 2m}$ with $m = 2n\lceil \log q\rceil$ where $\bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2 \in \mathbb{Z}_q^{n\times m}$ and a real number β, the simulator $\mathcal{B}$ interacts with $\mathcal{A}$ and finds a vector $\mathbf{z} \in \mathbb{Z}^{2m}$ with $||\mathbf{z}|| \leq \beta$ satisfying $\bar{\mathbf{A}}\mathbf{z} = 0 \bmod q$.

The simulator $\mathcal{B}$ acts as the KGC as well as the GM. It first chooses randomly $\widehat{\mathsf{id}} \hookleftarrow U(\{0,1\}^\mu)$, $i^* \hookleftarrow U([1,\delta])$ and $\widehat{t} \in [1,T]$ where δ is the cardinality of the set U^a of adversarially controlled users since their admission and $T = 2^l$ is the maximum allowable time periods with $\mu \geq \log l$. The total number of users supported by the system is $N = 2^\mu$. Also $\mathcal{B}$ has access to a knowledge extractor for each cheating prover by the soundness property of zero knowledge argument of knowledge ZKAoK for the relation $\mathbf{P}\mathbf{x} = \mathbf{v}$ as described in Section 1.5 where $\mathbf{P}$, $\mathbf{v}$ are public and $\mathbf{x}$ is secret witness. The knowledge extractor extracts the witness $\mathbf{x}$ involved in the interaction between the cheating prover $\mathcal{A}$ and the verifier $\mathcal{B}$.

After repeated executions of the adversary $\mathcal{A}$, let $\mathsf{id}^* \in \{0,1\}^\mu$ be the identifier in the witness $\mathbf{x}$ revealed by the knowledge extractor of the proof system for $\mathbf{P}\mathbf{x} = \mathbf{v}$ associated in $\mathcal{A}$'s forgery (M^*, σ^*, t^*) where σ^* is a forged signature on a message M^* at time t^*. The identifier $\widehat{\mathsf{id}}$ chosen by $\mathcal{B}$ at the beginning before the Setup works as a suspection of the identifier id^* disclosed by the knowledge extractor from σ^* which can be used in solving the given SIS instance. We have the following two cases.

Case I : $\mathsf{id}^* = \widehat{\mathsf{id}}$ and id^* does not belong to any user in U^a.

Case II : $\mathsf{id}^* = \widehat{\mathsf{id}}$ and id^* belongs to a certain group member in U^a but $t^* \notin [t_1, t_2]$ where $[t_1, t_2]$ is the time period for which $\widehat{\mathsf{id}}$ has been issued certificate.

Subcase I : $\mathbf{r}^* \neq \mathbf{r}_{i^*}$ where

$$\mathbf{r}^* = \mathsf{bin}(\mathbf{D}_0\mathsf{bin}(\mathbf{v}^*) + \mathbf{D}_1\mathbf{s}^*), \text{ and}$$

$$\mathbf{r}_{i^*} = \mathsf{bin}(\mathbf{D}_0\mathsf{bin}(\mathbf{v}_{i^*}) + \mathbf{D}_1\mathbf{s}_{i^*}).$$

Here $\mathbf{r}^*$ is contained in the witness $\mathbf{x}$ that is disclosed to $\mathcal{B}$ by the knowledge extractor for the relation $\mathbf{P}\mathbf{x} = \mathbf{v}$ associated in $\mathcal{A}$'s forgery σ^* on message M^* at time t^* and $\mathbf{r}_{i^*}$ corresponds to an adversarially controlled user $\mathcal{U}_{i^*} \in U^a$. Since the forgery σ^* on a message M^* at time t^* is framed by $\mathcal{A}$ and $\mathcal{U}_{i^*} \in U^a$ with i^* pre selected by $\mathcal{B}$ before the Setup, both the vectors $\mathbf{v}^*, \mathbf{v}_{i^*} \in \mathbb{Z}_q^{2m}$ are chosen by $\mathcal{A}$ and $\mathbf{s}^*$ is also selected by $\mathcal{A}$ while $\mathbf{s}_{i^*} \in \mathbb{Z}^{2m}$ are selected by $\mathcal{B}$. The matrices $\mathbf{D}_0, \mathbf{D}_1 \in \mathbb{Z}_q^{2n\times 2m}$ are part of the public parameters $\mathcal{Y}$ set by $\mathcal{B}$.

Subcase II : $\mathbf{r}^* = \mathbf{r}_{i^*}$ but $(\mathsf{bin}(\mathbf{v}^*), \mathbf{s}^*) \neq (\mathsf{bin}(\mathbf{v}_{i^*}), \mathbf{s}_{i^*})$, where $\mathbf{r}^*, \mathbf{r}_{i^*}, \mathbf{s}^*, \mathbf{s}_{i^*}$ are as above in Subcase I.

$\langle A\rangle$ Case I ($\mathsf{id}^* = \widehat{\mathsf{id}}$ and id^* does not belong to any user in U^a): The simulator $\mathcal{B}$ sets the group public key $\mathcal{Y}$ using $\widehat{\mathsf{id}}$ and the given SIS instance $\bar{\mathbf{A}} = [\bar{\mathbf{A}}_1|\bar{\mathbf{A}}_2]$, $\bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2 \in \mathbb{Z}_q^{n\times m}$, $m = 2n\lceil \log q\rceil$ as follows.

• ***Setup***: The simulator $\mathcal{B}$ runs the algorithm $\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{L}, \mathsf{T}_{\mathbf{L}})$ with $||\widetilde{\mathsf{T}}_{\mathbf{L}}|| \leq \mathcal{O}(\sqrt{n\log q})$. It samples $(\mu+2)$ Gaussian matrices $\{\mathbf{L}_k\}_{k=0}^{\mu}, \mathbf{L}_D \in \mathbb{Z}^{m\times m}$ with each matrix having its columns sampled independently from $D_{\mathbb{Z}^m,\sigma}$ where σ is of size $\Omega(\sqrt{n\log q}\log n)$ and sets the matrices $\{\mathbf{A}_k\}_{k=0}^{\mu}$, $\mathbf{D}$ and $\mathbf{A}$ as

$$\mathbf{A}_0 = \bar{\mathbf{A}}_1\mathbf{L}_0 + \left(\sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\cdot \mathbf{L}\right) \tag{15}$$

$$\mathbf{A}_k = \bar{\mathbf{A}}_1 \cdot \mathbf{L}_k + (-1)^{\widehat{\mathsf{id}}[k]}\cdot \mathbf{L} \quad \text{for } k \in [1,\mu] \tag{16}$$

$$\mathbf{D} = \bar{\mathbf{A}}_1 \cdot \mathbf{L}_D, \tag{17}$$

$$\mathbf{A} = \bar{\mathbf{A}}_1.$$

Next, $\mathcal{B}$ picks $\mathbf{h}_u \hookleftarrow D_{\mathbb{Z}^m,\sigma}$ and sets the vector $\mathbf{u}$ as

$$\mathbf{u} = \bar{\mathbf{A}}_1.\mathbf{h}_u \in \mathbb{Z}_q^n.$$

It sets the error bound γ of size $\sqrt{n}\omega(\log n)$ for encryption scheme $\mathsf{GPV\text{-}IBE}$, chooses matrices

$$\mathbf{M}_0, \mathbf{M}_1 \hookleftarrow U(\mathbb{Z}_q^{n\times m}),\ \mathbf{D}_0, \mathbf{D}_1 \hookleftarrow U(\mathbb{Z}_q^{2n\times 2m}),\ \mathbf{F} \hookleftarrow U(\mathbb{Z}_q^{4n\times 4m})$$

at random, selects three hash functions

$$H : \{0,1\}^* \rightarrow \{1,2,3\}^s,\ H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^{n\times 2m} \text{ and } H_2 : \{0,1\}^* \rightarrow \mathcal{I}$$

and a one time signature OTS=$(\mathcal{G}, \mathcal{S}, \mathcal{V})$. Here s is an integer of size $\omega(\log n)$ and $\mathcal{G}, \mathcal{S}, \mathcal{V}$ are the key generation, signing and verification algorithms respectively of OTS. The simulator $\mathcal{B}$ generates signing-verification key pair ($\mathsf{usk}[i]$, $\mathsf{uvk}[i]$) for each user $\mathcal{U}_i$, $i \in [N]$, by running the key generation algorithm $\mathsf{DSig.KeyGen}$ of a digital signature scheme DSig=(KeyGen, Sign, Verify). The simulator sends $\mathsf{usk}[i]$ to $\mathcal{U}_i$ secrectly and $\mathsf{uvk}[i]$ is made public. Finally, $\mathcal{B}$ honestly generates the master secret key $\mathsf{S}_{\mathsf{OA}} = \mathsf{T}_{\mathbf{B}}$ of the opening authority OA by running $\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{B}, \mathsf{T}_{\mathbf{B}})$. The simulator $\mathcal{B}$ sets the group public key

$$\mathcal{Y} = (\mathbf{M}_0, \mathbf{M}_1, \mathbf{A}, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{B}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}, \mathsf{OTS}, \mathsf{DSig}, H, H_0, \beta, \gamma, \sigma)$$

and initializes the public state $\mathsf{St} = (\mathsf{St}_{\mathsf{users}},\ \mathsf{St}_{\mathsf{trans}}) = (\emptyset, \epsilon)$.

• ***Query Phase***: The simulator $\mathcal{B}$ answers to the queries $\mathsf{Q}_{\mathsf{pub}}$, $\mathsf{Q}_{\mathsf{read}}$, $\mathsf{Q}_{\mathsf{keyOA}}$, hash and $\mathsf{Q}_{\mathsf{a-join}}$ made by $\mathcal{A}$ as follows:

- $\mathsf{Q_{pub}}$: In response to this query, $\mathcal{B}$ returns $\mathcal{Y}$ to $\mathcal{A}$.
- $\mathsf{Q_{read}}$: On this query, $\mathcal{A}$ gets access to the current state St.
- $\mathsf{Q_{keyOA}}$: This query enables the adversary $\mathcal{A}$ to receive the secret key $\mathsf{T}_{\mathbf{B}}$ of the opening authority from $\mathcal{B}$.
- Hash Queries: To respond hash queries, $\mathcal{B}$ maintains two lists for the hash functions H and H_0. These lists store records of the form $(\mathbf{x}, H(\mathbf{x}))$ and $(\mathbf{y}, H_0(\mathbf{y}))$ for some $\mathbf{x}, \mathbf{y} \in \{0,1\}^*$. Fresh hash value is generated on a string $\mathbf{x} \in \{0,1\}^*$ if it is not already been queried and stored in the corresponding hash list. Otherwise, responses are given utilizing the respective hash lists.
- $\mathsf{Q_{a-join}}$: Simulation of this query by $\mathcal{B}$ is as follows:
  - (i) To introduce a malicious user $\mathcal{U}_i \in U^a$, the adversary $\mathcal{A}$ chooses a vector $\mathbf{v}_i^{(0)} \in \mathbb{Z}_q^n$ and computes $\mathsf{sig}_i = \mathsf{DSig.Sign}(\mathsf{usk}[i], \mathbf{v}_i^{(0)})$. Here $\mathsf{usk}[i]$ is the signing key of user $\mathcal{U}_i$ issued by $\mathcal{B}$ and as user $\mathcal{U}_i$ is adversarially controlled, $\mathcal{A}$ has the knowledge of $\mathsf{usk}[i]$. The adversary $\mathcal{A}$ sends $(\mathbf{v}_i^{(0)}, \mathsf{sig}_i)$ to $\mathcal{B}$.
  - (ii) To answer $\mathcal{A}$'s query, $\mathcal{B}$ acts as the GM. The simulator $\mathcal{B}$ proceeds for the following steps only if $\mathsf{DSig.Verify}(\mathsf{uvk}[i], \mathsf{sig}_i, \mathbf{v}_i^{(0)}) = 1$ where $\mathsf{uvk}[i]$ is the verification key of user $\mathcal{U}_i$.
    - (a) It chooses a fresh μ-bit identifier $\mathsf{id}_i \in \{0,1\}^\mu$ such that $\mathsf{id}_i \neq \widehat{\mathsf{id}}$ and a time interval $[t_1, t_2]$ where $t_1, t_2 \in \{1, 2, \ldots, T\}$.
    - (b) It sets the matrix $\mathbf{A}_{\mathsf{id}_i}$ as

$$\mathbf{A}_{\mathsf{id}_i} = \left[\bar{\mathbf{A}}_1 | \bar{\mathbf{A}}_1 \left(\mathbf{L}_0 + \sum_{k=0}^{\mu} \mathsf{id}_i[k]\mathbf{L}_k\right) + \mathbf{h}_{\mathsf{id}_i}.\mathbf{L}\right]$$

where

$$\mathbf{h}_{\mathsf{id}_i} = \sum_{k=1}^{\mu} (\widehat{\mathsf{id}}[k] + (-1)^{\widehat{\mathsf{id}}[k]}\mathsf{id}_i[k]) \qquad (18)$$

is the Hamming distance between id_i and $\widehat{\mathsf{id}}$ as

$$\widehat{\mathsf{id}}[k] + (-1)^{\widehat{\mathsf{id}}[k]}\mathsf{id}_i[k] = \begin{cases} 1, & \widehat{\mathsf{id}}[k] \neq \mathsf{id}_i[k] \\ 0, & \widehat{\mathsf{id}}[k] = \mathsf{id}_i[k] \end{cases}$$

The matrix $\mathbf{A}_{\mathsf{id}_i}$ has the same distribution as in the real protocol since

$$\begin{aligned}
\mathbf{A}_{\mathsf{id}_i} &= \left[\bar{\mathbf{A}}_1|\bar{\mathbf{A}}_1\left(\mathbf{L}_0+\sum_{k=0}^{\mu}\mathsf{id}_i[k]\mathbf{L}_k\right)+\mathbf{h}_{\mathsf{id}_i}.\mathbf{L}\right]\\
&= \left[\bar{\mathbf{A}}_1|\bar{\mathbf{A}}_1.\left(\mathbf{L}_0+\sum_{k=0}^{\mu}\mathsf{id}_i[k]\mathbf{L}_k\right)+\sum_{k=0}^{\mu}\left(\widehat{\mathsf{id}}[k]+(-1)^{\widehat{\mathsf{id}}[k]}\mathsf{id}_i[k]\right)\cdot\mathbf{L}\right]\\
&= \left[\bar{\mathbf{A}}_1|\left(\bar{\mathbf{A}}_1.\mathbf{L}_0+\sum_{k=0}^{\mu}\widehat{\mathsf{id}}[k]\cdot\mathbf{L}\right)+\sum_{k=0}^{\mu}\mathsf{id}_i[k]\left(\bar{\mathbf{A}}_1\mathbf{L}_k+(-1)^{\widehat{\mathsf{id}}[k]}\cdot\mathbf{L}\right)\right]\\
&= \left[\mathbf{A}|\mathbf{A}_0+\sum_{k=1}^{\mu}\mathsf{id}_i[k]\mathbf{A}_k\right]
\end{aligned}$$

