GROUP SIGNATURE FROM LATTICES PRESERVING FORWARD SECURITY IN DYNAMIC SETTING

MEENA KANSAL∗, RATNA DUTTA AND SOURAV MUKHOPADHYAY

Department of Mathematics
Indian Institute of Technology Kharagpur
Kharagpur-721302, India

(Communicated by Subhamoy Maitra)

Abstract. We propose the first lattice-based dynamic group signature scheme achieving forward security. Our scheme is proven to be secure against framing attack, misidentification attack and preserves anonymity under the learning with errors (LWE) and short integer solution (SIS) assumptions in the random oracle model. More interestingly, our setting allows the group manager to generate distinct certificates to distinct users which can be updated by the users themselves without any interaction with the group manager. Furthermore, our scheme is dynamic where signing key of a user is not fixed during the setup and is issued only at the joining time of the user.

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). 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. 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 signature has received considerable attention in the recent research community as it has tremendous appeal in various applications of group signature schemes. In contrast to static model, member identities in dynamic group setting are not fixed at the initial setup phase.

In this paper we address the problem of designing efficient group signature scheme in dynamic setting from lattice featuring forward security. We improvise the dynamic group signature scheme of Libert et al. [11] into a forward secure dynamic group signature scheme. We build our group signature adapting the techniques from [12]. We emphasize that the major advantage in our construction is by updating the procedure which can be performed entirely by the users without interacting
with the group manager, thereby, minimizing the interaction between users and the group manager. Similar to [11], we employ Gentry-Peikert-Vaikuntanathan identity based encryption (GPV-IBE) [7] and zero knowledge argument of knowledge [11] to generate signature. We briefly summarize the comparison of our scheme with the existing lattice based group signature schemes [5], [8], [10], [11], [14], [15] in Table 1, where $N$ denotes the total number of allowable group members and $n$ is the security parameter.

### Table 1. Comparative summary of lattice based group signature schemes

| Scheme | Forward secure | Dynamic size | Signature size | Public key size | Certificate size | Signer’s SK size |
|--------|----------------|--------------|----------------|----------------|-----------------|-----------------|
| [8]    | No             | No           | $N \cdot \mathcal{O}(n^2)$ | $N \cdot \mathcal{O}(n^2)$ | -               | $\mathcal{O}(n^3)$ |
| [5]    | No             | No           | $N \cdot \mathcal{O}(n^2)$ | $N \cdot \mathcal{O}(n^2)$ | -               | $\mathcal{O}(n^3)$ |
| [10]   | No             | No           | $\log N \cdot \mathcal{O}(n)$ | $\log N \cdot \mathcal{O}(n^2)$ | -               | $\mathcal{O}(n^3)$ |
| [14]   | No             | No           | $\log N \cdot \mathcal{O}(n)$ | $\log N \cdot \mathcal{O}(n^2)$ | -               | $\mathcal{O}(n)$ |
| [15]   | No             | Yes          | $\log N \cdot \mathcal{O}(n)$ | $\log N \cdot \mathcal{O}(n^2)$ | -               | $\mathcal{O}(n^3)$ |
| Ours   | Yes            | Yes          | $\log N \cdot \mathcal{O}(n)$ | $\log N \cdot \mathcal{O}(n^2)$ | -               | $\mathcal{O}(n^3)$ |

2. **Background and assumptions**

**Definition 2.1.** (Lattice). Let $B = \{b_i\}_{i \leq n}$ be a linearly independent set of vectors of $\mathbb{R}^n$. A lattice generated by $B$ is defined as $\Lambda(B) = \{ \sum_{b_i \in B} c_i b_i : c_i \in \mathbb{Z} \}$, the set of all integer linear combinations of $b_i \in B$. The set $B$ is said to be a basis of the lattice $\Lambda$. For $q \in \mathbb{N}$, matrix $A \in \mathbb{Z}_q^{n \times m}$ and vector $u \in \mathbb{Z}_q^n$, we define the following $q$-ary lattices generated by $A$: $\Lambda_q^1(A) = \{ x \in \mathbb{Z}^m : Ax = 0 \bmod q \}$ and $\Lambda_q^n(A) = \{ x \in \mathbb{Z}^m : Ax = u \bmod q \}$ where $m, n$ are integers with $m \geq n \geq 1$.

**Definition 2.2.** (Gaussian distribution over a lattice). For a lattice $\Lambda$ and a real number $\sigma > 0$, discrete Gaussian distribution over $\Lambda$ centered at $0$, denoted by $D_{\Lambda, \sigma}$, is defined as: $\forall y \in \Lambda, D_{\Lambda, \sigma}[y] \sim \text{exp}(-\pi ||y||^2/\sigma^2)$.

2.1. **Hardness assumptions.**

**Definition 2.3.** (Inhomogeneous short integer solution I(SIS) [2]). Given an integer $q$, a matrix $A \in \mathbb{Z}_q^{n \times m}$, a vector $u \in \mathbb{Z}_q^n$ and a real number $\beta$, the I(SIS) problem is to find an integer vector $e \in \mathbb{Z}^m$ such that $Ae = u \bmod q$ with $||e|| \leq \beta$. If $u = 0 \in \mathbb{Z}_q^n$, then it is known as short integer solution (SIS) problem.

**Definition 2.4.** (Learning with errors (LWE) [16]). Let $\chi$ be a distribution on $\mathbb{Z}$, $n \geq 1$ be any integer and $p \geq 2$ be any prime. For any $s \in \mathbb{Z}_p^n$, given arbitrarily many samples of the form $(a, (a, s) + e)$ with $a \in U(\mathbb{Z}_p^n)$ and $e$ is sampled from $\gamma$, the search LWE problem is to find $s$ and the decisional LWE problem is to distinguish the distribution of $(a, (a, s) + e)$ from the uniform distribution $U(\mathbb{Z}_p^n \times \mathbb{Z}_p)$.

2.2. **Node select algorithm [12].** Nodes$(t_1 + 1, t_2, G) \rightarrow (\text{SubsetHN})$. Following Libert et al. [12], 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 $\text{SubsetHN}$ of hanging nodes in the given time interval. Let $u_0$ and $u_1$ respectively denote the left child and right child of the node $u$.

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
Figure 1. Node Labeling

![Node Labeling Diagram]

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 $w_{\text{dep}}$ denotes its depth and $w_{\text{num}}$ denotes its assigned label. Note that $w_{\text{num}}$ is a binary string of length $2^{w_{\text{dep}}}$ − 1 where $0 \leq w_{\text{dep}} \leq l$. 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 $w_t^{l-1}$.

**Algorithm 1:** Nodes$(t_1 + 1, t_2, G) \rightarrow \text{SubsetHN}$

if ($t_1 + 1 > t_2$) then return 0;
else $X_1, X_2, \text{SubsetHN} \leftarrow \emptyset$;

$X_1 \leftarrow X_1 \cup \text{Path}(w_{t_2}^{l-1});$
$X_2 \leftarrow X_2 \cup \text{Path}(w_{t_2}^{l-1});$

for ($w \in X_1$): if ($w_1 \notin X_1 \cup X_2$) then \text{SubsetHN} $\leftarrow$ \text{SubsetHN} $\cup \{w_1\}$
for ($w \in X_2$): if ($w_0 \notin X_1 \cup X_2$) then \text{SubsetHN} $\leftarrow$ \text{SubsetHN} $\cup \{w_0\}$

return \text{SubsetHN};

3. Description of our FSGS scheme

- **FSGS.Setup**$(\lambda, T) \rightarrow (\mathcal{Y}, S_{\text{GM}}, S_{\text{OA}}, St)$: Given a security parameter $\lambda > 0$ and maximum allowable time period $T = 2^l$ for some integer $l$, the key generation center (KGC) chooses an integer $n$ of size $O(\lambda)$, a prime modulus $q$ of size $O(\lambda n^2)$ such that $T \leq q$ and an integer $\mu$ with $\mu \geq \log l$. The system supports at most $N = 2^\mu$ members. The KGC sets $m = 2n \lfloor \log q \rfloor$, selects real numbers $\sigma$ of size $\Omega(\sqrt{n \log q \log n}), \beta$ of size $\sigma \omega(\log m)$ and $\gamma$ of size $\sqrt{n \omega(\log n)}$.

1. It chooses random matrices $M_0, M_1, \{A_k\}_{k=0}^{\mu}, D \leftarrow U(Z_q^{n \times m}), D_0, D_1 \leftarrow U(Z_q^{2n \times 2m}),$ and a vector $u \leftarrow U(Z_q^n)$. It also selects three cryptographically secure hash functions $H : \{0, 1\}^s \rightarrow \{1, 2, 3\}^s, H_0 : \{0, 1\}^* \rightarrow Z_q^{n \times 2m}$ and $H_2 : \{0, 1\}^* \rightarrow \mathcal{I}$, 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 OTS = $(\mathcal{G}, \mathcal{S}, \mathcal{V})$, where $\mathcal{G}, \mathcal{S}, \mathcal{V}$ are the key generation, signing and verification algorithms respectively.