The simulator $\mathcal{B}$ finds the basis $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ of $\Lambda_q^{\perp}(\mathbf{A}_{\mathsf{id}_i})=\{\mathbf{x}:\mathbf{A}_{\mathsf{id}_i}\mathbf{x}=0 \bmod q\}$ for $\mathsf{id}_i\neq\widehat{\mathsf{id}}$ using the knowledge of $\mathsf{T}_{\mathbf{L}}$ as follows. Note that

$$\begin{aligned}
\mathbf{x}\in\Lambda_q^{\perp}(\mathbf{L}) &\Rightarrow \mathbf{L}\mathbf{x}=\mathbf{0} \bmod q\\
&\Rightarrow \mathbf{h}_{\mathsf{id}_i}\mathbf{L}\mathbf{x}=\mathbf{0} \bmod q \Rightarrow \mathbf{x}\in\Lambda_q^{\perp}(\mathbf{h}_{\mathsf{id}_i}\mathbf{L})
\end{aligned}$$

Therefore, the basis $\mathsf{T}_{\mathbf{L}}$ of $\Lambda_q^{\perp}(\mathbf{L})=\{\mathbf{x}:\mathbf{L}\mathbf{x}=\mathbf{0} \bmod q\}$ serves as a basis $\mathsf{T}_{\mathbf{h}_{\mathsf{id}_i}\mathbf{L}}$ of $\Lambda_q^{\perp}(\mathbf{h}_{\mathsf{id}_i}\mathbf{L})=\{\mathbf{x}:\mathbf{h}_{\mathsf{id}_i}\mathbf{L}\mathbf{x}=\mathbf{0} \bmod q\}$.

It runs the randomized algorithm $\mathsf{SampleRight}(\bar{\mathbf{A}}_1,\mathbf{h}_{\mathsf{id}_i}\mathbf{L},(\mathbf{L}_0+\sum_{k=0}^{\mu}\mathsf{id}_i[k]\mathbf{L}_k),$ $\mathsf{T}_{\mathbf{h}_{\mathsf{id}_i}\mathbf{L}},\sigma,0)\rightarrow\mathbf{b}\in\mathbb{Z}^{2m}$ as in Lemma 1.1$(b)(v)$ of Section 2 where $\mathbf{b}\in\mathbb{Z}^{2m}$ satisfies

$$\left[\bar{\mathbf{A}}_1|\bar{\mathbf{A}}_1\left(\mathbf{L}_0+\sum_{k=0}^{\mu}\mathsf{id}_i[k]\mathbf{L}_k\right)+\mathbf{h}_{\mathsf{id}_i}\mathbf{L}\right]\mathbf{b}=0 \bmod q \text{ i.e., } \mathbf{A}_{\mathsf{id}_i}\mathbf{b}=0 \bmod q$$

A maximal set of linearly independent such vectors $\mathbf{b}\in\mathbb{Z}^{2m}$ forms a basis of $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$.

(c) Next as in the original protocol $\mathcal{B}$ samples $\mathbf{s}_i\hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and uses $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ to compute a short vector $\mathbf{d}_i=[\mathbf{d}_{i,1}|\mathbf{d}_{i,2}]^t\in\mathbb{Z}^{2m}$, with $||\mathbf{d}_i||_2\leq\sigma\sqrt{2m}$, $\mathbf{d}_{i,1},\ \mathbf{d}_{i,2}\in\mathbb{Z}^m$ such that $\mathbf{A}_{\mathsf{id}_i}\mathbf{d}_i=\mathbf{u}+\mathbf{D}\cdot\mathbf{r}_i \bmod q$ where $\mathbf{r}_i=\mathsf{bin}(\mathbf{D}_0\mathsf{bin}(\mathbf{v}_i^{(0)})+\mathbf{D}_1\cdot\mathbf{s}_i)$

(d) Also for each $w=(w_{\mathsf{num}},w_{\mathsf{dep}})\in\mathsf{SubsetHN}\hookleftarrow\mathsf{Nodes}(t_1+1,t_2,G)$, the simulator $\mathcal{B}$ computes the matrix

$$\mathbf{A}_{i,w}=[\mathbf{A}_{\mathsf{id}_i}|\bar{\mathbf{A}}_w] \text{ with } \bar{\mathbf{A}}_w=\left[\mathbf{M}_0+\mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1|\mathbf{A}_0+\sum_{k=1}^{\mu}\alpha_k\mathbf{A}_k\right]$$

Here G is a binary tree of height l with $T=2^l$ leaves indicating time periods, $w_{\mathsf{num}}\in\{0,1\}^*$ is the label of the node w, w_{dep} is its depth where $0\leq w_{\mathsf{dep}}\leq l$,

and $\alpha_1\alpha_2...\alpha_\mu = 0^{\mu-\eta}||\mathsf{bin}(w_{\mathsf{dep}})$, $\alpha_k \in \{0,1\}$, $\eta = \mathsf{len}(\mathsf{bin}(w_{\mathsf{dep}}))$ denotes the length of $\mathsf{bin}(w_{\mathsf{dep}})$ and $\mathsf{wt}(w_{\mathsf{num}})$ represents the Hamming weight of w_{num}.
As in the real protocol, $\mathcal{B}$ samples $\mathbf{s}_{i,w}^{(0)} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and uses $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ to compute a short vector $\mathbf{x}_{i,w}^{(0)} = \left[\mathbf{x}_{i,w,1}^{(0)}|\mathbf{x}_{i,w,2}^{(0)}|\mathbf{x}_{i,w,3}^{(0)}\right]^t \in \mathbb{Z}^{4m}$, $||\mathbf{x}_{i,w}^{(0)}||_2 \leq \sigma\sqrt{4m}$, $\mathbf{x}_{i,w,1}^{(0)} \in \mathbb{Z}^{2m}$, $\mathbf{x}_{i,w,2}^{(0)}, \mathbf{x}_{i,w,3}^{(0)} \in \mathbb{Z}^m$ such that

$$\mathbf{A}_{i,w}\mathbf{x}_{i,w}^{(0)} = \mathbf{u} + \mathbf{D} \cdot \mathbf{m}_{i,w}$$

where

$$\mathbf{m}_{i,w} = \mathsf{bin}(\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_i^{(0)}) + \mathbf{D}_1 \cdot \mathbf{s}_{i,w}^{(0)}) \bmod q$$

(e) For the delegation purpose, $\mathcal{B}$ uses the algorithm $\mathsf{SampleRwithBasis}(\mathbf{A}) \to (\mathbf{R}_i^{(0)}, \mathsf{T}_{\mathbf{C}_i^{(0)}})$ described in Lemma (1.1)$b(vii)$ of Section 2 where $H_2(\mathsf{id}_i||0) = \mathbf{R}_i^{(0)} \in \mathbb{Z}_q^{m\times m}$, $\mathbf{C}_i^{(0)} = \mathbf{A}(\mathbf{R}_i^{(0)})^{-1}$ and $\mathsf{T}_{\mathbf{C}_i^{(0)}}$ is the basis of the lattice $\Lambda_q^{\perp}(\mathbf{C}_i^{(0)}) = \{\mathbf{x} : \mathbf{C}_i^{(0)}\mathbf{x} = 0 \bmod q\}$.

(iii) The simulator $\mathcal{B}$ issues the certificate,

$$\mathsf{cert}_{i,t_1\to t_2} = \left(\mathsf{id}_i, \mathbf{d}_i, \mathbf{s}_i, \{\mathbf{x}_{i,w}^{(0)}, \mathbf{s}_{i,w}^{(0)}\}_{w\in\mathsf{SubsetHN}\leftarrow\mathsf{Nodes}(t_1+1,t_2,G)}, \mathbf{C}_i^{(0)}, \mathsf{T}_{\mathbf{C}_i^{(0)}}, [t_1,t_2]\right)$$

to the adversary $\mathcal{A}$ as a response to $\mathsf{Q}_{\mathsf{a-join}}$ query.
The simulator $\mathcal{B}$ finally updates the public state St by storing i in $\mathsf{St}_{\mathsf{users}}$ and $\mathsf{transcript}_i = (\mathbf{v}_i^{(0)}, i, \mathsf{uvk}[i], \mathsf{sig}_i, [t_1,t_2])$ in $\mathsf{St}_{\mathsf{trans}}$.

- ***Forgery*:** At the end, $\mathcal{A}$ outputs the forgery (M^*, σ^*, t^*) for a user $\mathcal{U} \notin U^a$ where

$$\sigma^* = (\mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \Pi^*, \mathsf{osig}^*, \mathbf{C}_R^*, [t_1^*, t_2^*])$$

such that $\mathsf{FSGS.Verify}(\sigma^*, t^*, M^*, \mathcal{Y}) = 1$ with $t^* \in [t_1^*, t_2^*]$ where

VK^*	:	verification key generated by $\mathsf{OTS}\cdot\mathcal{G}$.
$\mathbf{c}_{\mathbf{v}^*}$	:	ciphertext generated by running the algorithm $\mathsf{GPV\text{-}IBE}\cdot\mathsf{Enc}(\mathbf{B}, \mathsf{VK}^*, \mathbf{y}^*) \to$
	:	$\mathbf{c}_{\mathbf{v}^*} \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{2m}$ where $\mathbf{B}$ is extracted from $\mathcal{Y}$ and $\mathbf{y}^* = \mathsf{bin}(\mathbf{v}^*) \in \{0,1\}^{2m}$
	:	for some $\mathbf{v}^* \in \mathbb{Z}_q^{2m}$.
Π^*	:	zero knowledge proof for $\mathbf{Px} = \mathbf{v} \bmod q$ equivalent to the system of equation
	:	similar to Eq 12.
osig^*	:	one time signature generated by running $\mathsf{OTS}.\mathcal{S}(\mathsf{SK}^*, \mathbf{c}_{\mathbf{v}^*}, \Pi^*)$ where SK^* is the
	:	signing key corresponding to the verification key VK^* generated by $\mathsf{OTS}\cdot\mathcal{G}$.
$[t_1^*, t_2^*]$	:	a subset of time interval $[1, T]$.

We explain below how $\mathcal{B}$ can construct a knowledge extractor for Π^* using the improved forking lemma stated in Theorem 2.3 in Section 2.5.

Parse the proof of knowledge Π^* as $(\{\mathsf{COM}_l^*\}_{l=1}^s, \mathsf{Ch}^*, \{\mathsf{RSP}_l^*\}_{l=1}^s)$ where $\mathsf{Ch}^* = H(M^*, \mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \{\mathsf{COM}_l^*\}_{l=1}^s) \in \{1,2,3\}^s$. With high probability, $\mathcal{A}$ must have invoked random oracle H and with probability atleast $\epsilon' = \epsilon - 3^{-s}$, the tuple $(M^*, \mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \{\mathsf{COM}_l^*\}_{l=1}^s)$ coincides with the k^*-th random oracle query of H for some $k^* \leq Q_H$ where Q_H is the number of H-queries of $\mathcal{A}$ and ϵ is the non-negligible advantage of $\mathcal{A}$ in breaking our FSGS group signature scheme. Following the improved forking lemma, $\mathcal{B}$ runs $\mathcal{A}$ upto $(1+24Q_H l \log(2l))/\epsilon'$ times with $l = 2$ and $\epsilon' = 3(\epsilon - 3^{-s})$. The adversary $\mathcal{A}$ will receive the same answer from the list corresponding to H maintained by $\mathcal{B}$ for the initial run of first $k^* - 1$ queries of H with the same random tape and input. From k^*-th query onwards, $\mathcal{A}$ will get fresh values of H. The improved forking lemma ensures that $\mathcal{B}$ can obtain a $(l+1)$-fork i.e., 3-fork involving the same tuple $(M^*, \mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \{\mathsf{COM}_l^*\}_{l=1}^s)$ with pairwise *distinct* answers $\mathsf{Ch}^*, \widehat{\mathsf{Ch}}^*, \widetilde{\mathsf{Ch}}^* \in \{1,2,3\}^s$ with probability atleast 1/2. This in turn implies that there exists atleast one index $j \in \{1, 2, ..., s\}$ for which the j-th bits of $\mathsf{Ch}^*, \widehat{\mathsf{Ch}}^*, \widetilde{\mathsf{Ch}}^*$ differ with probability $1 - (21/27)^s = 1 - (7/9)^s$. Consequently, $\mathcal{B}$ has three distinct responses on the same tuple, indicating existence of a knowledge extractor Π^* for $\mathcal{B}$. From the corresponding responses, $\mathcal{B}$ is able to extract witnesses $(\mathbf{d}_1^*, \mathbf{d}_2^*) \in \mathbb{Z}^m \times \mathbb{Z}^m, \mathsf{id}^* \in \{0,1\}^\mu$, $\mathbf{r}^* \in \{0,1\}^m$ from the proof of knowledge Π^* with $||\mathbf{d}_1^*||_2 \leq \sigma\sqrt{m}$, $||\mathbf{d}_2^*||_2 \leq \sigma\sqrt{m}$, $||\mathbf{r}^*||_2 \leq \sigma\sqrt{m}$ satisfying

$$\begin{aligned} \mathbf{A}_{\mathsf{id}^*}\begin{bmatrix}\mathbf{d}_1^* \\ \mathbf{d}_2^*\end{bmatrix} &= \mathbf{u} + \mathbf{D}\cdot\mathbf{r}^* \bmod q \qquad (19)\\ \text{where} \qquad \mathbf{r}^* &= \mathsf{bin}(\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}^*) + \mathbf{D}_1 \cdot \mathbf{s}^* \end{aligned}$$