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

3. The KGC runs $\text{TrapGen}(1^n, 1^m, q) \rightarrow (A, T_A)$ and $\text{TrapGen}(1^n, 1^m, q) \rightarrow (B, T_B)$ [3] and outputs matrices $A, B \in Z_q^{n \times m}$ with short bases $T_A$ of $A_q^+(A)$ and $T_B$ of $A_q^+(B)$. The short basis $T_A$ is the secret key $S_{\text{GM}}$ of the GM utilized to issue certificates to the new users while $T_B$ is the master secret key $S_{\text{OA}}$ of the OA employed to trace the signer.

4. Initializes the public state $\text{St} = (\text{Stusers}, \text{Sttrans}) = (\emptyset, \varepsilon)$.
(5) The KGC publishes the group public key $Y = (M_0, M_1, A, \{A_k\})_{k=0}^{\mu}, B, D, D_0, D_1, F, u, OTS, DSig, H, H_0, H_2, \beta, \gamma, \sigma$ and sends $S_{OA} = T_B$ to the opening authority (OA) and $S_{GM} = T_A$ to the group manager (GM) through a secure communication channel.

- **FSGS.Join($Y, T$)$(GM, U_i) \rightarrow (cert_{i,t_1\rightarrow t_2}, sec_{i,t_1\rightarrow t_2}, St)$:** It is an interactive protocol between the GM and the user $U_i$. The GM and the user $U_i$ runs the Turing machines $J_{GM}$ and $J_{user}$ respectively. On completion of the protocol, $U_i$ receives its membership certificate $cert_{i,t_1\rightarrow t_2}$ from the GM through a secure communication channel for the time interval $[t_1, t_2]$ and fixes its secret key $sec_{i,t_1\rightarrow t_2}$ while the GM updates the public state $St = (St_{users}, St_{trans})$ (initially set to be $(\emptyset, \emptyset)$) as follows:

1. The user $U_i$ samples $z_i^{(0)} \leftarrow D_{2^{\omega m}, \sigma}$ i.e., $||z_i^{(0)}|| \leq \sigma \sqrt{4m}$ with high probability whereby $||z_i^{(0)}||_\infty \leq \sigma \omega (\log m) = \beta$. The user $U_i$ computes $v_i^{(0)} = F \cdot z_i^{(0)} \mod q$, $\sigma_i = DSig.Sign(usk[i], v_i^{(0)})$ and sends $(v_i^{(0)}, \sigma_i)$ to the GM.

2. The GM runs $DSig.Verify(usk[i], \sigma_i, v_i^{(0)})$. If the verification fails then the GM aborts. Otherwise, if $v_i^{(0)}$ is not previously used by a registered member, then the GM chooses a $\mu$-bit identity $id_i = (id_i[1], id_i[2], ..., id_i[\mu]) \in \{0, 1\}^\mu$ and a time interval $[t_1, t_2] \subset [1, T]$ for user $U_i$ such that $U_i$ can issue group signature in the interval $[t_1 + 1, t_2]$ and uses the secret key $S_{GM} = T_A$ to issue a new certificate to $U_i$ as follows:

   - The GM computes the matrix $A_{id_i} = [A|\tilde{A}] \in Z_q^{n \times 2m}$ with $\tilde{A} = A_0 + \sum_{k=1}^{\mu} id_i[k]A_k$ where $id_i[k]$ indicates the $k$th bit of the identity $id_i$ of user $U_i$.

   - It runs $ExtBasis(A, A, T_A) \rightarrow (T_{A|\tilde{A}} = T_{A,w})$ [6].

   - The GM samples a short vector $s_i \leftarrow D_{2^{\omega m}, \sigma}$ i.e., $||s_i||_\infty \leq \sigma \omega (\log m) = \beta$. By using the short basis $T_{A_{id_i}}$ of $A_{id_i}$, the GM computes a short vector $d_i = [d_{i,1}, d_{i,2}]^t \in Z^{2m}$ with $||d_i||_\infty \leq \beta$, $d_{i,1}, d_{i,2} \in Z^m$ satisfying

   $$(3.1) \quad A_{id_i}d_i = u + D \cdot r_i \mod q.$$ 

   $$(3.2) \quad r_i = \text{bin}(D_0 \cdot \text{bin}(v_i^{(0)}) + D_1 \cdot s_i) \mod q.$$ 

   - The GM then runs the algorithm Nodes($t_1 + 1, t_2, G$) to identify the collection of subset of hanging nodes SubsetHN, the GM does the following:

(a) compute the matrix $A_{i,w} = [A_{id_i}|\tilde{A}_w]$ where $\tilde{A}_w = [M_0 + wt(w_{num})M_1|A_0 + \sum_{k=1}^{\mu} \alpha_k A_k]$ where $\alpha_1 \alpha_2 ... \alpha_\mu = 0^{\mu-\delta}||\text{bin}(w_{dep})||$, $\alpha_k \in \{0, 1\}$, $\delta = \text{len}(\text{bin}(w_{dep}))$ denotes the length of $\text{bin}(w_{dep})$ and $\text{wt}(w_{num})$ represents the Hamming weight of $w_{num}$.

(b) execute $ExtBasis(A_{id_i}, \tilde{A}, T_{A_{id_i}}) \rightarrow (T_{A_{id_i}|\tilde{A}_w} = T_{A_{i,w}})$ to generate a short basis $T_{A_{i,w}}$ of $A_{i,w}$;

(c) sample short vector $s_{i,w}^{(0)} \leftarrow D_{2^{\omega m}, \sigma}$ i.e., $||s_{i,w}^{(0)}|| \leq \sigma \sqrt{2m}$ with high probability whereby $||s_{i,w}^{(0)}||_\infty \leq \sigma \omega (\log m) = \beta$.

(d) compute a short vector $x_{i,w}^{(0)} = [x_{i,w,1}^{(0)}, x_{i,w,2}^{(0)}, x_{i,w,3}^{(0)}]^t$ with $||x_{i,w}^{(0)}||_\infty \leq \beta$ using $T_{A_{i,w}}$ satisfying

$$(3.3) \quad A_{i,w}x_{i,w}^{(0)} = u + D \cdot \text{bin}(D_0 \cdot \text{bin}(v_i^{(0)}) + D_1 \cdot s_{i,w}^{(0)}) \mod q.$$ 

- The GM computes $H_2(id_i || 0 || r_i^{(0)}) = R_i^{(0)}$, where $r_i^{(0)} \in \{0, 1\}^m$ is randomly chosen $m$ length vector and $R_i^{(0)}$ is a $Z_q$-invertible matrix of size $m \times m$ whose
columns have low norm. It generates $(C_i^{(0)}, T_{C_i^{(0)}}) ← \text{BasisDel}(A_i, R_i^{(0)}, T_A, σ)$ [1], where $C_i^{(0)} = A_i R_i^{(0)}$.

(3) The GM finally issues the membership certificate

cert_{i,t_1→t_2} = (id_i, s_i, \{x_{i,w}^{(0)}, s_{i,w}^{(0)}\}_{w \in \text{SubsetHN}→ \text{Nodes}(t_1+1,t_2,G)}, C_i^{(0)}, T_{C_i^{(0)}}, [t_1,t_2])

to the user $U_i$ through a secure communication channel. The GM updates the public state $St$ by storing $i$ in $St_{\text{users}}$ and transcript, $t_i = (v_i^{(0)}, 1, i, uv[k], \sigma_i, [t_1,t_2])$ in $St_{\text{trans}}$.

(4) The user $U_i$ verifies $||d_i|| < \beta$, $||s_i|| < \beta$ satisfying Eq. 3.1 and for all $w \in \text{SubsetHN} ← \text{Nodes}(t_1+1,t_2,G)$, $||x_{i,w}|| < \beta$ and $||s_{i,w}|| < \beta$ satisfying Eq. 3.3. If any of these verification fails, $U_i$ aborts. Otherwise $U_i$ sets its secret key

$sec_{i,t_1→t_2} = z_i^{(0)}$

and defines

cert_{i,t_1→t_2} = (id_i, s_i, \{x_{i,w}^{(0)}, s_{i,w}^{(0)}\}_{w \in \text{SubsetHN}→ \text{Nodes}(t_1+1,t_2,G)}, C_i^{(0)}, T_{C_i^{(0)}}, [t_1,t_2])

as its membership certificate corresponding to its secret key $sec_{i,t_1→t_2}$.

- **FSGS.Update($\mathcal{Y}$, $t_1$, $t_2$, sec_{i,t(j)→t_2}, cert_{i,t(j)→t_2}) → (sec_{i,t(j+1)→t_2}, cert_{i,t(j+1)→t_2}).**

A user $U_i$ with a valid certificate $cert_{i,t(j)→t_2}$ and the corresponding secret key $sec_{i,t(j)→t_2}$ for the time interval $[t(j), t_2]$, computes $R_i^{(j)} = H_2(id_i || j + 1 || r_i^{(j+1)})$ where $r_i^{(j+1)} \in \{0,1\}^m$ is a randomly chosen $m$ length vector and executes the following steps to output the updated certificate $cert_{i,t(j+1)→t_2}$ and secret key $sec_{i,t(j+1)→t_2}$ for the time interval $[t(j+1), t_2]$, where $t(j) = t_1 + j$ for $j = 0, 1, 2, \ldots, (t_2 - t_1)$.

(1) The user $U_i$ runs $\text{BasisDel}(C_i^{(j)}, R_i^{(j+1)}, T_{C_i^{(j)}}, σ) → (C_i^{(j+1)}, T_{C_i^{(j+1)}})$ to generate $C_i^{(j+1)} = C_i^{(j)} R_i^{(j+1)}$ and a short basis $T_{C_i^{(j+1)}} = \Lambda_i^{(j)}(C_i^{(j+1)})$.

(2) The user $U_i$ solves $C_i^{(j+1)} \cdot z_i^{(j+1)} = \mathcal{Y}_i$ for a vector $z_i^{(j+1)}$ such that $||z_i^{(j+1)}|| < σ \sqrt{m}$ and sets user’s secret key $sec_{i,t(j+1)→t_2} = z_i^{(j+1)}$ for the time period $[t(j+1), t_2]$. The user also defines $\mathcal{Y}_i^{(j+1)} = F \cdot z_i^{(j+1)}$ mod $q$.

(3) For all $(w = (w_{num}, w_{\text{dep}}) \in \text{SubsetHN} ← \text{Nodes}(t^{(j)} + 1, t_2, G)$, user $U_i$ does the following:

- set

$$C_{i,w}^{(j)} = [C_i^{(j)} | C_{id_{i,w}}^{(j)}]$$

with $C_{id_{i,w}}^{(j)} = [A_0 + \sum_{k=1}^{\mu} i_d[k]A_k | M_0 + \text{wt}(w_{num})M_1 | A_0 + \sum_{k=1}^{\mu} \alpha_k A_k]$ where $\alpha_k \in \{0, 1\}$ is the $k$th bit of the $μ$-bit string $(0^{m-2}||\text{bin}(w_{\text{dep}}))$. Define $C_0^{(i)} = A_{i,w}$;

- run $\text{ExtBasis}(C_i^{(j)}, C_{id_{i,w}}^{(j)}, T_{C_i^{(j)}}) → (T_{C_i^{(j)}}, C_{i,w}^{(j)})$ to get a short basis $T_{C_i^{(j)}}, \Lambda_i^{(j)}(C_{i,w}^{(j)})$;

- sample short vectors $s_{i,w}^{(j)} ← D_{Z^{2m}}$ and computes $x_{i,w}^{(j)} = [x_{i,w,1}^{(j)}, x_{i,w,2}^{(j)}, x_{i,w,3}^{(j)}]^t \in Z^{4m}$ with $||x_{i,w}^{(j)}|| < \beta$ and $x_{i,w,1}^{(j)} = [x_{i,w,1,1}, x_{i,w,1,2}, x_{i,w,1,3}] \in Z^{2m}$ satisfying

$$C_{i,w}^{(j)} x_{i,w,1,1} + C_{id_{i,w}}^{(j)} [x_{i,w,1,2}, x_{i,w,1,3}]^t = u + D \cdot m_{i,w}^{(j)}$$ mod $q$,

where $m_{i,w}^{(j)} = \text{bin}((D_0 \cdot \text{bin}(v_{i,w}^{(j)})) + D_1 \cdot s_{i,w}^{(j)})$ mod $q$.  

**Group signature from lattices preserving forward security in dynamic setting**

*Advances in Mathematics of Communications* Volume X, No. X (200X), X–XX
(4) The updated certificate is

\[ \text{cert}_{i,(t+1)\rightarrow t_2} = \left( \text{id}_i, \ d_i, \ s_i, \ \{x_{i,w}^{(j)}, s_{i,w}^{(j)} \}_{w \in \text{SubsetHN}(t, t_2, G)} \right), \ \text{C}^{(j+1)}_{i}, \ \text{T}^{(j+1)}_{c_i}, \ [t^{(j+1)}, t_2] \]

and the associated updated secret is \( \text{sec}_{i,(t+1)\rightarrow t_2} = \{ v_i^{(0)}, z_i^{(j+1)} \} \).

(5) The user \( U_i \) sends \( (v_i^{(j)}, v_i^{(j+1)}), \text{DSig.Sign}(\text{usk}[i], v_i^{(j+1)}) \) to \( \text{GM} \). The GM runs \( \text{DSig.Verify}(\text{usv}[i], \text{DSig.Sign}(\text{usk}[i], v_i^{(j+1)}), v_i^{(j+1)}) \). If verification fails the GM aborts. Otherwise, the GM updates the public state \( \text{St} = (\text{St}_{\text{users}}, \text{St}_{\text{trans}}) \). In particular, the update algorithm updates the \( \text{transcript}_i \) in \( \text{St}_{\text{trans}} \) as \( \text{transcript}_i = (v_i^{(0)}, v_i^{(j)}, v_i^{(j+1)}, i, \text{usv}[i], \text{sig}_i, [t^{(j+1)}, t_2]) \).

(6) After updating the keys for the period \( [t^{(j+1)}, t_2] \), the algorithm erases \( \text{cert}_{i,(t)\rightarrow t_2} \) and \( \text{sec}_{i,(t)\rightarrow t_2} \) for the time period \( [t^{(j)}, t_2] \).

\[ \text{FSGS.Sign}(\text{sec}_{i,(t)\rightarrow t_2}, \text{cert}_{i,(t)\rightarrow t_2}, \gamma, M) \rightarrow (\sigma) \): To sign a message \( M \in \{0, 1\}^* \), the user \( U_i \) generates a one time signature signing-verify key-pair \((\text{VK}, \text{SK})\) by running OTS \( \mathcal{O} \) and performs the following steps:

(1) Compute \( \text{G}_0 = \text{H}_0(\text{VK}) \in \mathbb{Z}_q^{2m} \), randomly chooses \( e_0 \leftarrow \chi^n \), \( e_1 \leftarrow \chi^m \), \( e_2 \leftarrow \chi^{2m} \), where \( \chi \) is a \( \gamma \)-bounded distribution, extracts \( F, B \) from \( \gamma \), \( z_i^{(j)} \) from \( \text{sec}_{i,(t)\rightarrow t_2} \), calculates

\[ v_i^{(j)} = F \cdot z_i^{(j)}, \]

(3.7)

(3.8)

\[ y_i^{(j)} = \text{bin}(v_i^{(j)}) \in \{0, 1\}^{2m} \]

and generates the ciphertext by using the encryption of GPV-IBE \([7]\)

\[ \text{c}_{v_i^{(j)}} = (c_1, c_2) = (B_i^t e_0 + e_1, \text{G}_0^t e_0 + e_2 + y_i^{(j)} \cdot g/2) \in \mathbb{Z}_q^m \times \mathbb{Z}_q^{2m}. \]

(2) For each signature, the \( i \)-th signer chooses a new random \( R_i \in \mathbb{Z}_q^{n \times n} \) and computes \( \text{C}^{(j)}_{i,R} = R_i \cdot \text{C}^{(j)}_i \). Note that \( \text{C}^{(j)}_i x_i^{(j)}_{1,w,1} \equiv 0 \mod q \Rightarrow R_i \cdot \text{C}^{(j)}_i x_i^{(j)}_{1,w,1} \equiv 0 \mod q \) and thus basis \( \text{T}^{(j)}_{c_i} \) of \( \Lambda_q^{\perp}(\text{C}^{(j)}_{i,R}) \) serves as a basis \( \text{T}^{(j)}_{c_i} \) of \( \Lambda_q^{\perp}(\text{C}^{(j)}_{i,R}) \).

Random matrix \( R_i \) helps in maintaining anonymity.

(3) Rewrite Eq. 3.4 as

\[ \text{C}^{(j)}_{i,w} = [\text{C}^{(j)}_{i,R}] M_0 + \text{wt}(w_{\text{num}}) M_1 + \sum_{k=1}^{\mu} \alpha_k A_k \]

where \( \text{C}^{(j)}_{i,R} = [\text{C}^{(j)}_{i,R}] A_0 + \sum_{k=1}^{\mu} \text{id}_i[k] A_k \). Let \( g = \{1, 2, 2^2, \ldots, 2^{\lfloor \log q \rfloor - 1}\} \) then \( \text{H}_{2n \times m} = I_{2n \times 2n} \otimes g \) and \( \text{H}_{1n \times 2m} = I_{4n \times 4n} \otimes g \) be power of 2 matrices. We perform the following steps in order to prove the knowledge of \( s_i, d_i, z_i^{(j)} \) with infinity norm bound \( \beta, x_i^{(j)}_{1,w}, s_i^{(j)}_{1,w} \), for all \( w \in \text{SubsetHN} \Rightarrow \text{Nodes}(t^{(j)} + 1, t_2, G) \) with infinity norm bound \( \beta, e_0, e_1, e_2 \) with infinity norm bound \( \gamma \), \( r_i, m_i^{(j)} \), satisfying Eq 3.2, Eq 3.6 respectively and \( \text{id}_i \in \{0, 1\}^{\mu} \), \( y_i^{(j)} = \text{bin}(v_i^{(j)}) \) satisfying the following system of equations:
We write the above system of equations in the form $P\cdot x = v$ where $P, v$ are public and $x$ is secret and employ the non-interactive ZKAoK [11] protocol for zero knowledge proof of argument to generate a proof $\Pi_i = \left(\{\text{COM}_{i,k}\}_{k=1}^s, z_{i}, \{\text{COM}_{j,k}\}_{k=1}^s \right)$ to prove the knowledge of the witness $x$ for the relation $R = \{(P, v) \in \mathbb{Z}_q^{D \times L} \times \mathbb{Z}_q^D, x \in \text{VALID} : P\cdot x = v \mod q \}$ satisfying $P\cdot x = v$.