The simulator $\mathcal{B}$ declares failure if either (i) id^* is any user in U^a or (ii) $\mathsf{id}^* \neq \widehat{\mathsf{id}}$. We denote the event by fail when any one of the above circumstances occur. With a prediction that fail does not occur, $\mathcal{B}$ can solve the given SIS instance as follows:

As $\mathcal{B}$ samples the short vector $\mathbf{h}_u \hookleftarrow D_{\mathbb{Z}^m,\sigma}$, in setting the vector $\mathbf{u}$ as $\mathbf{u} = \bar{\mathbf{A}}_1 \cdot \mathbf{h}_u \bmod q$, and $\{\mathbf{L}_k\}_{k=0}^{\mu}$, $\mathbf{L}_\mathbf{D}$ are sampled by $\mathcal{B}$ itself in the Setup phase, it can compute the vector

$$\mathbf{h} = \mathbf{d}_1^* + (\mathbf{L}_0 + \sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\mathbf{L}_k)\mathbf{d}_2^* - \mathbf{L}_D \cdot \mathbf{r}^* - \mathbf{h}_u \in \mathbb{Z}^m$$

where $\mathbf{d}_1^*, \mathbf{d}_2^*$ and $\mathbf{r}^*$ are extracted by $\mathcal{B}$ from the knowledge extractor. Then we have

$$\begin{aligned}
\bar{\mathbf{A}}_1\mathbf{h} \quad &= \bar{\mathbf{A}}_1\mathbf{d}_1^* + \left(\bar{\mathbf{A}}_1\mathbf{L}_0 + \sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k]\bar{\mathbf{A}}_1\mathbf{L}_k\right)\mathbf{d}_2^* - \bar{\mathbf{A}}_1\mathbf{L}_{\mathbf{D}}\mathbf{r}^* - \bar{\mathbf{A}}_1\mathbf{h}_u \\
&= \bar{\mathbf{A}}_1\mathbf{d}_1^* + \left\{\left(\mathbf{A}_0 - \sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k]\cdot\mathbf{L}) + \sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k](\mathbf{A}_k - (-1)^{\widehat{\mathsf{id}}[k]}\mathbf{L}\right)\right\}\mathbf{d}_2^* - \mathbf{D}\mathbf{r}^* - \mathbf{u} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{(from Eq 15, Eq 16, and Eq 17)} \\
&= \bar{\mathbf{A}}_1\mathbf{d}_1^* + \mathbf{A}_0\mathbf{d}_2^* + \sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k]\mathbf{A}_k\mathbf{d}_2^* - \sum_{k=1}^{\mu}\left(\widehat{\mathsf{id}}[k] + (-1)^{\widehat{\mathsf{id}}[k]}\widehat{\mathsf{id}}[k]\right)\mathbf{L}\mathbf{d}_2^* - \mathbf{D}\mathbf{r}^* - \mathbf{u} \\
&= \bar{\mathbf{A}}_1\mathbf{d}_1^* + \mathbf{A}_0\mathbf{d}_2^* + \sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k]\mathbf{A}_k\mathbf{d}_2^* - \mathbf{h}_{\widehat{\mathsf{id}}}\,\mathbf{L}\mathbf{d}_2^* - \mathbf{D}\mathbf{r}^* - \mathbf{u} \quad \text{(from Eq 18)} \\
&= \bar{\mathbf{A}}_1\mathbf{d}_1^* + \mathbf{A}_0\mathbf{d}_2^* + \sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k]\mathbf{A}_k\mathbf{d}_2^* - \mathbf{D}\mathbf{r}^* - \mathbf{u} \\
&= \mathbf{A}\mathbf{d}_1^* + \mathbf{A}_0\mathbf{d}_2^* + \sum_{k=1}^{\mu}\mathsf{id}^*[k]\mathbf{A}_k\mathbf{d}_2^* - \mathbf{D}\mathbf{r}^* - \mathbf{u} \\
&= [\mathbf{A}|\mathbf{A}_0 + \sum_{k=1}^{\mu}\mathsf{id}^*[k]\mathbf{A}_k]\begin{bmatrix}\mathbf{d}_1^* \\ \hline \mathbf{d}_2^*\end{bmatrix} - \mathbf{D}\mathbf{r}^* - \mathbf{u} \\
&= \mathbf{A}_{\mathsf{id}^*}\begin{bmatrix}\mathbf{d}_1^* \\ \hline \mathbf{d}_2^*\end{bmatrix} - \mathbf{D}\mathbf{r}^* - \mathbf{u} \\
&= 0 \bmod q \quad (\text{ from Eq 19})
\end{aligned}$$

Thus $\mathbf{h}$ is a short nonzero vector satisfying $\bar{\mathbf{A}}_1.\mathbf{h} = 0 \bmod q$ with

$$\begin{aligned}
||\mathbf{h}||_2 &\le ||\mathbf{d}_1^* + (\mathbf{L}_0 + \sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k]\mathbf{L}_k)\mathbf{d}_2^* - \mathbf{L}_D.\mathbf{r}^* - \mathbf{h}_u||_2 \\
&\le ||\mathbf{d}_1^*||_2 + ||\mathbf{L}_0 + \textstyle\sum_{k=1}^{\mu}\widehat{\mathsf{id}}[k]\mathbf{L}_k||_2 \cdot ||\mathbf{d}_2^*||_2 + ||\mathbf{L}_D||_2 \cdot ||\mathbf{r}^*||_2 + ||\mathbf{h}_u||_2 \\
&\le \sigma\sqrt{m} + \{m \times (\mu+1)\sigma\sqrt{m}\}\sigma\sqrt{m} + (m \times \sigma\sqrt{m})\sigma\sqrt{m} + \sigma\sqrt{m} \\
&\le 2\sigma\sqrt{m} + (\mu+2)m^2\sigma^2
\end{aligned}$$

Furthermore, $(\mathbf{h}^t|0^m)^t$ is a short non-zero vector in $\Lambda_q^{\perp}(\bar{\mathbf{A}})$ and hence serves as a solution of the given SIS instance.

$\langle B\rangle$ $\mathsf{Case\ II}$($\mathsf{id}^* = \widehat{\mathsf{id}}$ and id^* belongs to certain group member in U^a but $t^* \notin [t_1, t_2]$ where $[t_1, t_2]$ is the time period for which $\widehat{\mathsf{id}}$ has been issued certificate)

Subcase I($\mathbf{r}^* \neq \mathbf{r}_{i^*}$): The simulator $\mathcal{B}$ performs the following steps to set the group public key $\mathcal{Y}$ using the given SIS instance $\bar{\mathbf{A}} = [\bar{\mathbf{A}}_1|\bar{\mathbf{A}}_2]$, $\bar{\mathbf{A}}_1, \bar{\mathbf{A}}_2 \in \mathbb{Z}_q^{n\times m}$, $m = 2n\lceil\log q\rceil$ and pre selected $\widehat{\mathsf{id}} \in \{0,1\}^\mu$, $i^* \in U([1,\delta])$, $\widehat{t} \in [1,T]$.

Setup: The simulator $\mathcal{B}$ chooses an interval $[t_1,t_2] \subset [1,T]$ such that $\widehat{t} \in [t_1,t_2]$, picks $\delta - 1$ distinct identifiers $\mathsf{id}_1, \mathsf{id}_2, \ldots, \mathsf{id}_{i^*-1}, \mathsf{id}_{i^*+1}, \ldots, \mathsf{id}_\delta \in \{0,1\}^\mu$ and sets $\mathsf{id}_{i^*} = \widehat{\mathsf{id}}$. It also chooses randomly $d_0, d_1, \ldots, d_\mu \in \mathbb{Z}_q$ such that

$$d_{\mathsf{id}_i} = d_0 + \sum_{k=1}^{\mu} \mathsf{id}_i[k].d_k = 0 \bmod q \text{ iff } i = i^*, \text{ for } i \in \{1,2,\ldots,\delta\} \tag{20}$$

This in turn implies that $d_{\widehat{\mathsf{id}}} = d_{\mathsf{id}_{i^*}} = 0 \bmod q$. Here $\mathsf{id}_i = \mathsf{id}_i[1]\mathsf{id}_i[2]\ldots\mathsf{id}_i[\mu]$ will be utilized to respond the $\mathsf{Q}_{\mathsf{a-join}}$ query for the adversarially controlled user $\mathcal{U}_i \in \{1,2,\ldots,\delta\}$. Next $\mathcal{B}$ runs $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{C}, \mathsf{T}_C)$, $\mathsf{TrapGen}(1^{2n}, 1^{2m}, q) \to (\mathbf{D}_1, \mathsf{T}_{\mathbf{D}_1})$, $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{B}, \mathsf{T}_\mathbf{B})$. It samples matrices $\mathbf{D}_0 \hookleftarrow U(\mathbb{Z}_q^{2n\times 2m})$, $\mathbf{M}_0, \mathbf{M}_1 \hookleftarrow U(\mathbb{Z}_q^{n\times m})$. It also selects Gaussian matrices $\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_\mu \hookleftarrow \mathbb{Z}^{m\times m}$ whose columns are sampled from the distribution $D_{\mathbb{Z}^m,\sigma}$ where σ is of size $\Omega(\sqrt{n\log q}\log n)$, also selects $\mathbf{S} \hookleftarrow U(\mathbb{Z}^{m\times m})$ such that $\mathbf{S}^{-1}$ has low norm and sets the matrices $\mathbf{D}, \mathbf{A}, \mathbf{A}_0, \{\mathbf{A}_k\}_{k=1}^{\mu}$ as

$\mathbf{D} = \bar{\mathbf{A}}_1$
$\mathbf{A} = \bar{\mathbf{A}}_1 \cdot \mathbf{S}$
$\mathbf{A}_0 = \bar{\mathbf{A}}_1 \cdot \mathbf{S}_0 + d_0 \cdot \mathbf{C}$
$\mathbf{A}_k = \bar{\mathbf{A}}_1 \cdot \mathbf{S}_k + d_k \cdot \mathbf{C}$, for $k = 1, 2, \cdots, \mu$.
$\mathbf{M}_0 = (\mathbf{A}_0 + \sum\limits_{k=1}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{A}_k)\mathbf{S}$ and $\mathbf{M}_1 = \mathbf{C}$.

The simulator $\mathcal{B}$ then finds a vector $\mathbf{u} \in \mathbb{Z}_q^n$ as follows:

– It computes $\mathbf{A}_{\mathsf{id}_{i^*}} = \left[\mathbf{A} | \mathbf{A}_0 + \sum\limits_{k=1}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{A}_k\right]$

– Now $\mathcal{B}$ chooses two short vectors $\mathbf{d}_{i^*,1}, \mathbf{d}_{i^*,2} \in D_{\mathbb{Z}^m,\sigma}$, sets $\mathbf{d}_{i^*} = [\mathbf{d}_{i^*,1} | \mathbf{d}_{i^*,2}]^t$ and picks $\mathbf{r}_{i^*} = \mathsf{bin}(\mathbf{c}_M) \in \{0,1\}^m$ and thus it sets

$$\mathbf{u} = \mathbf{A}_{\mathsf{id}_{i^*}}\mathbf{d}_{i^*} - \mathbf{D}\cdot \mathsf{bin}(\mathbf{c}_M) \tag{21}$$

$$i.e., \quad \mathbf{A}_{\mathsf{id}_{i^*}}\mathbf{d}_{i^*} = \mathbf{u} + \mathbf{D}\cdot\mathbf{r}_{i^*} \tag{22}$$

The distribution of $\mathbf{u}$ is statistically close to $U(\mathbb{Z}_q^n)$ since $\mathbf{A}$ is statistically uniform and $\mathbf{d}_{i^*,1} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$ and $\mathbf{d}_{i^*,2} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$.

The remaining parameters $\mathbf{F} \hookleftarrow \mathbb{Z}^{4n\times 4m}$, $H : \{0,1\}^* \to \{1,2,3\}^s$, $H_0 : \{0,1\}^* \to \mathbb{Z}_q^{n\times 2m}$, $H_2 : \{0,1\}^* \to \mathcal{I}$, error bound γ of size $\sqrt{n}\omega(\log n)$, one time signature $\mathsf{OTS} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$, digital signature $\mathsf{DSig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ are generated faithfully as in the original protocol. Also $\mathcal{B}$ runs the key generation algorithm $\mathsf{DSig.KeyGen}$ of a digital signature scheme DSig to generate signing-verification key pair $(\mathsf{usk}[i], \mathsf{uvk}[i])$. The simulator sends

$\mathsf{usk}[i]$, $i \in [N]$ to $\mathcal{U}_i$ secretly and $\mathsf{uvk}[i]$ is made public.
The simulator $\mathcal{B}$ sets the public key

$$\mathcal{Y} = (\mathbf{M}_0, \mathbf{M}_1\mathbf{A}, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{B}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}, \mathsf{OTS}, \mathsf{DSig}, H, H_0, \beta, \gamma, \sigma)$$

and initializes the public state $\mathsf{St} = (\mathsf{St}_{\mathsf{users}},\ \mathsf{St}_{\mathsf{trans}}) = (\emptyset, \epsilon)$.

***Query Phase*(when $(i \neq i^*)$)**: The queries $\mathsf{Q}_{\mathsf{pub}}$, $\mathsf{Q}_{\mathsf{read}}$, $\mathsf{Q}_{\mathsf{keyOA}}$, and hash queries made by $\mathcal{A}$ are answered by $\mathcal{B}$ as in Case I. The response to $\mathsf{Q}_{\mathsf{a-join}}$ query is as follows:

(i) To introduce a malicious user $\mathcal{U}_i \in U^a$, the adversary $\mathcal{A}$ chooses a vector $\mathbf{v}_i^{(0)} \in \mathbb{Z}_q^n$ and computes $\mathsf{sig}_i = \mathsf{DSig.Sign}(\mathsf{usk}[i], \mathbf{v}_i^{(0)})$. Here $\mathsf{usk}[i]$ is the signing key of user $\mathcal{U}_i$ issued by $\mathcal{B}$ and as user $\mathcal{U}_i$ is adversarially controlled, $\mathcal{A}$ has the knowledge of $\mathsf{usk}[i]$. The adversary $\mathcal{A}$ sends $(\mathbf{v}_i^{(0)}, \mathsf{sig}_i)$ to $\mathcal{B}$.