(4) Compute one time signature, $\text{osig}_i = \text{OTS}.S(\text{SK}, c_{v^{(j)}}^{i}, \Pi_i)$, and finally outputs the message signature pair $(M, \sigma)$ where

$$\sigma = \langle \text{VK}, c_{v^{(j)}}, \Pi, \text{osig}_i, C_{i,R}^{(j)} \rangle.$$

- $\text{FSGS}.\text{Verify}(\sigma, t, M, \mathcal{Y}) \rightarrow (1 \lor 0)$: The verifier parses $\sigma$ as in Eq (3.12) and returns 1 if and only (i) $t \in [t_1, t_2]$, (ii) $\text{OTS}.\text{V}(\text{VK}, (c_{v^{(j)}}^{i}, \Pi_i), \text{osig}_i) = 1$ and (iii) the proof of knowledge $\Pi_i$ properly verifies.

- $\text{FSGS}.\text{Open}(\sigma, t, M, \mathcal{Y}, \text{SOA}, \text{St}) \rightarrow (i \lor \perp)$: The opening authority parses $\sigma$ as in Eq 3.12 and executes the following steps using its secret key $\text{SOA} = \text{T}_B$.

1. Computes $G_0 = H_0(\text{VK}) \in \mathbb{Z}_q^{2m}$ and use $\text{T}_B$ to compute a small norm matrix $\text{K}_{\text{VK}} \in \mathbb{Z}_q^{m \times 2m}$ using the algorithm SamplePre [7].
2. Decrypts $c_{v^{(j)}}^{i}$ using the secret key $\text{K}_{\text{VK}}$ and obtains $\text{bin}(v_2) \in \{0, 1\}^{2m}$. It checks if $v_2 = \text{H} \cdot \text{bin}(v_2) \mod q$ appears in a record transcript$_i = \langle v, v_1, v_2, i, uvk[i], \text{sig}_i, [t_1, t_2] \rangle$ of the database $\text{St}_{\text{trans}}$ for some $i$. If so, outputs $i$ indicating that the user $\mathcal{U}_i$ is the signer. Otherwise, outputs $\perp$. Note that if the obtained $v$ is available as one of the records in the transcript, then the OA will output the corresponding index $i$ appears in the corresponding tuple.

### 3.0.1. Correctness

The correctness of $\text{FSGS}$ is as follows:

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

(b) If $(i, \text{sec}_{i,t_1 \rightarrow t_2}, \text{cert}_{i,t_1 \rightarrow t_2})$ is obtained by $\mathcal{U}_i$ (i.e., $\mathcal{U}_i$) on honestly running \( \lfloor \text{user}(\mathcal{Y}, T), \text{J}_G(\mathcal{Y}, t_1, t_2, T, \text{SOA}, \text{St}) \rfloor \), then $\text{cert}_{i,t_1 \rightarrow t_2} := \text{y enc}_{i,t_1 \rightarrow t_2}$. Here $\mathcal{Y}, \text{G}_{\text{SK}}, \text{SOA}, \text{St}$ are generated by executing $\text{FSGS}.\text{Setup}(\lambda, T)$.

(c) Let $\text{cert}_{i,t_1 \rightarrow t_2} := \text{y enc}_{i,t_1 \rightarrow t_2}$ be as in (b) and $(i, \text{sec}_{i,t \rightarrow t_2}, \text{cert}_{i,t \rightarrow t_2})$ for $t_1 \leq t \leq t_2$ is obtained by running $\text{FSGS}.\text{Update}(\mathcal{Y}, k, t_2, \text{sec}_{i,k \rightarrow t_2}, \text{cert}_{i,k \rightarrow t_2} \rightarrow (\text{sec}_{i,k+1 \rightarrow t_2}, \text{cert}_{i,k+1 \rightarrow t_2})$, successively for $k = t_1, t_1 + 1, ..., t - 1$. Then $\text{cert}_{i,t \rightarrow t_2} := \text{y enc}_{i,t \rightarrow t_2}$ and $\text{FSGS}.\text{Verify}(\text{FSGS}.\text{Sign}(\text{sec}_{i,t_1 \rightarrow t_2}, \text{cert}_{i,t \rightarrow t_2}, \mathcal{Y}, M_i, t, M, \mathcal{Y}), 1) = 1$.

(d) For any $(i, \text{cert}_{i,t_1 \rightarrow t_2}, \text{sec}_{i,t_1 \rightarrow t_2})$ obtained by $\mathcal{U}_i$ on execution of $\lfloor \mathcal{U}_i(\mathcal{Y}, t_1, t_2, T), \text{J}_G(\mathcal{Y}, t_1, t_2, T, \text{SOA}, \text{St}) \rfloor$ for some valid state $\mathcal{St}_i$, let $(i, \text{cert}_{i,t_1 \rightarrow t_2}, \text{sec}_{i,t_1 \rightarrow t_2})$
be derived by FSGS.Update algorithm as in (c). Now if \( \sigma = \text{FSGS.Sign} (\text{sec}_{i,t} \leftarrow t_2, \text{cert}_{i,t} \leftarrow t_2, Y, M) \), then FSGS.Open(\( \sigma, t, M, Y, \text{St}_{\text{OA}}, \text{St}_t \rangle = i \).

**Remark 3.1.** Note that in our construction, we do not expose \( [A_0 + \sum_{i=1}^{\mu} \text{id}_i [k] A_k] \) in public while executing zero-knowledge between the prover the verifier which preserves the anonymity of our scheme. If \( [A_0 + \sum_{i=1}^{\mu} \text{id}_i [k] A_k] \) gets exposed then we can construct an adversary breaking the anonymity as shown below:

- By making two join queries for two users \( U_0 \) and \( U_1 \), the adversary obtains two pairs \( (\text{sec}_0^i, \text{cert}_0^i) \) and \( (\text{sec}_1^i, \text{cert}_1^i) \). Note that adversary knows \( \text{id}_0 \) and \( \text{id}_1 \) for \( U_0 \) and \( U_1 \) respectively.
- The challenger returns \( \sigma^* \) signed by \( U_d \) where \( d \in \{0, 1\} \).
- Given \( \sigma^* \), the adversary checks the matrix \( C_{i,d}^{(j)} \) in \( \sigma^* \). Adversary can determine \( d \) by checking whether the right block of the matrix \( C_{i,d}^{(j)} \) is equivalent to \( [A_0 + \sum_{i=1}^{\mu} \text{id}_i [k] A_k] \) or not.

**Remark 3.2.** Our scheme withstands mis-identification attack because after a user’s secret key and certificate gets updated, the update algorithm updates the transcript in \( \text{St}_{\text{trans}} \). The updated transcript helps the open algorithm to run and enables the OA to find the record available in the transcript. If the update algorithm does not update the transcript then we can construct an adversary against mis-identification attack as follows:

- The adversary queries to \( Q_0 \cdot \text{join} \) with \( [t_1, t_2] = [1, T] \) and obtains user’s secret key and certificate. The database contains transcript\( _t = (v_1^{(0)}, 1, uv[k], sig, [1, T]) \).
- The adversary updates the certificate and secret key. In this procedure, the adversary randomly chooses \( z_1^{(j)} \) from a Gaussian distribution and computes \( v_1^{(1)} = F \cdot z_1^{(1)} \mod q \). As the transcript is not updated by the update algorithm, \( v_1^{(1)} \) is not available in the transcript.
- The adversary generates the signature \( \sigma^* \) using the updated certificate and secret key. Consequently, \( \sigma^* \) contains the updated value \( v_1^{(1)} \) instead of \( v_1^{(0)} \).
- The adversary outputs \( M^*, t^* = 1 \), and \( \sigma^* \).
- The OA will output nothing since there are no transcript = \( (v, i, uv[k], sig_i, [t_1, t_2]) \) such that \( v_1^{(1)} = v_1^{(1)} \).

3.1. The underlying zero knowledge for our scheme. It is the argument system for the zero knowledge \( \Pi_1 \). The argument system upon which our group signature scheme is built can be summarized as follows:

**Common Input:** \( C_{i,R}^{(j)} \in \mathbb{Z}_q^{n \times 2m}, M_0, M_1 \in \mathbb{Z}_q^{n \times m}, A_{id_1} \in \mathbb{Z}_q^{n \times 2m}, A, A_0, \cdots, A_{\mu}, D \in \mathbb{Z}_q^{n \times m}, D_0, D_1 \in \mathbb{Z}_q^{2n \times 2m}, F \in \mathbb{Z}_q^{4n \times 4m}, H \in \mathbb{Z}_q^{2n \times m}, H \in \mathbb{Z}_q^{4n \times 2m}, B \in \mathbb{Z}_q^{n \times m}, G_0 \in \mathbb{Z}_q^{n \times 2m}, u \in \mathbb{Z}_q^{n}, c_1 \in \mathbb{Z}_q^{m}, c_2 \in \mathbb{Z}_q^{2m} \).