(ii) To answer $\mathcal{A}$'s query, $\mathcal{B}$ acts as the GM and proceeds for the following steps only if $\mathsf{DSig.Verify}(\mathsf{uvk}[i], \mathsf{sig}_i, \mathbf{v}_i^{(0)}) = 1$ where $\mathsf{uvk}[i]$ is the verification key of user $\mathcal{U}_i$.

   (a) It selects a time interval $[t_1, t_2] \subset [1, T]$ and chooses the i-th identifier $\mathsf{id}_i \in \{0,1\}^{\mu}$ from the preselected identifiers in the Setup phase such that $\mathsf{id}_i \neq \mathsf{id}_{i^*}$.

   (b) It sets the matrix $\mathbf{A}_{\mathsf{id}_i}$ as

$$\mathbf{A}_{\mathsf{id}_i} = \left[\bar{\mathbf{A}}_1\mathbf{S}|\bar{\mathbf{A}}_1\mathbf{S}\left\{\mathbf{S}^{-1}\left(\mathbf{S}_0 + \sum_{k=0}^{\mu}\mathsf{id}_i[k]\mathbf{S}_k\right)\right\} + d_{\mathsf{id}_i}.\mathbf{C}\right]$$

   The matrix $\mathbf{A}_{\mathsf{id}_i}$ has the same distribution as in the real protocol since

$$\begin{aligned}
\mathbf{A}_{\mathsf{id}_i} &= \left[\bar{\mathbf{A}}_1\mathbf{S}|\bar{\mathbf{A}}_1\mathbf{S}\left\{\mathbf{S}^{-1}\left(\mathbf{S}_0 + \sum_{k=0}^{\mu}\mathsf{id}_i[k]\mathbf{S}_k\right)\right\} + d_{\mathsf{id}_i}.\mathbf{C}\right] \\
&= \left[\mathbf{A}|\bar{\mathbf{A}}_1.\left(\mathbf{S}_0 + \sum_{k=0}^{\mu}\mathsf{id}_i[k]\mathbf{S}_k\right) + \left(d_0 + \sum_{k=1}^{\mu}\mathsf{id}_i[k]d_k\right)\mathbf{C}\right] \\
&= \left[\mathbf{A}|\bar{\mathbf{A}}_1.\mathbf{S}_0 + d_0 \cdot \mathbf{C} + \sum_{k=1}^{\mu}\mathsf{id}_i[k](\bar{\mathbf{A}}_1\mathbf{S}_k + d_k\mathbf{C})\right] \\
&= \left[\mathbf{A}|\mathbf{A}_0 + \sum_{k=1}^{\mu}\mathsf{id}_i[k]\mathbf{A}_k\right]
\end{aligned}$$

   The simulator $\mathcal{B}$ finds the basis $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ of $\Lambda_q^{\perp}(\mathbf{A}_{\mathsf{id}_i}) = \{\mathbf{x} : \mathbf{A}_{\mathsf{id}_i}\mathbf{x} = 0 \bmod q\}$ for $\mathsf{id}_i \neq \widehat{\mathsf{id}}$ using the knowledge of $\mathsf{T}_{\mathbf{C}}$ as follows. Note that

$$\begin{aligned}
\mathbf{x} \in \Lambda_q^{\perp}(\mathbf{C}) &\Rightarrow \mathbf{C}\mathbf{x} = \mathbf{0} \bmod q \\
&\Rightarrow d_{\mathsf{id}_i}\mathbf{C}\mathbf{x} = \mathbf{0} \bmod q \Rightarrow \mathbf{x} \in \Lambda_q^{\perp}(d_{\mathsf{id}_i}\mathbf{C})
\end{aligned}$$

Therefore, the basis $\mathsf{T}_{\mathbf{C}}$ of $\Lambda_q^{\perp}(\mathbf{C}) = \{\mathbf{x} : \mathbf{C}\mathbf{x} = \mathbf{0} \bmod q\}$ serves as a basis $\mathsf{T}_{d_{\mathsf{id}_i}\mathbf{C}}$ of $\Lambda_q^{\perp}(d_{\mathsf{id}_i}\mathbf{C}) = \{\mathbf{x} : d_{\mathsf{id}_i}\mathbf{C}\mathbf{x} = \mathbf{0} \bmod q\}$.

It runs the randomized algorithm $\mathsf{SampleRight}\big(\bar{\mathbf{A}}_1\mathbf{S}, d_{\mathsf{id}_i}\mathbf{C}, \mathbf{S}^{-1}\big(\mathbf{S}_0 + \sum_{k=0}^{\mu} \mathsf{id}_i[k]\mathbf{S}_k\big),$ $\mathsf{T}_{d_{\mathsf{id}_i}\mathbf{C}}, \sigma, 0\big) \to \mathbf{b} \in \mathbb{Z}^{2m}$ as in Lemma 1.1$(b)(v)$ of Section 2 where $\mathbf{b} \in \mathbb{Z}^{2m}$ satisfies

$$\left[\bar{\mathbf{A}}_1 | \bar{\mathbf{A}}_1 \left(\mathbf{S}_0 + \sum_{k=0}^{\mu} \mathsf{id}_i[k]\mathbf{S}_k\right) + d_{\mathsf{id}_i}\mathbf{C}\right]\mathbf{b} = 0 \bmod q \ \text{ i.e., } \mathbf{A}_{\mathsf{id}_i}\mathbf{b} = 0 \bmod q$$

A maximal set of linearly independent such vectors $\mathbf{b} \in \mathbb{Z}^{2m}$ forms a basis of $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$.

(c) Next as in the original protocol $\mathcal{B}$ samples $\mathbf{s}_i \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and uses $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ to compute a short vector $\mathbf{d}_i = [\mathbf{d}_{i,1}|\mathbf{d}_{i,2}]^t \in \mathbb{Z}^{2m}$, with $||\mathbf{d}_i||_2 \le \sigma\sqrt{2m}$, $\mathbf{d}_{i,1}, \ \mathbf{d}_{i,2} \in \mathbb{Z}^m$ such that $\mathbf{A}_{\mathsf{id}_i}\mathbf{d}_i = \mathbf{u} + \mathbf{D}\cdot\mathbf{r}_i \bmod q$ where $\mathbf{r}_i = \mathsf{bin}(\mathbf{D}_0\mathsf{bin}(\mathbf{v}_i^{(0)}) + \mathbf{D}_1\cdot\mathbf{s}_i)$

(d) Also for each $w = (w_{\mathsf{num}}, w_{\mathsf{dep}}) \in \mathsf{SubsetHN} \hookleftarrow \mathsf{Nodes}(t_1+1, t_2, G)$, the simulator $\mathcal{B}$ computes the matrix

$$\mathbf{A}_{i,w} = [\mathbf{A}_{\mathsf{id}_i}|\bar{\mathbf{A}}_w] \text{ with } \bar{\mathbf{A}}_w = \left[\mathbf{M}_0 + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1 | \mathbf{A}_0 + \sum_{k=1}^{\mu}\alpha_k\mathbf{A}_k\right]$$

Here G is a binary tree of height l with $T = 2^l$ leaves indicating time periods, $w_{\mathsf{num}} \in \{0,1\}^*$ is the label of the node w, w_{dep} is the depth of w with $0 \le w_{\mathsf{dep}} \le l$, and $\alpha_1\alpha_2...\alpha_\mu = 0^{\mu-\eta}||\mathsf{bin}(w_{\mathsf{dep}})$, $\alpha_k \in \{0,1\}$, $\eta = \mathsf{len}(\mathsf{bin}(w_{\mathsf{dep}}))$ denotes the length of $\mathsf{bin}(w_{\mathsf{dep}})$ and $\mathsf{wt}(w_{\mathsf{num}})$ represents the Hamming weight of w_{num}.

As in the real protocol, $\mathcal{B}$ samples $\mathbf{s}_{i,w}^{(0)} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and uses $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_i}}$ to compute a short vector $\mathbf{x}_{i,w}^{(0)} = \left[\mathbf{x}_{i,w,1}^{(0)}|\mathbf{x}_{i,w,2}^{(0)}|\mathbf{x}_{i,w,3}^{(0)}\right]^t \in \mathbb{Z}^{4m}$, $||\mathbf{x}_{i,w}^{(0)}||_2 \le \sigma\sqrt{4m}$, $\mathbf{x}_{i,w,1}^{(0)} \in \mathbb{Z}^{2m}$, $\mathbf{x}_{i,w,2}^{(0)}, \mathbf{x}_{i,w,3}^{(0)} \in \mathbb{Z}^m$ such that

$$\mathbf{A}_{i,w}\mathbf{x}_{i,w}^{(0)} = \mathbf{u} + \mathbf{D}\cdot\mathbf{m}_{i,w}$$

where

$$\mathbf{m}_{i,w} = \mathsf{bin}(\mathbf{D}_0\cdot\mathsf{bin}(\mathbf{v}_i^{(0)}) + \mathbf{D}_1\cdot\mathbf{s}_{i,w}^{(0)}) \bmod q.$$

(e) For the delegation purpose, $\mathcal{B}$ uses the algorithm $\mathsf{SampleRwithBasis}(\mathbf{A}) \to (\mathbf{R}_i^{(0)}, \mathsf{T}_{\mathbf{C}_i^{(0)}})$ described in Lemma (1.1)$b(vii)$ of Section 2 where $H_2(\mathsf{id}_i||0) = \mathbf{R}_i^{(0)} \in \mathbb{Z}_q^{m\times m}$, $\mathbf{C}_i^{(0)} = \mathbf{A}(\mathbf{R}_i^{(0)})^{-1}$ and $\mathsf{T}_{\mathbf{C}_i^{(0)}}$ is the basis of the lattice $\Lambda_q^{\perp}(\mathbf{C}_i^{(0)}) = \{\mathbf{x} : \mathbf{C}_i^{(0)}\mathbf{x} = 0 \bmod q\}$.

(iii) The simulator $\mathcal{B}$ issues the certificate,

$$\mathsf{cert}_{i,t_1\to t_2} = \left(\mathsf{id}_i, \mathbf{d}_i, \mathbf{s}_i, \{\mathbf{x}_{i,w}^{(0)}, \mathbf{s}_{i,w}^{(0)}\}_{w\in\mathsf{SubsetHN}\leftarrow\mathsf{Nodes}(t_1+1,t_2,G)}, \mathbf{C}_i^{(0)}, \mathsf{T}_{\mathbf{C}_i^{(0)}}, [t_1, t_2]\right)$$

to the adversary $\mathcal{A}$ as a response to $\mathsf{Q}_{\mathsf{a-join}}$ query and updates the public state St by storing i in $\mathsf{St}_{\mathsf{users}}$ and $\mathsf{transcript}_i = (\mathbf{v}_i^{(0)}, i, \mathsf{uvk}[i], \mathsf{sig}_i, [t_1, t_2])$ in $\mathsf{St}_{\mathsf{trans}}$.

***Query Phase*(when $(i = i^*)$)**: Here $\mathsf{id}_i = \mathsf{id}_{i^*} = \widehat{\mathsf{id}}$.
The queries $\mathsf{Q}_{\mathsf{pub}}$, $\mathsf{Q}_{\mathsf{read}}$, $\mathsf{Q}_{\mathsf{keyOA}}$, and hash queries made by $\mathcal{A}$ are answered by $\mathcal{B}$ as in Case I. The response to $\mathsf{Q}_{\mathsf{a-join}}$ query is as follows:

(i) To introduce the malicious user $\mathcal{U}_{i^*} \in U^a$, the adversary $\mathcal{A}$ chooses a vector $\mathbf{v}_{i^*}^{(0)} \in \mathbb{Z}_q^n$ and computes $\mathsf{sig}_{i^*} = \mathsf{DSig.Sign}(\mathsf{usk}[i^*], \mathbf{v}_{i^*}^{(0)})$ using the signing key $\mathsf{usk}[i^*]$ of user $\mathcal{U}_{i^*}$ issued by $\mathcal{B}$. A user $\mathcal{U}_{i^*}$ is adversarially controlled, $\mathcal{A}$ has the knowledge of $\mathsf{usk}[i^*]$. The adversary $\mathcal{A}$ sends $(\mathbf{v}_{i^*}^{(0)}, \mathsf{sig}_{i^*})$ to $\mathcal{B}$.

(ii) To answer $\mathcal{A}$'s query, $\mathcal{B}$ acts as the GM. The simulator $\mathcal{B}$ proceeds for the following steps only if $\mathsf{DSig.Verify}(\mathsf{uvk}[i^*], (\mathbf{v}_{i^*}^{(0)}, \mathsf{sig}_{i^*})) = 1$ where $\mathsf{uvk}[i^*]$ is the verification key of the user $\mathcal{U}_{i^*}$.

(a) It selects a time interval $[t_1, t_2]$ where $t_1, t_2 \in \{1, 2, \ldots, T\}$ and chooses the i-th identifier $\mathsf{id}_{i^*} \in \{0,1\}^\mu$ such that $\mathsf{id}_{i^*} = \widehat{\mathsf{id}}$ as set in the Setup phase.

(b) It sets the matrix $\mathbf{A}_{\mathsf{id}_{i^*}}$ as

$$\mathbf{A}_{\mathsf{id}_{i^*}} = \left[\bar{\mathbf{A}}_1\mathbf{S}|\bar{\mathbf{A}}_1\left(\mathbf{S}_0 + \sum_{k=0}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{S}_k\right)\right]$$

The matrix $\mathbf{A}_{\mathsf{id}_{i^*}}$ has the same distribution as in the real protocol since

$$\begin{aligned} & \bar{\mathbf{A}}_1\left(\mathbf{S}_0 + \sum_{k=0}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{S}_k\right) \\ = \quad & \bar{\mathbf{A}}_1\left(\mathbf{S}_0 + \sum_{k=0}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{S}_k\right) + d_{\mathsf{id}_{i^*}}.\mathbf{C} \\ = \quad & \bar{\mathbf{A}}_1.\left(\mathbf{S}_0 + \sum_{k=0}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{S}_k\right) + \left(d_0 + \sum_{k=1}^{\mu} \mathsf{id}_{i^*}[k]d_k\right)\mathbf{C} \\ = \quad & \bar{\mathbf{A}}_1.\mathbf{S}_0 + d_0 \cdot \mathbf{C} + \sum_{k=1}^{\mu} \mathsf{id}_{i^*}[k](\bar{\mathbf{A}}_1\mathbf{S}_k + d_k\mathbf{C}) \end{aligned}$$

and consequently,

$$\begin{aligned} \mathbf{A}_{\mathsf{id}_{i^*}} \quad &= \left[\bar{\mathbf{A}}_1\mathbf{S}|\bar{\mathbf{A}}_1.\mathbf{S}_0 + d_0 \cdot \mathbf{C} + \sum_{k=1}^{\mu} \mathsf{id}_{i^*}[k](\bar{\mathbf{A}}_1\mathbf{S}_k + d_k\mathbf{C})\right] \\ &= \left[\mathbf{A}|\mathbf{A}_0 + \sum_{k=1}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{A}_k\right] \end{aligned}$$

(c) As $\mathbf{A}_{\mathsf{id}_{i^*}}$ is independent of $\mathbf{C}$, the simulator $\mathcal{B}$ generates the certificate by making use of the vectors $\mathbf{d}_{i^*} = [\mathbf{d}_{i^*,1}|\mathbf{d}_{i^*,2}]^t$, where $\mathbf{d}_{i^*,1}, \mathbf{d}_{i^*,2} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$, $\mathbf{c}_M \in \mathbb{Z}_q^{2n}$. On receiving $\mathbf{v}_{i^*}^{(0)}$ from $\mathcal{A}$, the simulator $\mathcal{B}$ computes

$$\mathbf{c}_{i^*} = \mathbf{c}_M - \mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_{i^*}^{(0)}) \bmod q.$$

It runs

$$\mathsf{SamplePre}(\mathbf{D}_1, \mathsf{T}_{\mathbf{D}_1}, \mathbf{c}_{i^*}, \sigma) \rightarrow \mathbf{s}_{i^*} \in \Lambda_q^{\mathbf{c}_{i^*}}(\mathbf{D}_1)$$

using $\mathsf{T}_{\mathbf{D}_1}$ to sample short vectors $\mathbf{s}_{i^*} \hookleftarrow D_{\Lambda_q^{\mathbf{c}_{i^*}}(\mathbf{D}_1),\sigma}$ satisfying $\mathbf{D}_1\mathbf{s}_{i^*} = \mathbf{c}_{i^*} \bmod q$ *i.e.*,

$$\mathbf{D}_1.\mathbf{s}_{i^*} = \mathbf{c}_M - \mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_{i^*}^{(0)}) \bmod q.$$

Note that in the Setup phase, we set $\mathbf{r}_{i^*}$ as

$$\mathbf{r}_{i^*} = \mathsf{bin}(\mathbf{c}_M) = \mathsf{bin}(\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_{i^*}^{(0)}) + \mathbf{D}_1\mathbf{s}_{i^*}).$$

(d) Also, $\mathbf{A}' = \left[\mathbf{A}_0 + \sum_{k=1}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{A}_k | \mathbf{M}_0 + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1\right]$, where

$$\mathbf{M}_0 = (\mathbf{A}_0 + \sum_{k=1}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{A}_k)\mathbf{S} \quad \text{and} \quad \mathbf{M}_1 = \mathbf{C}.$$

Note that basis of $\mathbf{M}_1$ is a superset of $\mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1$. Then using $\mathsf{SampleRight}(\mathbf{A}_0 + \sum_{k=1}^{\mu} \mathsf{id}_{i^*}[k]\mathbf{A}_k, \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1, \mathbf{S}, \mathsf{T}_{\mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1}, \sigma, 0) \rightarrow (\mathbf{b} \in \mathbb{Z}_{2m})$. A maximal linearly independent set of such vectors $\mathbf{b} \in \mathbb{Z}_{2m}$ forms a basis of $\mathsf{T}_{\mathbf{A}'}$. The algorithm $\mathsf{ExtBasis}(\mathbf{A}_{\mathsf{id}_{i^*}}, \mathbf{A}', \mathsf{T}_{\mathbf{A}'}) \rightarrow \mathsf{T}_{\mathbf{A}_{\mathsf{id}_{i^*}}|\mathbf{A}'} = \mathsf{T}_{\mathbf{A}_{\mathsf{id}_{i^*}},w'}$.

Now, for each $w = (w_{\mathsf{num}}, w_{\mathsf{dep}}) \in \mathsf{SubsetHN} \hookleftarrow \mathsf{Nodes}(t_1 + 1, t_2, G)$, the simulator $\mathcal{B}$ computes the matrix

$$\mathbf{A}_{i^*,w} = [\mathbf{A}_{\mathsf{id}_{i^*}} | \bar{\mathbf{A}}_w] \text{ with } \bar{\mathbf{A}}_w = \left[\mathbf{M}_0 + \mathsf{wt}(w_{\mathsf{num}})\mathbf{M}_1 | \mathbf{A}_0 + \sum_{k=1}^{\mu} \alpha_k \mathbf{A}_k\right]$$

Here G is a binary tree of height l with $T = 2^l$ leaves indicating time periods, $w_{\mathsf{num}} \in \{0,1\}^*$ is the label of the node w, w_{dep} is the depth of the w with $0 \leq w_{\mathsf{dep}} \leq l$, and $\alpha_1\alpha_2...\alpha_\mu = 0^{\mu-\eta}||\mathsf{bin}(w_{\mathsf{dep}})$, $\alpha_k \in \{0,1\}$, $\eta = \mathsf{len}(\mathsf{bin}(w_{\mathsf{dep}}))$ denotes the length of $\mathsf{bin}(w_{\mathsf{dep}})$ and $\mathsf{wt}(w_{\mathsf{num}})$ represents the Hamming weight of w_{num}.
As in the real protocol, $\mathcal{B}$ samples $\mathbf{s}_{i^*,w}^{(0)} \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and uses $\mathsf{T}_{\mathbf{A}_{\mathsf{id}_{i^*}},w'}$ to compute a short vector $\mathbf{x}_{i^*,w}^{(0)} = \left[\mathbf{x}_{i^*,w,1}^{(0)} | \mathbf{x}_{i^*,w,2}^{(0)} | \mathbf{x}_{i^*,w,3}^{(0)}\right]^t \in \mathbb{Z}^{4m}$, $||\mathbf{x}_{i^*,w}^{(0)}||_2 \leq \sigma\sqrt{4m}$, $\mathbf{x}_{i^*,w,1}^{(0)} \in \mathbb{Z}^{2m}$, $\mathbf{x}_{i^*,w,2}^{(0)}, \mathbf{x}_{i^*,w,3}^{(0)} \in \mathbb{Z}^m$ such that

$$\mathbf{A}_{i^*,w}\mathbf{x}_{i^*,w}^{(0)} = \mathbf{u} + \mathbf{D} \cdot \mathbf{m}_{i^*,w}$$

where

$$\mathbf{m}_{i^*,w} = \mathsf{bin}(\mathbf{D}_0 \cdot \mathsf{bin}(\mathbf{v}_{i^*}^{(0)}) + \mathbf{D}_1 \cdot \mathbf{s}_{i^*,w}^{(0)}) \bmod q.$$

(e) For the delegation purpose, $\mathcal{B}$ finds $\mathbf{R}_{i^*}^{(0)} = H_2(\mathsf{id}_i||0)$ and uses the algorithm $\mathsf{SampleRwithBasis}(\mathbf{A}) \rightarrow (\mathbf{R}_{i^*}^{(0)}, \mathsf{T}_{\mathbf{C}_{i^*}^{(0)}})$ described in Lemma (1.1)$b(vii)$ of Section 2 where $\mathbf{R}_{i^*}^{(0)} \in \mathbb{Z}_q^{m\times m}$, $\mathbf{C}_{i^*}^{(0)} = \mathbf{A}(\mathbf{R}_{i^*}^{(0)})^{-1}$ and $\mathsf{T}_{\mathbf{C}_{i^*}^{(0)}}$ is the basis of the lattice $\Lambda_q^{\perp}(\mathbf{C}_{i^*}^{(0)}) = \{\mathbf{x} : \mathbf{C}_{i^*}^{(0)}\mathbf{x} = 0 \bmod q\}$.

Finally, $\mathcal{B}$ returns the certificate,

$$\mathsf{cert}_{i^*,t_1\to t_2} = \Big(\mathsf{id}_{i^*}, \mathbf{d}_{i^*}, \mathbf{s}_{i^*}, \{\mathbf{x}^{(0)}_{i^*,w}, \mathbf{s}^{(0)}_{i^*,w}\}_{w\in\mathsf{SubsetHN}\leftarrow\mathsf{Nodes}(t_1+1,t_2,G)}, \mathbf{C}^{(0)}_{i^*}, \mathsf{T}_{\mathbf{C}^{(0)}_{i^*}}, [t_1,t_2]\Big)$$

to the adversary $\mathcal{A}$ and updates the public state St by storing i^* in $\mathsf{St}_{\mathsf{users}}$ and $\mathsf{transcript}_{i^*} = (\mathbf{v}^{(0)}_{i^*}, i^*, \mathsf{uvk}[i^*], \mathsf{sig}_{i^*}, [t_1,t_2])$ in $\mathsf{St}_{\mathsf{trans}}$.

***Forgery*:** At the end, $\mathcal{A}$ outputs the forgery (M^*, σ^*, t^*) for a user $\mathcal{U} \notin U^a$ where

$$\sigma^* = (\mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \Pi^*, \mathsf{osig}^*, \mathbf{C}^*_R, [t_1^*, t_2^*])$$

such that $\mathsf{FSGS.Verify}(\sigma^*, t^*, M^*, \mathcal{Y}) = 1$ with $t^* \in [t_1^*, t_2^*]$ but $t^* \notin [t_1, t_2]$ where $[t_1,t_2]$ is the time period for which $\mathsf{id}^* = \widehat{\mathsf{id}}$ has been issued a certificate. The notations VK^*, $\mathbf{c}_{\mathbf{v}^*}$, Π^*, osig^*, $[t_1^*, t_2^*]$ are as described in Case I. As $\mathcal{B}$ can construct a knowledge extractor for the proof of knowledge Π^* as explained in Case I, the simulator $\mathcal{B}$ is able to extract witnesses $(\mathbf{d}_1^*, \mathbf{d}_2^*) \in \mathbb{Z}^m \times \mathbb{Z}^m, \mathsf{id}^* \in \{0,1\}^\mu$, $\mathbf{r}^* \in \{0,1\}^m$ from Π^* with $||\mathbf{d}_1^*||_2 \le \sigma\sqrt{m}$, $||\mathbf{d}_2^*||_2 \le \sigma\sqrt{m}$, $||\mathbf{r}^*||_2 \le \sigma\sqrt{m}$ satisfying Eq 19.

The simulator $\mathcal{B}$ declares failure in the following situations:

(i) No membership certificate is issued by $\mathcal{B}$ for the identifier $\mathsf{id}^* \in \{0,1\}^\mu$ on the $\mathsf{Q}_{\mathsf{a-join}}$ query by $\mathcal{A}$ *i.e.*, id^* does not belong to any user in U^a.

(ii) The identifier $\mathsf{id}^* \in \{0,1\}^\mu$ belongs to some user i in U^a, but this user is not the one introduced at the i^*-th $\mathsf{Q}_{\mathsf{a-join}}$ query *i.e.*, $i^* \neq \widehat{i}$ and $\mathsf{id}^* \neq \widehat{\mathsf{id}}$.

(iii) The knowledge extractor reveals vectors $\mathsf{bin}(\mathbf{v}^*) \in \{0,1\}^{2m}$ and $\mathbf{s}^* \in \mathbb{Z}^{2m}$ with $\mathbf{r}^* = \mathbf{r}_{i^*}$ where $\mathsf{bin}(\mathbf{v}_{i^*})$ and $\mathbf{s}_{i^*}$ are the vectors involved in the i^*-th $\mathsf{Q}_{\mathsf{a-join}}$ query *i.e.*, $\mathsf{id}^* = \widehat{\mathsf{id}}$ and $\mathbf{r}^* = \mathbf{r}_{i^*}$.

We denote the event by fail when any one of the above circumstances occur. With a prediction that fail does not occur, in which case $\mathsf{id}^* = \widehat{\mathsf{id}} = \mathsf{id}_{i^*}$ belongs to user $i^* \in U^a$, but $\mathbf{r}^* \neq \mathbf{r}_{i^*}$ $\mathcal{B}$ can solve the given SIS instance as follows:

The simulator $\mathcal{B}$ computes the vector

$$\mathbf{h} = \mathbf{S}(\mathbf{d}_1^* - \mathbf{d}_{i^*,1}) + (\mathbf{S}_0 + \sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\mathbf{S}_k)(\mathbf{d}_2^* - \mathbf{d}_{i^*,2}) + (\mathbf{r}_{i^*} - \mathbf{r}^*)$$

using $\widehat{\mathsf{id}}$ pre selected by $\mathcal{B}$ before the Setup and $\mathbf{S}, \{\mathbf{S}_k\}_{k=0}^{\mu}, \mathbf{d}_{i^*,1}, \mathbf{d}_{i^*,2}, \mathbf{r}_{i^*}$ set by $\mathcal{B}$ itself during the Setup phase and $\mathbf{d}_1^*$, $\mathbf{d}_2^*$, $\mathbf{r}^*$ are extracted by $\mathcal{B}$ using the knowledge extractor from the witness $\mathbf{x}$ of the proof of knowledge Π^* of $\mathbf{Px} = \mathbf{v}$ associated in the forgery σ^* on message