**Prover’s Input:** \( x_{i,w,1} \in [-\beta, \beta]^{2m}, x_{i,w,2} \in [-\beta, \beta]^{m}, x_{i,w,3} \in [-\beta, \beta]^{m}, y_i \in \{0, 1\}^{2m}, y_j \in \{0, 1\}^{2m}, s_{i,j} \in [-\beta, \beta]^{2m}, m_{i,w} \in \{0, 1\}^m, r_i \in \{0, 1\}^m, s_i \in [-\beta, \beta]^{2m}, d_{i,j}, d_{i,2} \in [-\beta, \beta]^{m}, z_{i,j} \in [-\beta, \beta]^{4m}, v_{i,j} \in [-\beta, \beta]^{4m}, id_i = (id_i[1], id_i[2], \cdots, id_i[m]) \in \{0, 1\}^m, e_0 \in [-\gamma, \gamma]^{2m}, e_1 \in [-\gamma, \gamma]^{2m}, e_2 \in [-\gamma, \gamma]^{2m}. \) Note that \( x \in [-\beta, \beta]^{m} \) means \( x \in \mathbb{Z}_q \) with \( ||x||_\infty \leq \beta \).

**Prover’s Goal:** Convince the verifier in zero knowledge of the system of equations
given in Eq 3.11. To achieve this, we rewrite Eq 3.11 as

\[(3.13) \quad N_0x_0 + N_1x_1 + N_2x_2 + N_3x_3 + N_4x_4 = v \mod q,\]

\[N_0 = \begin{bmatrix} A_{n \times m} & 0_{n \times 2m} & 0_{n \times 4m} \\ 0_{2n \times m} & (D_1)_{2n \times 2m} & 0_{2n \times 4m} \\ 0_{4n \times m} & 0_{4n \times 2m} & F_{4n \times 4m} \end{bmatrix},\]

\[N_1 = \begin{bmatrix} A_1' | A_2' | \cdots | A_m' \end{bmatrix} \in \mathbb{F}_{(3m+11n) \times (m+1)m} \text{ with } A_i' = [A_i | 0]_{(3m+11n) \times m},\]

\[N_2 = \begin{bmatrix} (-D)_{m \times m} & 0_{2n \times 2m} & 0_{2n \times m} & 0_{2n \times 2m} \\ -H_{2n \times m} & (D_0)_{2n \times 2m} & 0_{2n \times m} & 0_{2n \times 2m} \\ 0_{4n \times m} & 0_{4n \times m} & 0_{4n \times m} & -H_{4n \times 2m} \\ 0_{2m \times m} & 0_{2n \times 2m} & 0_{2n \times m} & 0_{2n \times 2m} \end{bmatrix},\]

\[N_3 = \begin{bmatrix} 0_{7n \times n} & 0_{7n \times m} & 0_{7n \times 2m} \\ 0 & B_{m \times n} & 1_{m \times 2m} \\ 0 & 0_{n \times m} & 0_{n \times 2m} \end{bmatrix},\]

\[N_4 = \begin{bmatrix} 0_{(7n+3m) \times (m+2)} & 0_{(7n+3m) \times m} & 0_{2n \times 2m} \\ 0_{2n \times m} & 0_{2n \times m} & 0_{2n \times 2m} \\ 0_{2n \times m} & 0_{2n \times m} & 0_{2n \times 2m} \end{bmatrix},\]

\[Z = \begin{bmatrix} C_{i,j}^n_{1,m} | A_0 | A_1 \cdots | A_m \end{bmatrix}_{n \times (m+2)}, \quad x_0 = \begin{bmatrix} d_{i,1} & s_i & z_i^{(j)} \end{bmatrix}^t,\]

\[x_1 = \begin{bmatrix} d_{i,1} | [d_{i,1}]_1 | [d_{i,1}]_2 | \cdots | [d_{i,1}]_n \end{bmatrix}^t,\]

\[x_2 = \begin{bmatrix} r_i & 0 & m_{i,w} & y_i^{(j)} \end{bmatrix}^t,\]

\[x_3 = \begin{bmatrix} e_0 & e_1 & e_2 \end{bmatrix}^t,\]

\[x_4 = \begin{bmatrix} x' & x_{i,w,2}' & x_{i,w,3}' & s_{i,w}' \end{bmatrix}^t,\]

\[x' = \begin{bmatrix} x_{i,w,1}' \cdots | [d_{i,1}]_n \cdots | [d_{i,1}]_n \cdots \end{bmatrix}^t,\]

\[v = \begin{bmatrix} u_0 & 0 & 0 & e_1 & e_2 & u_0 \end{bmatrix}^t.\]

Let \(\tilde{p} = \log \gamma + 1, p = \log \beta + 1\) and \(B^2\) be the collection of 2l length strings over the alphabet \(\{0, 1\}\) having equal number of 0s and 1s. Let \(B^3\) be the collection of 3l length strings over the alphabet \(\{-1, 0, 1\}\) having equal number of 0s, 1s, and \(-1\)s.

Let \(N_1'\) and \(\bar{x}_i\) for \(i = 0, 1, \ldots, 4\) are obtained as follows using the technique Dec-Ext \[13\] and \(\tilde{K}_{m, \beta}\) with the property that \(\tilde{K}_{m, \beta} \tilde{x} = x.\)

Dec-Ext \(7m, p(x_0) \rightarrow (\tilde{x}_0 \in B_{3mp}^3, N_{0, \tilde{K}_{7m, \beta}} \rightarrow (N'_{0} \in \mathbb{Z}_q^{(3m+11n) \times (7m)p}),\)

Dec-Ext \((\mu+1)m, p(x_1) \rightarrow (\tilde{x}_1 \in B_{3(\mu+1)p}^3, N_{1, \tilde{K}_{(\mu+1)m, \beta}} \rightarrow (N'_{1} \in \mathbb{Z}_q^{(3m+11n) \times (\mu+1)mp}),\)

Dec-Ext \(6m, p(x_2) \rightarrow (\tilde{x}_2 \in B_{6mp}^3, N_{2} \in \mathbb{Z}_q^{(3m+11n) \times 2(6m)p}),\)

Dec-Ext \((3m+n), p(x_3) \rightarrow (\tilde{x}_3 \in B_{3(3m+n)p}^3, N_{3, \tilde{K}_{(3m+n), \beta}} \rightarrow (N'_{3} \in \mathbb{Z}_q^{(3m+11n) \times 3(3m+n)p}),\)

Dec-Ext \(6m, p(x_4) \rightarrow (\tilde{x}_4 \in B_{(\mu+6)p}^3, N_{4, \tilde{K}_{6m, \beta}} \rightarrow (N'_{4} \in \mathbb{Z}_q^{(3m+11n) \times 3(\mu+6)mp}),\)

Next we set

\[P = [N'_{0} \mid N'_{1} \mid N'_{2} \mid N'_{3} \mid N'_{4}] \in \mathbb{Z}_q^{5(3m+11n) \times ((52+4\gamma)mp+9m+3n)p},\]

\[x = [\tilde{x}_0 \mid \tilde{x}_1 \mid \tilde{x}_2 \mid \tilde{x}_3 \mid \tilde{x}_4] \in \mathbb{Z}_q^{(52+4\gamma)mp+9m+3n}p.\]
$L = (52 + 4\mu)mp + (9m + 3n)p$, $D = 5(3m + 11n)$.

$\text{VALID} = \{w \in \{-1, 0, 1\}^L : w = (w_0^1 || w_1^1 || w_2^1 || w_3^1) \text{ for some } w_0 \in B_3^{2mp}, w_1 \in B_{(\mu+1)mp}, w_2 \in B_{2mp}, w_3 \in B_{(3m+n)p}, w_4 \in B_{2(\mu+6)mp}\}$. Then $x \in \text{VALID}$. Note that $N_0 \tilde{K}_{m,\beta} \tilde{x}_0 + N_1 \tilde{K}_{m,\beta} \tilde{x}_1 + [N_2](0^{(3m+11n)\times 6mp}) \tilde{x}_2 + N_3 \tilde{K}_{m,\gamma} \tilde{x}_3 + N_4 \tilde{K}_{m,\beta} \tilde{x}_4 = N_0x_0 + N_1x_1 + N_2x_2 + N_3x_3 + N_4x_4 = v \mod q$.

Let us now define the set $S$ and permutations of $L$ elements $T_\pi : \pi \in S$ satisfying the following conditions:

(i) $z \in \text{VALID} \iff T_\pi(z) \in \text{VALID}$

(ii) $z \in \text{VALID} \Rightarrow T_\pi(z)$ is uniform in $\text{VALID}$ whenever $\pi$ is uniform in $S$.