M^* at time t^*. Also $||\mathbf{d}_1^*||_2 \leq \sigma\sqrt{m}$, $||\mathbf{d}_2^*||_2 \leq \sigma\sqrt{m}$ and $||\mathbf{r}^*||_2 \leq \sigma\sqrt{m}$. Then

$$\begin{aligned}
\bar{\mathbf{A}}_1\mathbf{h} &= \bar{\mathbf{A}}_1\mathbf{S}(\mathbf{d}_1^* - \mathbf{d}_{i^*,1}) + \bar{\mathbf{A}}_1(\mathbf{S}_0 + \sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\mathbf{S}_k)(\mathbf{d}_2^* - \mathbf{d}_{i^*,2}) + \bar{\mathbf{A}}_1(\mathbf{r}_{i^*} - \mathbf{r}^*) \\
&= \mathbf{A}(\mathbf{d}_1^* - \mathbf{d}_{i^*,1}) + \left(\mathbf{A}_0 + \sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\mathbf{A}_k\right)(\mathbf{d}_2^* - \mathbf{d}_{i^*,2}) + \bar{\mathbf{A}}_1(\mathbf{r}_{i^*} - \mathbf{r}^*) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{(from Eq 20 and } \mathsf{id}^* = \widehat{\mathsf{id}} = \mathsf{id}_{i^*}) \\
&= \left[\mathbf{A}|\left(\mathbf{A}_0 + \sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\mathbf{A}_k\right)\right][\mathbf{d}_1^*, \mathbf{d}_2^*]^t - \left[\mathbf{A}|\left(\mathbf{A}_0 + \sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\mathbf{A}_k\right)\right][\mathbf{d}_{i^*,1}, \mathbf{d}_{i^*,2}]^t \\
&\qquad + \bar{\mathbf{A}}_1(\mathbf{r}_{i^*} - \mathbf{r}^*) \\
&= \mathbf{A}_{\mathsf{id}^*}\begin{bmatrix}\mathbf{d}_1^* \\ \hline \mathbf{d}_2^*\end{bmatrix} - \mathbf{A}_{\mathsf{id}_{i^*}}\begin{bmatrix}\mathbf{d}_{i^*,1} \\ \hline \mathbf{d}_{i^*,2}\end{bmatrix} + \bar{\mathbf{A}}_1(\mathbf{r}_{i^*} - \mathbf{r}^*) \\
&= (\mathbf{u} + \mathbf{D}\mathbf{r}^*) - (\mathbf{u} + \mathbf{D}\mathbf{r}_{i^*}) + \bar{\mathbf{A}}_1(\mathbf{r}_{i^*} - \mathbf{r}^*) \quad \text{(from Eq 19)} \\
&= \mathbf{D}(\mathbf{r}^* - \mathbf{r}_{i^*}) + \bar{\mathbf{A}}_1(\mathbf{r}_{i^*} - \mathbf{r}^*) \\
&= \bar{\mathbf{A}}_1(\mathbf{r}^* - \mathbf{r}_{i^*}) + \bar{\mathbf{A}}_1(\mathbf{r}_{i^*} - \mathbf{r}^*) \\
&= \mathbf{0} \bmod q
\end{aligned}$$

Thus $\mathbf{h}$ is short vector satisfying $\bar{\mathbf{A}}_1 \cdot \mathbf{h} = 0 \bmod q$ with

$$\begin{aligned}
||\mathbf{h}||_2 &\leq ||\mathbf{S}(\mathbf{d}_1^* - \mathbf{d}_{i^*,1}) + (\mathbf{S}_0 + \sum_{k=1}^{\mu} \widehat{\mathsf{id}}[k]\mathbf{S}_k)(\mathbf{d}_2^* - \mathbf{d}_{i^*,2}) + (\mathbf{r}_{i^*} - \mathbf{r}^*)|| \\
&\leq 2m\sigma^2 + m^{3/2}\sigma(\mu+1)2\sigma\sqrt{m} + 2\sigma\sqrt{m} \\
&\leq 2m\sigma^2(1 + m(\mu+2)) + 2\sigma\sqrt{m}.
\end{aligned}$$

Note that $\mathbf{r}^* \neq \mathbf{r}_{i^*}$, ensures that $\mathbf{h} \neq \mathbf{0}$ with probability. It gives, $(\mathbf{h}^t|0^m)^t$ is a short vector in $\Lambda_q^{\perp}(\bar{\mathbf{A}})$ and hence a solution of the given SIS instance.

Subcase II$(\mathbf{r}^* = \mathbf{r}_{i^*}$ but $(\mathsf{bin}(\mathbf{v}^*), \mathbf{s}^*) \neq (\mathsf{bin}(\mathbf{v}_{i^*}), \mathbf{s}_{i^*}))$:

Setup: The simulator $\mathcal{B}$ generates the group public key

$$\mathcal{Y} = (\mathbf{M}_0, \mathbf{M}_1, \mathbf{A}, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{B}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}, \mathsf{OTS}, \mathsf{DSig}, H, H_0, \beta, \gamma, \sigma)$$

faithfully as in the original protocol except the matrices $\mathbf{D}_0, \mathbf{D}_1 \hookleftarrow U(\mathbb{Z}_q^{2n\times 2m})$ which are set by $\mathcal{B}$ as follows: It samples $\bar{\mathbf{A}}' \hookleftarrow U(\mathbb{Z}_q^{n\times 2m})$, randomly selects $\mathbf{Q} \hookleftarrow \mathbb{Z}^{2m\times 2m}$ whose columns are sampled from $D_{\mathbb{Z}^{2m},\sigma}$ and sets $\mathbf{D}_0 = \begin{bmatrix}\bar{\mathbf{A}} \\ \hline \bar{\mathbf{A}}'\end{bmatrix} \in \mathbb{Z}_q^{2n\times 2m}$ and $\mathbf{D}_1 = \mathbf{D}_0.\mathbf{Q} \bmod q$. The distribution of each of the matrices $\mathbf{D}_0, \mathbf{D}_1$ is statistically close to $U(\mathbb{Z}_q^{2n\times 2m})$. The simulator initializes the public state $\mathsf{St} = (\mathsf{St}_{\mathsf{users}}, \mathsf{St}_{\mathsf{trans}}) = (\emptyset, \epsilon)$.

Query Phase: The respond to $\mathcal{A}$'s queries for $\mathsf{Q}_{\mathsf{pub}}$, $\mathsf{Q}_{\mathsf{read}}$, $\mathsf{Q}_{\mathsf{keyOA}}$, and hash queries are simulated by $\mathcal{B}$ as in Case I whereas $\mathsf{Q}_{\mathsf{a-join}}$ queries are answered as in the original protocol.

Forgery: Let (M^*, σ^*, t^*) be the forgery of $\mathcal{A}$ for a user $\mathcal{U} \notin U^a$ where

$$\sigma^* = (\mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \Pi^*, \mathsf{osig}^*, \mathbf{C}_R^*, [t_1^*, t_2^*])$$

such that $\mathsf{FSGS.Verify}(\sigma^*, t^*, M^*, \mathcal{Y}) = 1$ with $t^* \in [t_1, t_2]$ where VK^*, $\mathbf{c}_{\mathbf{v}^*}$, Π^*, osig^*, $[t_1^*, t_2^*]$ are as described in Case I.
Using improved forking lemma explained in Theorem 2.3 of Section 2, the simulator $\mathcal{B}$ has access to a knowledge extractor as described in case I. From the knowledge extractor $\mathcal{B}$ extracts $\mathsf{bin}(\mathbf{v}^*) \in \{0,1\}^{2m}, \mathbf{s}^* \in \mathbb{Z}^{2m}, \mathbf{r}^* \in \{0,1\}^m$.
The simulator $\mathcal{B}$ declares failure if $\mathbf{r}^* \neq \mathbf{r}_{i^*}$.
We denote the event by fail when the above situation occurs. When fail does not occur, $\mathcal{B}$ solves the given SIS instance by computing the vector

$$\mathbf{h} = \mathsf{bin}(\mathbf{v}^*) - \mathsf{bin}(\mathbf{v}_{i^*}) + \mathbf{Q}.(\mathbf{s}^* - \mathbf{s}_{i^*}) \in \mathbb{Z}^{2m}.$$

Observe that $\mathbf{h} \in \Lambda_q^{\perp}(\mathbf{D}_0)$ is a short vector and

$$\begin{aligned}\mathbf{D}_0.\mathbf{h} &= (\mathbf{D}_0\mathsf{bin}(\mathbf{v}^*) + \mathbf{D}_0\mathbf{Q}\mathbf{s}^*) - (\mathbf{D}_0\mathsf{bin}(\mathbf{v}_{i^*}) + \mathbf{D}_0\mathbf{Q}\mathbf{s}_{i^*})\\ &= (\mathbf{D}_0\mathsf{bin}(\mathbf{v}^*) + \mathbf{D}_1\mathbf{s}^*) - (\mathbf{D}_0\mathsf{bin}(\mathbf{v}_{i^*}) + \mathbf{D}_1\mathbf{s}_{i^*}) \quad \text{(as by } \mathsf{Setup}\ \mathbf{D}_1 = \mathbf{D}_0\mathbf{Q})\\ &= \mathbf{0} \bmod q \quad (\text{ as } \mathbf{r}^* = \mathbf{r}_{i^*}),\end{aligned}$$

which in turn implies that

$$\begin{aligned}&\left[\frac{\bar{\mathbf{A}}}{\bar{\mathbf{A}}'}\right]\mathbf{h} = 0 \bmod q\\ &\Rightarrow \left[\frac{\bar{\mathbf{A}}\mathbf{h}}{\bar{\mathbf{A}}'\mathbf{h}}\right] = 0 \bmod q\\ &\Rightarrow \bar{\mathbf{A}}\mathbf{h} = 0 \bmod q\\ &\Rightarrow \mathbf{h} \in \Lambda_q^{\perp}(\bar{\mathbf{A}})\end{aligned}$$

Further $\mathbf{h} \in \mathbb{Z}^{2m}$ is nonzero with high probability since $\mathsf{bin}(\mathbf{v}^*) \neq \mathsf{bin}(\mathbf{v}_{i^*})$ and

$$\begin{aligned}||\mathbf{h}||_2 &\leq ||\mathsf{bin}(\mathbf{v}^*) - \mathsf{bin}(\mathbf{v}_{i^*}) + \mathbf{Q} \cdot (\mathbf{s}^* - \mathbf{s}_{i^*})||_2\\ &\leq (||\mathsf{bin}(\mathbf{v}^*)||_2 + ||\mathsf{bin}(\mathbf{v}_{i^*}||_2) + ||\mathbf{Q}||_2(||\mathbf{s}^*||_2 + ||\mathbf{s}_{i^*}||_2)\\ &\leq 2^{\frac{3}{2}} m^{\frac{1}{2}}\sigma + 2^{\frac{7}{2}} m^2 \sigma^2\end{aligned}$$

Given the type of attack is completely independent of $i^* \hookleftarrow U([1, \delta])$, pre selected by $\mathcal{B}$ from adversary's view, the probability of correct attack is 1/3. In Case I, the correctly guess of $\widehat{\mathsf{id}} \in \{0,1\}^\mu$ has probability $1/(N-\delta) \geq 1/N$ and for Case II, the probability of correctly guess $i^* \in \{1, 2, \ldots, \delta\}$ is $1/\delta$. Hence $\mathsf{Pr}[\neg\text{fail}] \geq 1/\{3T\mathsf{Max}(N, \delta)\} \geq 1/(3TN)$.

Theorem 6.2. *The group signature* FSGS *scheme described in Section 4 is secure against framing attack as per the security framework given in Algorithm 3 of Section 3.2 under the* SIS *assumption in the random oracle model.*
Proof: Let $\mathcal{A}$ be any PPT adversary to our group signature scheme FSGS with non-negligible advantage ϵ. We show that there exists a simulator $\mathcal{B}$ that solves the SIS problem using the non-negligible advantage of $\mathcal{A}$ i.e., given $\bar{\mathbf{A}} = [\bar{\mathbf{A}}_1|\bar{\mathbf{A}}_2] \in \mathbb{Z}_q^{4n\times 4m}$, $\bar{\mathbf{A}}_1$, $\bar{\mathbf{A}}_2 \in \mathbb{Z}_q^{4n\times 2m}$, $m = 2n\lceil\log q\rceil$, a real number β, the simulator $\mathcal{B}$ finds a vector $\mathbf{z} \in \Lambda_q^{\perp}(\bar{A})$ such that $||\mathbf{z}|| \leq \beta$ using $\mathcal{A}$ as a subroutine.
Setup: The simulator $\mathcal{B}$ acts as the KGC and honest users while the adversary $\mathcal{A}$ acts as the group manager. First, $\mathcal{B}$ chooses $i^* \hookleftarrow U(\{1, 2, ..., q_b\})$ with a guess that the forged signature produced by $\mathcal{A}$ reveals $\mathsf{bin}(\mathbf{v}^*)$ coincides with $\mathsf{bin}(\mathbf{v}_{i^*})$ of i^*-th user in U^b and also chooses $\widehat{t} \hookleftarrow U(\{1, 2, ..., T\})$ where q_b is the cardinality of the set U^b of honest users with $\mathcal{A}$ as the dishonest group manager and $T = 2^l$ is allowable time periods with $T \leq q$ and $N = 2^{\mu}$ is maximum number of group members with $\mu \geq \log l$.
Also $\mathcal{B}$ has access to a knowledge extractor for every cheating prover for ZKAoK of the relation $\mathbf{Px} = \mathbf{v} \bmod q$ associated in $\mathcal{A}$'s forgery (M^*, σ^*, t^*) where σ^* is a forged signature σ^* on a message M^* at time t^*.
The simulator $\mathcal{B}$ faithfully generates the group public key

$$\mathcal{Y} = (\mathbf{M}_0, \mathbf{M}_1\mathbf{A}, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{B}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}, \mathsf{OTS}, \mathsf{DSig}, H, H_0, \beta, \gamma, \sigma)$$

as in the original protocol except the matrix $\mathbf{F} \hookleftarrow U(\mathbb{Z}_q^{4n\times 2m})$ which is set by $\mathcal{B}$ as $\mathbf{F} = \bar{\mathbf{A}} \in \mathbb{Z}_q^{4n\times 2m}$. It also initializes the public state $\mathsf{St} = (\mathsf{St}_{\mathsf{users}}, \mathsf{St}_{\mathsf{trans}}) = (\emptyset, \epsilon)$ and sets $\mathsf{S}_{\mathsf{OA}} = \mathsf{T}_{\mathbf{B}}$, $\mathsf{S}_{\mathsf{GM}} = \mathsf{T}_{\mathbf{A}}$ where $\mathsf{T}_{\mathbf{A}}, \mathsf{T}_{\mathbf{B}}$ are the secret keys of the group manager and the opening authority respectively.