Let us define $S = S_{21mp} \times S_{3(\mu+1)mp} \times S_{12mp} \times S_{(9m+3n)p} \times S_{(3m+18)mp}$.

Then for any randomly selected $\pi = (\pi_0, \pi_1, \pi_2, \pi_3, \pi_4)$ and $z = (z_0, z_1, z_2, z_3, z_4) \in \text{VALID}$, we have $T_\pi(z) = (\pi_0(z_0), \pi_1(z_1), \pi_2(z_2), \pi_3(z_3), \pi_4(z_4))$ satisfying above conditions (i) and (ii). Finally, we invoke the algorithm $\text{ZKAoK}$ for the relation $\mathcal{R} = \{(P, v) \in \mathbb{Z}_q^D \times \mathbb{Z}_q^D, x \in \text{VALID} : Px = v \mod q\}$ for statistical zero knowledge argument of knowledge.

3.2. DYNAMIC FORWARD SECURE GROUP SIGNATURE SCHEME (FSGS) [12]. The adversary $A$ is given access to the following oracles while mounting attacks.

- $Q_{\text{pub}}$, $Q_{\text{keyGM}}$, $Q_{\text{keyOA}}$. When adversary $A$ invokes the oracles $Q_{\text{pub}}$, $Q_{\text{keyGM}}$, $Q_{\text{keyOA}}$, the interface $I$ with state $I$ returns $Y$, $S_{\text{GM}}$, $S_{\text{OA}}$ respectively to $A$ by extracting them from $\text{State}_I$.

- $Q_{\text{a}}$. Join. The adversary $A$ can introduce users under its control (users in $U^a$) by querying the oracle $Q_{\text{a}}$. Join. On input $i_1, t_2 \in \{1, 2, ..., T\}$, the interface $I$ executes $\text{FSGS.Join}$ acting as the GM that controls the Turing machine $J_{\text{GM}}$ and interacts with a malicious user $U_i$ that controls the Turing machine $J_{\text{user}}$. If the protocol is successful, then the interface $I$ increments $U$ by 1, and updates $\text{St} = (\text{St}_{\text{users}}, \text{St}_\text{trans})$ adding index $i$ of the malicious user $U_i$ to both $U^a$ and $\text{St}_{\text{users}}$, and updating $\text{St}_\text{trans}$ as $\text{St}_\text{trans} = \text{St}_\text{trans}(i,t_1,t_2,\text{transcript})$.

- $Q_{\text{b}}$. Join. This query enables the adversary $A$ to introduce honest users while acting as the dishonest GM. The adversary $A$ acts as the GM and handles the Turing machine $J_{\text{GM}}$ in executing algorithm $\text{FSGS.Join}$ while the Turing machine $J_{\text{user}}$ is controlled by an honest user $U_i$, 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 $\text{St}_{\text{users}}$, and updates $\text{St}_\text{trans}$ as $\text{St}_\text{trans} = \text{St}_\text{trans}(i,t_1,t_2,\text{transcript})$. The Interface $I$ finally stores the membership certificate $\text{cert}_{i,t_1,t_2}$ in a private part of $\text{State}_I$.

- $Q_{\text{sig}}(M, i, t) \rightarrow (\sigma \vee \bot)$. 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 $\text{State}_{\text{pri}}$ contains $\text{sec}_{i,t_1,t_2}, \text{cert}_{i,t_1,t_2}$ for some $t_1, t_2 \in \{1, 2, ..., T\}$ with $t_1 \leq t \leq t_2$ and $U_i \in U^b$. If so, $I$ calls $\text{FSGS.Update}(Y, k, t_2, \text{sec}_{i,t-k,t_2}, \text{cert}_{i,t-k,t_2})$ for $k = t_1, t_1 + 1, ..., t$ to generate the pair $(\text{cert}_{i,t,t_2}, \text{sec}_{i,t-t_2})$. Generates a signature $\sigma \leftarrow \text{FSGS.Sign}((\text{sec}_{i,t-t_2}, \text{cert}_{i,t-t_2}, Y, M))$ to $A$ on behalf of user $U_i$ for the period $t$ and updates $\text{Sigs} = \text{Sigs}(i, t, M, \sigma)$. Otherwise, the interface $I$ returns $\bot$ to $A$.

- $Q_{\text{open}}(M, \sigma, t) \rightarrow (i \vee \bot)$. Given a valid tuple $(M, \sigma, t)$ from the adversary, the interface $I$ runs the opening algorithm $\text{FSGS.Open}(\sigma, t, M, Y, S_{\text{OA}}, \text{St}) \rightarrow (i \vee \bot)$ with the current state $\text{St} = (\text{St}_{\text{users}}, \text{St}_\text{trans})$. Let $S = \{(M, \sigma, t)| \text{FSGS.Verify}(\sigma, t, M, Y) = 1\}$ and $Q_{\text{open}}$ denotes a restricted oracle that applies opening algorithm $\text{FSGS.Open}$ only for the valid tuples $(M, \sigma, t)$ which are not in $S$. 


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

• $Q_{\text{corrupt}}(i,t) \rightarrow ((\text{cert}_{i,t\rightarrow t_{2}}, \text{sec}_{i,t\rightarrow t_{2}}) \vee \bot)$. On receiving query $(i,t)$, where $i \in \text{St}_{\text{users}}$ and $t \in \{1, 2, ..., T\}$ from $A$, the interface $I$ checks if $i \in U^{b}$ and $\text{St}_{\text{trans}}$ has a record of the form $(i, t_{1}, t_{2}, \text{transcript}_{i})$ for some $t_{1}, t_{2} \in \{1, 2, ..., T\}$ with $t_{1} \leq t \leq t_{2}$. If not, returns $\bot$. Else, $I$ extracts $\text{cert}_{i,t_{1}\rightarrow t_{2}}, \text{sec}_{i,t_{1}\rightarrow t_{2}}$ from $\text{State}_{\text{pri}}$ and iteratively call algorithm $\text{FSGS.Update}(\mathcal{Y}, k, t_{2}, \text{sec}_{k,t_{2}})$ for $k = t_{1}, t_{1} + 1, ..., t - 1$ to generate $\text{cert}_{i,t\rightarrow t_{2}}$ and $\text{sec}_{i,t\rightarrow t_{2}}$ for $t > t_{1}$. It provides all these information to the adversary and stores $(i,t)$ in $\text{St}_{\text{corr}}$. 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 $\text{Exp}_{A}^{\text{mis-id}}(\lambda, T)$ described in Algorithm 2 between an adversary $A$ and the interface $I$. In this attack, $A$ gets the public parameter $\mathcal{Y}$ by $Q_{\text{pub}}$ query and $\text{State}_{\text{pub}}$ by $Q_{\text{read}}$ query. Besides, $A$ has the power to manipulate the opening authority given access to $\text{St}_{\text{OA}}$ by querying $Q_{\text{keyOA}}$ oracle. It can also launch new dishonest members in the group by making $Q_{\text{a-join}}$ queries whereby $\text{St}$ gets updated and $A$ receives all the secret information of the dishonest members. The aim of the adversary $A$ is to produce a valid signature $\sigma^{*}$ 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 3.3. (Mis-identification)** A FSGS scheme over $T$ periods is secure against mis-identification attack if $\text{Adv}_{A}^{\text{mis-id}}(\lambda, T) = \Pr[\text{Exp}_{A}^{\text{mis-id}}(\lambda, T) = 1] \in \text{negl}(\lambda)$, where $\text{negl}$ is a negligible function in $\lambda$.

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

\[
\text{State}_{I} = (\mathcal{Y}, \text{St}_{\text{GM}}, \text{St}_{\text{OA}}) \leftarrow \text{FSGS.Setup}(\lambda, T);
(\mathcal{M}^{*}, t^{*}, \sigma^{*}) \leftarrow A(\text{pub}, Q_{\text{a-join}}, Q_{\text{read}}, Q_{\text{keyOA}});
\]
if (FSGS.Verify($\sigma^{*}, t^{*}, \mathcal{M}^{*}$) = 0) then return 0;
else $i = \text{FSGS.Open}(\sigma^{*}, t^{*}, \mathcal{M}^{*}, \mathcal{Y}, \text{St}_{\text{OA}}, \text{St})$;
if ($i \notin U^{a}$) then return 1;
else if ($(i, t_{1}, t_{2}, \text{transcript}_{i}) \in \text{St}_{\text{trans}} \land [t_{1} \leq t^{*} \leq t_{2}]$) then return 0;
else return 1;

2. **Framing Attack**: This attack game is formally described by the experiment $\text{Exp}_{A}^{\text{fr}}(\lambda)$ in Algorithm 3 which is played between an adversary $A$ and the interface $I$. The adversary $A$ 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 $A$ is capable to access the secret keys of both the GM and the OA. Acting as a dishonest group manager, $A$ can introduce honest members using $Q_{\text{join}}$ queries. At the same moment, $A$ can corrupt the honest members of $U^{b}$ by invoking $Q_{\text{corrupt}}$ queries. It also has the power to observe the system when user produce signatures with $Q_{\text{sig}}$ queries, and can create dummy users with $Q_{\text{write}}$ queries.
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 3.4. (Framing) A FSGS scheme over T periods is secure against framing attack if $\text{Adv}^{\text{fr}}_A(\lambda) = Pr[\text{Exp}^{\text{fr}}_A(\lambda) = 1] \in \text{negl}(\lambda)$, where $\text{negl}$ is a negligible function in $\lambda$.

Algorithm 3: $\text{Exp}^{\text{fr}}_A(\lambda)$

State$_i = (\text{St}_i, \text{Y}_i, \text{SGM}_i, \text{SOA}_i) \leftarrow \text{FSGS.Setup}(\lambda, T);
(M^*, t^*, \sigma^*) \leftarrow A(Q_{\text{pub}}, Q_{\text{keyGM}}, Q_{\text{b-join}}, Q_{\text{corrupt}}, Q_{\text{write}}, Q_{\text{read}}, Q_{\text{keyOA}});
if (\text{FSGS.Verify}(\sigma^*, t^*, M^*, \text{Y}^*) = 0) \text{ then return } 0;
else i = \text{FSGS.Open}(\sigma^*, t^*, M^*, \text{Y}^*, \text{SOA}, \text{St});
if((\exists (i, t_1, t_2, \text{transcript}) \in \text{St}_{\text{trans}} \text{ such that } (t_1 \leq t^* \leq t_2)) \lor (\exists (i, t') \in \text{St}_{\text{corr}} \text{ given } t' \leq t^*)) \text{ then return } 0;
else if($\forall j \in \mathbb{U}$ such that $j \neq i$($j, t^*, M^*, \ast) \notin \text{Sigs}$) \text{ then return } 1;
else \text{ return } 0;

(3) Anonymity Attack: The formal notion of this attack game is modeled by the experiment $\text{Exp}^{\text{an}}_A(\lambda, T)$ outlined in Algorithm 4 between an adversary $A$ and the interface $I$. It consists of 2 stages. The first stage is the play stage, where the adversary $A$ is allowed to make $Q_{\text{write}}$ queries and has access to $Q_{\text{open}}$ oracle. At the end of this stage, $A$ chooses $(M^*, t^*)$ and two pairs $(\text{sec}_i, t^* \rightarrow t^*_1, \text{cert}_i, t^* \rightarrow t^*_1)$ ($k = 0, 1$) consisting of a well formed membership certificate and a membership certificate for periods $[t^*, t^*_2]$ for $b \in \{0, 1\}$, and auxiliary information $\text{aux}$. The interface generates a signature $\sigma^*$ using $(\text{sec}_i, t^* \rightarrow t^*_1, \text{cert}_i, t^* \rightarrow t^*_1)$ by flipping a fair coin $d \in \{0, 1\}$. The aim of the adversary $A$ is to output a guess $d'$ for the guess stage.

Definition 3.5. (Anonymity) A FSGS scheme is fully anonymous for any PPT adversary $A$, if $\text{Adv}^{\text{an}}(A) := |Pr[\text{Exp}^{\text{an}}_A(\lambda) = 1] - 1/2| \in \text{negl}(\lambda)$, where $\text{negl}$ is a negligible function in $\lambda$.

Algorithm 4: $\text{Exp}^{\text{an}}_A(\lambda)$

State$_1 = (\text{St}_1, \text{Y}_1, \text{SGM}_1, \text{SOA}_1) \leftarrow \text{FSGS.Setup}(\lambda, T);
(\text{aux}, M^*, t^*, (\text{sec}_i, t^* \rightarrow t^*_1), (\text{sec}_i, t^* \rightarrow t^*_2), (\text{sec}_i, t^* \rightarrow t^*_1), (\text{sec}_i, t^* \rightarrow t^*_2)) \leftarrow A(\text{Play}, Q_{\text{pub}}, Q_{\text{keyGM}}, Q_{\text{open}}, Q_{\text{read}}, Q_{\text{write}});
if (\{\text{cert}_{i_0}, t^* \rightarrow t^*_1 \text{ where } b \in \{0, 1\} \} \lor \{\text{cert}_{i_0}, t^* \rightarrow t^*_0 = \text{cert}_{i_1}, t^* \rightarrow t^*_1\}) \text{ then return } 0;
else d \leftarrow U\{0, 1\};
\sigma^* \leftarrow \text{FSGS.Sign}(\text{sec}_{i_0}, t^* \rightarrow t^*_2, \text{cert}_{i_0}, t^* \rightarrow t^*_2, Y^*, M^*);
d' \leftarrow A(\text{guess}, \sigma^*, \text{aux} : Q_{\text{pub}}, Q_{\text{keyGM}}, Q_{\text{open}}(M^*, \sigma^*, t^*), Q_{\text{read}}, Q_{\text{write}});
if (d' = d) \text{ then return } 1;
else \text{ return } 0;

4. Security

Theorem 4.1. The group signature scheme FSGS described in Section 3 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 $A$ be a PPT adversary that breaks the security of our group signature scheme $FSGS$ with non-negligible advantage $\epsilon$. We will construct a simulator $B$ that solves the SIS problem using $A$ as a subroutine i.e., given $A = [A_1|A_2] \in \mathbb{Z}_q^{n \times 2m}$ with $m = \ell n \log q$ where $A_1, A_2 \in \mathbb{Z}_q^{n \times m}$ and a real number $\beta$, the simulator $B$ interacts with $A$ and finds a vector $z \in \mathbb{Z}_q^{2m}$ with $|z| \leq \beta$ satisfying $A z = 0 \mod q$.

The simulator $B$ acts as the KGC as well as the GM. It first chooses randomly $\tilde{id} \leftarrow U([0,1])$, $t^* \leftarrow U([1,\delta])$ and $\ell \in [1, T]$ where $\delta$ 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 $\mu \geq \log l$. After repeated executions of the adversary $A$, let $\tilde{id}^* \in \{0, 1\}^\mu$ be the identifier in the witness $x$ revealed by the knowledge extractor of the proof system for $P \leftarrow v$ associated in $A$’s forgery $(M^*, \sigma^*, t^*)$ where $\sigma^*$ is a forged signature on a message $M^*$ at time $t^*$. The identifier $\tilde{id}$ chosen by $B$ at the beginning before the Setup works as a suspicion of the identifier $\tilde{id}^*$ disclosed by the knowledge extractor from $\sigma^*$ which can be used in solving the given SIS instance. We have the following two cases.

\(<A>\) Case I $|\tilde{id}^*| = \tilde{id}$ and $\tilde{id}^*$ does not belong to any user in $U^a$: The simulator $B$ sets the group public key $Y$ using $\tilde{id}$ and the given SIS instance $A = [A_1|A_2]$, $A_1, A_2 \in \mathbb{Z}_q^{n \times m}$, $m = \ell n \log q$ as follows.

- **Setup:** The simulator $B$ runs the algorithm $\text{TrapGen}(1^n, 1^{m}, q) \rightarrow (L, T_L)$ with $|T_L| \leq O(\sqrt{n \log q})$. It samples $(\mu + 2)$ Gaussian matrices $\{L_k\}_{k=0}^\mu, L_D \in \mathbb{Z}_q^{m \times m}$ with each matrix having its columns sampled independently from $D_{\mathbb{Z}_q, \sigma}$ where $\sigma$ is of size $\Omega(\sqrt{n \log q} \log n)$ and sets the matrices $\{A_k\}_{k=0}^\mu$, $D$ and $A$ as $A_0 = \tilde{A}_1 L_0 + \left( \sum_{k=1}^\mu \tilde{id}[k] \cdot L \right)$, $A_k = \tilde{A}_1 \cdot L_k + (-1)^{\tilde{id}[k]} \cdot L$ for $k \in [1, \mu]$, $D = \tilde{A}_1 \cdot L_D$, $A = \tilde{A}_1$, $B$ picks $h_u \leftarrow D_{\mathbb{Z}_q, \sigma}$ and sets the vector $u$ as $u = \tilde{A}_1, h_u \in \mathbb{Z}_q^n$. Rest all parameters are generated as in the real protocol.

- **Query Phase:** The simulator $B$ answers to the queries $Q_{pub}$, $Q_{read}$, $Q_{key/oa}$ and hash as in the original protocol but $Q_{u-join}$ is answered as follows:

$- Q_{u-join}$: Simulation of this query by $B$ is as follows:

- It sets the matrix $A_{id}$ as $A_{id} = [\tilde{A}_1(A_0 + \sum_{k=0}^\mu \tilde{id}[k] L_k) + h_{id}, L]$ where $h_{id} = \sum_{k=1}^\mu \tilde{id}[k] + (-1)\tilde{id}[k] \tilde{id}[k]$ is the Hamming distance between $\tilde{id}$ and $\tilde{id}$. The matrix $A_{id}$ has the same distribution as in the real protocol. The simulator $B$ finds the basis $T_{A_{id}}$ of $A_{id}^\perp (A_{id})$ using the knowledge of $T_L$ and the algorithm $\text{SampleRight}(\tilde{A}_1, h_{id}, L, (A_0 + \sum_{k=0}^\mu \tilde{id}[k] L_k), T_{h_{id}}, L, \sigma, 0) \rightarrow b \in \mathbb{Z}_q^{2m}$ [1]. Rest all are generated honestly and the certificate will be issued to the forger.