•***Query Phase***: The adversary $\mathcal{A}$ queries the oracles $\mathsf{Q}_{\mathsf{pub}}$, $\mathsf{Q}_{\mathsf{KeyGM}}$, $\mathsf{Q}_{\mathsf{b-join}}$, $\mathsf{Q}_{\mathsf{sig}}$, $\mathsf{Q}_{\mathsf{corrupt}}$, $\mathsf{Q}_{\mathsf{write}}$, $\mathsf{Q}_{\mathsf{read}}$ and $\mathsf{Q}_{\mathsf{KeyOA}}$ which are simulated by $\mathcal{B}$ as follows:

- $\mathsf{Q}_{\mathsf{pub}}$: In response to this query, $\mathcal{B}$ returns $\mathcal{Y}$ to $\mathcal{A}$.
- $\mathsf{Q}_{\mathsf{keyGM}}$ and $\mathsf{Q}_{\mathsf{keyOA}}$: If $\mathcal{A}$ chooses to corrupt the group manager and the opening authority, $\mathcal{B}$ passes S_{GM} and S_{OA} generated honestly in the Setup phase to $\mathcal{A}$.
- $\mathsf{Q}_{\mathsf{b-join}}$: To introduce a new member in $\mathcal{U}_i \in U^b$, the adversary $\mathcal{A}$ acting as the dishonest group manager $\mathcal{B}$ and handles the Turing machine J_{GM} of $\mathsf{FSGS.Join}$ while the simulator $\mathcal{B}$ faithfully runs the Turing machine $\mathsf{J}_{\mathsf{user}}$ of $\mathsf{FSGS.Join}$ on behalf of the user. If the protocol successfully terminates, then $\mathcal{B}$ increments U by 1, adds user index i to both U^b and $\mathsf{St}_{\mathsf{users}}$, and updates $\mathsf{St}_{\mathsf{trans}}$ as $\mathsf{St}_{\mathsf{trans}} = \mathsf{St}_{\mathsf{trans}}|\langle i, t_1, t_2, \mathsf{transcript}_i\rangle$. The simulator $\mathcal{B}$ finally stores the membership certificate $\mathsf{cert}_{i,t_1\to t_2}$ in a private part of State_I.
- $\mathsf{Q}_{\mathsf{sig}}$: On querying a tuple (M, i, t) consisting of a message M, an index i, and a time period t, the simulator $\mathcal{B}$ first checks if $\mathsf{State}_{\mathsf{pri}}$ contains $\mathsf{sec}_{i,t_1\to t_2}, \mathsf{cert}_{i,t_1\to t_2}$ for some $t_1, t_2 \in \{1, 2, ..., T\}$ with $t_1 \leq t \leq t_2$ and $\mathcal{U}_i \in U^b$. If so, $\mathcal{B}$ calls $\mathsf{FSGS.Update}(\mathcal{Y}, k, t_2, \mathsf{sec}_{i,k\to t_2}, \mathsf{cert}_{i,k\to t_2})$ for $k = t_1, t_1 + 1, ..., t$ to generate the pair $(\mathsf{cert}_{i,t\to t_2}, \mathsf{sec}_{i,t\to t_2})$, returns a signature $\sigma \longleftarrow \mathsf{FSGS.Sign}(\mathsf{sec}_{i,t\to t_2}, \mathsf{cert}_{i,t\to t_2}, \mathcal{Y}, M)$ to $\mathcal{A}$ on behalf of user

$\mathcal{U}_i$ for the period t and updates Sigs=$\mathsf{Sigs}|\langle i, t, M, \sigma\rangle$. Otherwise, the simulator $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

- $\mathsf{Q}_{\mathsf{corrupt}}$: On receiving query (i, t), where $i \in \mathsf{St}_{\mathsf{users}}$ and $t \in \{1, 2, ..., T\}$ from $\mathcal{A}$, the simulator $\mathcal{B}$ checks if $\mathcal{U}_i \in U^b$ and $\mathsf{St}_{\mathsf{trans}}$ has a record of the form $\langle i, t_1, t_2, \mathsf{transcript}_i\rangle$ for some $t_1, t_2 \in \{1, 2, ..., T\}$ with $t_1 \leq t \leq t_2$. If not, returns $\perp$. Else, $\mathcal{B}$ extracts $\mathsf{cert}_{i,t_1\to t_2}$, $\mathsf{sec}_{i,t_1\to t_2}$ from $\mathsf{State}_{\mathsf{pri}}$ and iteratively call algorithm $\mathsf{FSGS.Update}(\mathcal{Y}, k, t_2, \mathsf{sec}_{i,k\to t_2}, \mathsf{cert}_{i,k\to t_2})$ for $k = t_1, t_1 + 1, ..., t - 1$ to generate $\mathsf{cert}_{i,t\to t_2}$ and $\mathsf{sec}_{i,t\to t_2}$ for $t > t_1$. It provides all these information to the adversary and stores (i, t) in $\mathsf{St}_{\mathsf{corr}}$.
- $\mathsf{Q}_{\mathsf{read}}$ and $\mathsf{Q}_{\mathsf{write}}$: On access to these queries. $\mathcal{A}$ can read and write state St=$(\mathsf{St}_{\mathsf{users}}, \mathsf{St}_{\mathsf{trans}})$ without altering, removing or reusing the already existing certificates.

•***Forgery***: With access to all the above queries, $\mathcal{A}$ outputs the forgery (M^*, t^*, σ^*) for a user $\mathcal{U} \in U^b$ where σ^* is a forged signature on the message M^* at time t^*. Parsing σ^* as

$$\sigma^* = (\mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \Pi^*, \mathsf{osig}^*, \mathbf{C}_R^*, [t_1^*, t_2^*])$$

such that $\mathsf{FSGS.Verify}(\sigma^*, t^*, M^*, \mathcal{Y}) = 1$ where $\mathsf{VK}^*, \mathbf{c}_{\mathbf{v}^*}, \Pi^*, \mathsf{osig}^*, [t_1^*, t_2^*])$ is as explained in Case I of Theorem 6.1.
The simulator $\mathcal{B}$ declares failure if $\mathbf{v}^* \neq \mathbf{v}_{i^*}$. The opening of σ^* reveals $\mathsf{bin}(\mathbf{v}^*) = \mathsf{bin}(\mathbf{v}_{i^*}) \in \{0, 1\}^{2m}$ and as $\mathcal{B}$ has all the collection of short $\mathbf{z}_i \in \mathbb{Z}^{4m}$ for each user $\mathcal{U}_i \in U^b$ with $||\mathbf{z}_i|| \leq 2\sigma\sqrt{m}$, the simulator $\mathcal{B}$ can find a $\mathbf{z}_{i^*}$ such that $\mathbf{v}_{i^*} = \mathbf{F} \cdot \mathbf{z}_{i^*} \mod q$. Now, $\mathcal{B}$ finds another short vector $\widehat{\mathbf{z}}$ satisfying $\mathbf{v}_{i^*} = \mathbf{F} \cdot \widehat{\mathbf{z}} \mod q$ by invoking improved forking lemma explained in Theorem 2.3 of Section 2.3.
Hence $\mathcal{B}$ has $\mathbf{z}_{i^*}, \widehat{\mathbf{z}}$ such that $\mathbf{v}^* = \mathbf{v}_{i^*} = \mathbf{F} \cdot \mathbf{z}_{i^*} = \mathbf{F} \cdot \widehat{\mathbf{z}} \mod q$. Due to the statistical witness indistinguishability ZKAoK to generate signature, $\widehat{\mathbf{z}} \neq \mathbf{z}_{i^*}$ with overwhelming probability. Also the distribution of $\mathbf{z}_{i^*}$ is $D_{\Lambda_q^{\mathbf{v}_{i^*}}(\mathbf{F}),\sigma}$. Thus $\mathcal{B}$ has

$$\begin{aligned}\mathbf{F} \cdot \mathbf{z}_{i^*} = \mathbf{F} \cdot \widehat{\mathbf{z}} \mod q &\Rightarrow \mathbf{F}(\mathbf{z}_{i^*} - \widehat{\mathbf{z}}) = 0 \mod q \\ &\Rightarrow \bar{\mathbf{A}}(\mathbf{z}_{i^*} - \widehat{\mathbf{z}}) = 0 \mod q \quad (\text{by } \mathsf{Setup}) \\ &\Rightarrow \mathbf{z}_{i^*} - \widehat{\mathbf{z}} \in \Lambda_q^{\perp}(\bar{\mathbf{A}})\end{aligned}$$

with

$$||u|| \leq ||\mathbf{z}_{i^*}|| + ||\widehat{\mathbf{z}}|| \leq 2\sigma\sqrt{m} + 2\sigma\sqrt{m} = 4\sigma\sqrt{m}$$

is a short vector of $\Lambda_q^{\perp}(\bar{\mathbf{A}})$ i.e., a short solution to the given SIS instance.

Theorem 6.3. *The* FSGS *scheme described in Section 4 is fully anonymous as per the security framework given in Algorithm 4 of section 3.2 under the* LWE*,* SIS *assumption and the unforgeability of one time signature* OTS.

Proof: The proof of this theorem is structured as a sequence of seven computationally indistinguishable games. In the first game, each query is answered as in the real protocol. We then progressively change each game, show that each game is indistinguishable from the previous one, and finally prove that our FSGS construction is secure in the security model of Algorithm 4. Let $E_i = Adv^{anon}(\mathcal{A})$ in Game i for $i = 0, 1, \ldots, 6$. The game transition is described below.

• **Game** 0: Let $\mathcal{A}$ be the adversary and $\mathcal{B}$ be the simulator.

Setup: The challenger $\mathcal{B}$ runs

$$\mathsf{FSGS.Setup}(\lambda, T) \rightarrow (\mathcal{Y}, \mathsf{S_{GM}}, \mathsf{S_{OA}}, \mathsf{St})$$

where $\mathcal{Y} = (\mathbf{M}_0, \mathbf{M}_1, \mathbf{A}, \{\mathbf{A}_k\}_{k=0}^{\mu}, \mathbf{B}, \mathbf{D}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{F}, \mathbf{u}, \mathsf{OTS}, \mathsf{DSig}, H, H_0, \beta, \gamma, \sigma)$, $\mathsf{S_{GM}} = \mathsf{T}_\mathbf{A} \in \mathbb{Z}^{m \times m}$, $\mathsf{S_{OA}} = \mathsf{T}_\mathbf{B} \in \mathbb{Z}^{m \times m}$ and initializes the public state $\mathsf{St} = (\mathsf{St_{users}}, \ \mathsf{St_{trans}}) = (\emptyset, \epsilon)$.
Query Phase: The challenger $\mathcal{B}$ responds to $\mathsf{Q_{pub}}, \mathsf{Q_{keyGM}}, \mathsf{Q_{open}}, \mathsf{Q_{read}}, \mathsf{Q_{write}}$ and hash queries made by $\mathcal{A}$ as follows:

- $\mathsf{Q_{pub}}$: In response to this query, $\mathcal{B}$ returns $\mathcal{Y}$ to $\mathcal{A}$.
- $\mathsf{Q_{keyGM}}$: If $\mathcal{A}$ chooses to corrupt the group manager, $\mathcal{B}$ passes $\mathsf{S_{GM}}$ to $\mathcal{A}$.
- $\mathsf{Q_{open}}$: On access to this query, $\mathcal{A}$ asks $\mathcal{B}$ to open the message-signature pair (M, t, gsig) with the current state St where gsig is a group signature on a message M at time t. The challenger $\mathcal{B}$ uses the opening authority key $\mathsf{T}_\mathbf{A} \in \mathbb{Z}^{m \times m}$ to answer this query by running $\mathsf{FSGS.Open}(\mathsf{gsig}, t, M, \mathcal{Y}, \mathsf{S_{OA}}, \mathsf{St}) \longrightarrow (i \vee \perp)$
- $\mathsf{Q_{read}}$ and $\mathsf{Q_{write}}$: On access to these queries, $\mathcal{A}$ can respectively read and write state St without altering, removing or reusing the already existing certificates.
- Hash Queries: To respond hash queries, $\mathcal{B}$ maintains two lists for the hash functions H and H_0. These lists store records of the form $(\mathbf{x}, H(\mathbf{x}))$ and $(\mathbf{y}, H_0(\mathbf{y}))$ for some $\mathbf{x}, \mathbf{y} \in \{0,1\}^*$. Fresh hash value is generated on a string $\mathbf{x} \in \{0,1\}^*$ if it is not already been queried and stored in the corresponding hash list. Otherwise, responses are given utilizing the respective hash lists.

Play Phase: With access to all the above queries, $\mathcal{A}$ sends $(M^*, t^*, (\mathsf{sec}_{i_b, t^* \rightarrow t_b^*}, \mathsf{cert}_{i_b, t^* \rightarrow t_b^*}))$ with $b \in \{0,1\}$ to $\mathcal{B}$. The challenger $\mathcal{B}$ aborts if either $([\neg(\mathsf{cert}_{i_b, t^* \rightarrow t_b^*} \rightleftharpoons_{\mathcal{Y}} \mathsf{sec}_{i_b, t^* \rightarrow t_b^*})]$ or $[\mathsf{cert}_{i_0, t^* \rightarrow t_0^*} = \mathsf{cert}_{i_1, t^* \rightarrow t_1^*}]$.

Challenge Phase: In this phase, the challenger $\mathcal{B}$ flips a coin and selects a bit $d \in \{0,1\}$ to sign the message M^* at time t^*. The challenger $\mathcal{B}$ computes the challenge signature

$$\mathsf{gsig}^* \leftarrow \mathsf{FSGS.Sign}(\mathsf{sec}_{i_d, t^* \rightarrow t_d^*}, \mathsf{cert}^*_{i_d, t^* \rightarrow t_d^*}, \mathcal{Y}, M^*)$$

and sends it to $\mathcal{A}$ where $gsig^* = (\mathsf{VK}^*, \mathbf{c}_{\mathbf{v}_d^*}, \Pi^*, \mathsf{osig}^*, \mathbf{C}_R^*, [\hat{t}_1, \hat{t}_2])$ such that $\mathsf{FSGS.Verify}(\sigma^*, t^*, M^*, \mathcal{Y}) = 1$ with $t^* \in [\hat{t}_1, \hat{t}_2]$ where

VK^*	:	verification key generated by $\mathsf{OTS}\cdot\mathcal{G}$.
$\mathbf{c}_{\mathbf{v}^*}$	:	ciphertext generated by running the algorithm $\mathsf{GPVIBE}\cdot\mathsf{Enc}(\mathbf{B}, \mathsf{VK}^*, \mathbf{y}^*) \rightarrow$
	:	$\mathbf{c}_{\mathbf{v}^*} \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{2m}$ where $\mathbf{B}$ is extracted from $\mathcal{Y}$ and $\mathbf{y}^* = \mathsf{bin}(\mathbf{v}^*) \in \{0,1\}^{2m}$
	:	for some $\mathbf{v}^* \in \mathbb{Z}_q^{2m}$.
Π^*	:	zero knowledge proof for $\mathbf{Px} = \mathbf{v} \bmod q$ equivalent to the system of equation
	:	similar to Eq 12.
osig^*	:	one time signature generated by running $\mathsf{OTS}.\mathcal{S}(\mathsf{SK}^*, \mathbf{c}_{\mathbf{v}^*}, \Pi^*)$ where SK^* is the
	:	signing key corresponding to the verification key VK^* generated by $\mathsf{OTS}\cdot\mathcal{G}$.
$[\hat{t}_1, \hat{t}_2]$	:	a subset of time interval $[1, T]$.

Guess Phase: With access to the queries $\mathsf{Q}_{\mathsf{pub}}, \mathsf{Q}_{\mathsf{keyGM}}, \mathsf{Q}_{\mathsf{open}}^{\neg\{(\mathsf{M}^*,\sigma^*,\mathsf{t}^*)\}}, \mathsf{Q}_{\mathsf{read}}, \mathsf{Q}_{\mathsf{write}}$, the adversary $\mathcal{A}$ guesses a bit d' and wins the game if $d' = d$. The experiment returns 1 if $d' = d$, else returns 0.

• **Game** 1: It is similar to the above game with a slight modification. At the beginning of the game, the challenger $\mathcal{B}$ generates a OTS key pair $(\mathsf{VK}^*, \mathsf{SK}^*)$ before query phase starts. This is used by $\mathcal{B}$ in the challenge query. The challenger $\mathcal{B}$ aborts if following two cases occur:

- The adversary $\mathcal{A}$ asks $\mathsf{Q}_{\mathsf{open}}$ query to $\mathcal{B}$ for the signature $\mathsf{gsig} = (\mathsf{VK}, \mathbf{c}_{\mathbf{v}}, \Pi, \mathsf{osig}, \mathbf{C}_R, [t_1, t_2])$ with $\mathsf{VK} = \mathsf{VK}^*$.
- The adversary $\mathcal{A}$ produces a signature $\mathsf{gsig} = (\mathsf{VK}, \mathbf{c}_{\mathbf{v}}, \Pi, \mathsf{osig}, \mathbf{C}_R, [t_1, t_2])$ with $\mathsf{VK} = \mathsf{VK}^*$ after knowing the challenge signature gsig^*.

Since both the cases have negligible probability as they contradict the strong unforgeability of one time signature OTS, Game 0 and Game 1 are identical with high probability from $\mathcal{A}$'s point of view. That is, $|E_0 - E_1| = \epsilon_1$ where $\epsilon_1 > 0$ is negligible.

• **Game** 2: This game explains the simulation of the hash queries. Firstly, choose a random matrix $\mathbf{G}_0^* \in \mathbb{Z}_q^{n\times 2m}$ uniformly and define $H_0(\mathsf{VK}^*) = \mathbf{G}_0^*$. Observe that the distribution of $\mathbf{G}_0^*$ is still statistically close to that in the real game. To answer other queries for $\mathsf{VK} \neq \mathsf{VK}^*$, the challenger $\mathcal{B}$ selects a small norm matrix $\mathbf{K}_{\mathsf{VK}} \hookleftarrow D_{\mathbb{Z}^m,\sigma}^{2m}$ and defines $H_0(\mathsf{VK}) = \mathbf{B}\cdot\mathbf{K}_{\mathsf{VK}}$. The matrix $\mathbf{K}_{\mathsf{VK}}$ is stored for further usage. If any query is repeated for VK, the stored value will be produced. Note that in the real protocol, $\mathbf{K}_{\mathsf{VK}}$ is generated using the trapdoor $\mathsf{T}_{\mathbf{B}}$. As SIS is hard, finding $\mathbf{K}_{\mathsf{VK}}$ is difficult without $\mathsf{T}_{\mathbf{B}}$ and hence difficult to decide whether $\mathbf{K}_{\mathsf{VK}}$ is generated using $\mathsf{T}_{\mathbf{B}}$ or uniformly chosen, for all fixed choices of $\mathbf{B}$. Thus $H_0(\mathsf{VK})$ is statistically close to uniform. Also distinguishing between the pair $(\mathbf{B}, \mathbf{B}\cdot\mathbf{K}_{\mathsf{VK}})$ generated using $\mathsf{T}_{\mathbf{B}}$, and the uniformly generated pair $(\mathbf{B}, H_0(\mathsf{VK}))$ is a decisional LWE problem. Hence Game 1 and Game 2 are statistically indistinguishable. In other words, $|E_1 - E_2| = \epsilon_2$ where $\epsilon_2 > 0$ is negligible.

• **Game** 3: This game includes the modification in the opening algorithm. At the outset of the game, $\mathcal{B}$ samples a uniformly random matrix $\mathbf{B}^* \in \mathbb{Z}_q^{n\times m}$ to use in place of $\mathbf{B}$ to response H_0-queries. Hence to answer $\mathsf{Q}_{\mathsf{open}}$ queries, $\mathcal{B}$ does not use $\mathsf{T}_{\mathbf{B}}$ but recalls the stored

$\mathbf{K}_{\mathsf{VK}}$ matrices used for H_0-queries. This experiment is indistinguishable from the real one, under the LWE assumption. Thus Game 2 and Game 3 are statistically indistinguishable and $|E_2 - E_3| = \epsilon_3$ where $\epsilon_3 > 0$ is negligible.

• **Game** 4: Statistical zero knowledge states that the simulator can generate the proof that is statistically close to the real distribution [21]. Thus, $\mathcal{B}$ simulates the zero knowledge argument of knowledge $\Pi = (\{\mathsf{COM}_k\}_{k=1}^{s}, \mathsf{Ch}, \{\mathsf{RSP}_k\}_{k=1}^{s})$ where $\mathsf{Ch} = H(M, \mathsf{VK}, \mathbf{c}_{\mathbf{v}}, \{\mathsf{COM}_k\}_{k=1}^{s}) \in \{1, 2, 3\}^s$ instead of using the real witnesses. It is made possible by programming the hash queries according to the underlying interactive protocol by running the simulator for each $k \in \{1, 2, \ldots, s\}$. The interactive proof Π is statistically zero knowledge as stated in Theorem 1.2 and thus Game 3 and Game 4 are statistically indistinguishable. Consequently, $|E_3 - E_4| = \epsilon_4$ where $\epsilon_4 > 0$ is negligible.

• **Game** 5: Here the modification is made on the ciphertext $\mathbf{c}_{\mathbf{v}_{i_d}^*}$ of the challenge phase. The challenger $\mathcal{B}$ does not use the real $\mathsf{GPV\text{-}IBE}$ for encryption of $\mathsf{bin}(\mathbf{v}_{i_d}^*)$ but returns truly random ciphertext by setting

$$\mathbf{c}_{\mathbf{v}_{i_d}^*} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 + \mathsf{bin}(\mathbf{v}_{i_d}^*)\lfloor q/2 \rfloor \end{bmatrix}$$

where $\mathsf{bin}(\mathbf{v}_{i_d}^*) = \mathbf{F} \cdot \mathbf{z}_{i_d}^*$, and $\mathbf{z}_1 \hookleftarrow U(\mathbb{Z}_q^m)$, $\mathbf{z}_2 \hookleftarrow U(\mathbb{Z}_q^{2m})$ are randomly chosen. Note that in Game 0 to Game 5, $\mathbf{c}_{\mathbf{v}_{i_d}^*}$ is set as

$$\mathbf{c}_{\mathbf{v}_{i_d}^*} = \begin{bmatrix} \mathbf{B}^t \mathbf{e}_0 + \mathbf{e}_1 \\ \mathbf{G}_0^{*^t} \mathbf{e}_0 + \mathbf{e}_2 + \mathsf{bin}(\mathbf{v}_{i_d}^*)\lfloor q/2 \rfloor \end{bmatrix}$$

where $\mathbf{e}_0 \hookleftarrow \chi^n$, $\mathbf{x}_1 \hookleftarrow \chi^m$, $\mathbf{x}_2 \hookleftarrow \chi^{2m}$ and χ is a γ bounded distribution. If $\mathcal{A}$ can distinguish between $\mathbf{B}^t \mathbf{e}_0 + \mathbf{x}_1$ from $\mathbf{z}_1$ and $\mathbf{G}_0^{*^t} \mathbf{e}_0 + \mathbf{x}_2$ from $\mathbf{z}_2$, which would break the decisional LWE assumption. Thus Game 4 and Game 5 are computationally indistinguishable under the security of $\mathsf{GPV\text{-}IBE}$. Hence, $|E_4 - E_5| = \epsilon_5$ where $\epsilon_5 > 0$ is negligible.

• **Game** 6: This is the final game where $\mathcal{B}$ selects $\mathbf{z}_1' \hookleftarrow U(\mathbb{Z}_q^m), \mathbf{z}_2' \hookleftarrow U(\mathbb{Z}_q^{2m})$ uniformly and sets

$$\mathbf{c}_{\mathbf{v}_{i_d}}^* = \begin{bmatrix} \mathbf{z}_1' \\ \mathbf{z}_2' \end{bmatrix}$$

instead of using $\mathsf{bin}(\mathbf{v}_{i_d}^*)$. Observe that $\mathbf{c}_{\mathbf{v}_{i_d}}^*$ follows the same distribution as in Game 5. Also $\mathbf{c}_{\mathbf{v}_{i_d}^*}$ does not depend on $d \in \{0, 1\}$, giving zero advantage to the adversary $\mathcal{A}$. Thus, $|E_5 - E_6| = |E_5| = \epsilon_6$ where $\epsilon_6 > 0$ is negligible.
Thus the advantage of the adversary is

$|Adv(\mathcal{A})| \leq |E_0 - E_1| + |E_1 - E_2| + |E_2 - E_3| + |E_3 - E_4| + |E_4 - E_5| + |E_5 - E_6| = \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6 < \epsilon$. Hence the result.

7 Conclusion

In this work, we have proposed the first forward secure dynamic group signature scheme whose security relies on hard problems of lattices. The scheme achieves the strongest notion of security currently available in the literature. Our scheme is the first quantum resistant group signature scheme achieving forward secrecy in dynamic setting.

References

[1] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (h) ibe in the standard model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 553–572. Springer (2010)

[2] Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In: Annual Cryptology Conference. pp. 98–115. Springer (2010)

[3] 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)

[4] Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Theory of Computing Systems 48(3), 535–553 (2011)

[5] Anderson, R.: Two remarks on public key cryptology. Unpublished. Available from http://www. cl. cam. ac. uk/users/rja14 (1997)

[6] Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. Mathematische Annalen 296(1), 625–635 (1993)

[7] Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Eurocrypt. vol. 2656, pp. 614–629. Springer (2003)

[8] Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Annual International Cryptology Conference. pp. 431–448. Springer (1999)

[9] Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Cryptographers Track at the RSA Conference. pp. 136–153. Springer (2005)

[10] Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of fully dynamic group signatures. In: International Conference on Applied Cryptography and Network Security. pp. 117–136. Springer (2016)

[11] Boyen, X.: Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In: International Workshop on Public Key Cryptography. pp. 499–517. Springer (2010)

[12] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pp. 575–584. ACM (2013)

[13] Brickell, E., Pointcheval, D., Vaudenay, S., Yung, M.: Design validations for discrete logarithm based signature schemes. In: International Workshop on Public Key Cryptography. pp. 276–292. Springer (2000)

[14] Camenisch, J., Neven, G., Rückert, M.: Fully anonymous attribute tokens from lattices. In: International Conference on Security and Cryptography for Networks. pp. 57–75. Springer (2012)

[15] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 523–552. Springer (2010)

[16] Chaum, D., Van Heyst, E.: Group signatures. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 257–265. Springer (1991)

[17] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the Theory and Application of Cryptographic Techniques. pp. 186–194. Springer (1986)

[18] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206. ACM (2008)

[19] Goldreich, O., Sahai, A., Vadhan, S.: Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 399–408. ACM (1998)

[20] Gordon, S.D., Katz, J., Vaikuntanathan, V.: A group signature scheme from lattice assumptions. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 395–412. Springer (2010)

[21] Hohenberger, U.: Honest verifier zk and fiat-shamir (lecture 1) (2007), `https://www.cs.jhu.edu/~susan/600.641/scribes/lecture11.pdf`

[22] Kiayias, A., Yung, M.: Secure scalable group signature with dynamic joins and separable authorities. International Journal of Security and Networks 1(1-2), 24–45 (2006)

[23] Laguillaumie, F., Langlois, A., Libert, B., Stehlé, D.: Lattice-based group signatures with logarithmic signature size. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 41–61. Springer (2013)

[24] Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-based group signature scheme with verifier-local revocation. In: International Workshop on Public Key Cryptography. pp. 345–361. Springer (2014)

[25] Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In: Advances

in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II 22. pp. 373–403. Springer (2016)

[26] Libert, B., Yung, M.: Fully forward-secure group signatures. In: Cryptography and Security: From Theory to Applications, pp. 156–184. Springer (2012)

[27] Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the isis problem, and applications. In: Public-Key Cryptography–PKC 2013, pp. 107–124. Springer (2013)

[28] Ling, S., Nguyen, K., Wang, H.: Group signatures from lattices: simpler, tighter, shorter, ring-based. In: IACR International Workshop on Public Key Cryptography. pp. 427–449. Springer (2015)

[29] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM (JACM) 60(6), 43 (2013)

[30] Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In: Annual Cryptology Conference. pp. 465–484. Springer (2011)

[31] Nguyen, P.Q., Zhang, J., Zhang, Z.: Simpler efficient group signatures from lattices. In: IACR International Workshop on Public Key Cryptography. pp. 401–426. Springer (2015)

[32] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) 56(6), 34 (2009)

[33] Song, D.X.: Practical forward secure group signature schemes. In: Proceedings of the 8th ACM conference on Computer and Communications Security. pp. 225–234. ACM (2001)

[34] Stern, J.: A new paradigm for public key identification. IEEE Transactions on Information Theory 42(6), 1757–1768 (1996)