At the end, $A$ outputs the forgery $(M^*, \sigma^*, t^*)$ for a user $U \notin U^a$ where $\sigma^* = (\text{VK}^*, c_v, \Pi^*, \text{osig}^*, \Pi_R^*, [t_1^*, t_2^*])$ such that $\text{FSGS.Verify}(\sigma^*, t^*, M^*, Y) = 1$ with $t^* \in [t_1^*, t_2^*]$.

Observe that $B$ is able to extract witnesses $(d_1^*, d_2^*) \in \mathbb{Z}_q^{m \times m}, \tilde{id}^* \in \{0, 1\}^\mu$, $r^* \in \{0, 1\}^m$ from the proof of knowledge $\Pi^*$ using improved forking lemma [4] with $|d_1^*|_2 \leq \sigma \sqrt{m}$, $|d_2^*|_2 \leq \sigma \sqrt{m}$, $|r^*|_2 \leq \sigma \sqrt{m}$ satisfying $A_{id}^\perp(d_1^* d_2^*) = u + D \cdot r^* \mod q$ where $r^* = \text{bin}(D_0 \cdot \text{bin}(r^*) + D_1 \cdot s^*)$.

The simulator $B$ declares failure if either (i) $\tilde{id}^*$ is any user in $U^a$ or (ii) $\tilde{id}^* \neq \tilde{id}$. We denote the event by $\text{fail}$ when any one of the above circumstances occur. With a prediction that fail does not occur, $B$ can solve the given SIS instance by setting...
\[h = d_0^* + (L_0 + \sum_{k=1}^{\mu} \hat{id}[k]L_k) d_k^* - L_D \cdot r^* - h_u \in \mathbb{Z}^m.\]

Note that \(\hat{A}_i h = 0 \mod q\) and \(||h||_2 \leq 2\sqrt{\mu} + (\mu + 2)m^2\sigma^2\). Furthermore, \((h^t|0^m)^t\) is a short non-zero vector in \(\Lambda^+_q(\hat{A})\) and hence serves as a solution of the given SIS instance.

\(<B> \text{ Case II: if } \hat{id} = \hat{id} \text{ and } \hat{d}^* \text{ belongs to certain group member in } U^* \text{ but } t^* \notin [t_1, t_2]\) where [t_1, t_2] is the time period for which \(\hat{id}\) has been issued certificate

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

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

\[(4.1) \quad d_{\hat{id}_i} = d_0 + \sum_{k=1}^{\mu} \hat{id}[k]d_k = 0 \mod q \text{ iff } i = i^*, \text{ for } i \in \{1, 2, \ldots, \delta\}.
\]

This in turn implies that \(d_{\hat{id}} = d_{\hat{id}_*} = 0 \mod q\). Here \(\hat{id}_i = \hat{id} = [1]|\hat{id}[2], \ldots, \hat{id}_j|\mu]\) will be utilized to respond the \(Q_{a-join}\) query for the adversarially controlled user \(U_k \in \{1, 2, \ldots, \delta\}\).

Next \(B\) runs \(\text{TrapGen}(1^n, 1^m, q) \rightarrow (C, T_C), \text{TrapGen}(1^{2n}, 1^{2m}, q) \rightarrow (D_1, T_{D_1}), \text{TrapGen}(1^n, 1^m, q) \rightarrow (B, T_B)\). It samples matrices \(D_0 \leftarrow U(\mathbb{Z}^{2n \times 2m}_q), M_0, M_1 \leftarrow U(\mathbb{Z}^{m \times m}_q)\). It also selects Gaussian matrices \(S_0, S_1, \ldots, S_\mu \leftarrow \mathbb{Z}^{m \times m}\) whose columns are sampled from the distribution \(D_{Z_{2m}, \sigma}\) where \(\sigma\) is of size \(\Omega(\sqrt{n}\log q \log n)\), also selects \(S \leftarrow U(\mathbb{Z}^{m \times m})\) such that \(S^{-1}\) has low norm and sets the matrices \(D, A, A_0, \{A_k\}_{k=1}^\mu\) as \(D = \hat{A}_1, A = \hat{A}_1 \cdot S, A_0 = \hat{A}_1 \cdot S_0 + d_0 \cdot C, A_k = \hat{A}_1 \cdot S_k + d_k \cdot C\), for \(k = 1, 2, \ldots, \mu\), \(M_0 = (A_0 + \sum_{k=1}^\mu \hat{id}_k[k]A_k)S\) and \(M_1 = C\). The simulator \(B\) then finds a vector \(u \in \mathbb{Z}^m_q\) by computing \(A_{\hat{id}_*} = [A|A_0 + \sum_{k=1}^\mu \hat{id}_k[k]A_k]\), choosing two short vectors \(d_{r_*,1}, d_{r_*,2} \in D_{Z_{2m}, \sigma}\), sets \(d_r = [d_{r_*,1}|d_{r_*,2}]^t\), picks \(r_* = \bin(c_M) \in \{0, 1\}\) and thus it sets \(u = A_{\hat{id}_*} \cdot d_r - D \cdot \bin(c_M)\), \(A_{\hat{id}_*} \cdot d_r = u + D \cdot r_*\). The distribution of \(u\) is statistically close to \(U(\mathbb{Z}^n_q)\) since \(A\) is statistically uniform and \(d_{r_*,1}, d_{r_*,2} \leftarrow D_{Z_{2m}, \sigma}\). The remaining parameters \(F, H, H_0, H_2, \gamma, \text{OTS, DSig}\) are chosen as in the original protocol.

**Query Phase (when \((i \neq \hat{i})\):** The queries \(Q_{pub}, Q_{read}, Q_{keyOA}\), and hash queries made by \(\hat{A}\) are answered honestly by \(B\). The response to \(Q_{a-join}\) query is by setting \(A_{\hat{id}_*} = [A_1|A_1 \cdot S \{S^{-1} (S_0 + \sum_{k=0}^{\mu} \hat{id}_k[k]S_k)\} + d_{\hat{id}_*} - C]\). The matrix \(A_{\hat{id}_*}\) has the same distribution as in the real protocol.

**Query Phase (\(i = \hat{i}\):** Here \(\hat{id}_i = \hat{id}_* = \hat{id}\). The queries \(Q_{pub}, Q_{read}, Q_{keyOA}\), and hash queries made by \(\hat{A}\) are answered honestly by \(B\). The response to \(Q_{a-join}\) query is by setting \(A_{\hat{id}_*} = [A_1|A_1 \cdot S \{S^{-1} (S_0 + \sum_{k=0}^{\mu} \hat{id}_k[k]S_k)\}]\). As \(A_{\hat{id}_*}\) is independent of \(C\), the simulator \(B\) generates the certificate by making use of the vectors \(d_{r_*} = [d_{r_*,1}|d_{r_*,2}]\), where \(d_{r_*,1}, d_{r_*,2} \leftarrow D_{Z_{2m}, \sigma}, c_M \in \mathbb{Z}^{2n}_q\).

On receiving \(v^{(0)}_i\) from \(\hat{A}\), the simulator \(B\) computes \(c_r = c_M - D_0 \cdot \bin(v^{(0)}_i) \mod q\). It runs \(\text{SamplePre}(D_1, T_{D_1}, c_r, \sigma) \rightarrow s_r \in \Lambda^+_q(D_1)\) [7] using \(T_{D_1}\) to sample short vectors \(s_r \leftarrow D_{\Lambda^+_q(D_1), \sigma}\) satisfying \(D_1 s_r = c_r \mod q\). i.e., \(D_1 s_r = c_M - D_0 \cdot \bin(v^{(0)}_i) \mod q\). Note that in the Setup phase, we set \(r_* = \bin(c_M) = c_M - D_0 \cdot \bin(v^{(0)}_i) \mod q\). Note that in the Setup phase, we set \(r_* = \bin(c_M) =
Forgery: At the end, $A$ outputs a forgery $(M^*, \sigma^*, t^*)$ for a user $U \notin U^\alpha$ where $\sigma^* = (VK^*, c_{\star}, I^*, \text{osig}^*, C^*_{R^*}, [t_1^*, t_2^*])$ such that $\text{FSGS.\text{Verify}}(\sigma^*, t^*, M^*, Y) = 1$ 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 id$^* = \widehat{id}$ has been issued a certificate. The simulator $B$ can construct a knowledge extractor for the proof of knowledge $\Pi^*$ and extract witnesses $(d^*_1, d^*_2) \in \mathbb{Z}_m^m \times \mathbb{Z}_m^m, d^* \in \{0, 1\}^\mu, \ r^* \in \{0, 1\}^m$ from $\Pi^*$ with $||d^*_1||_2 \leq \sigma \sqrt{m}, ||d^*_2||_2 \leq \sigma \sqrt{m}, ||r^*||_2 \leq \sigma \sqrt{m}$ satisfying $A_{id^*} \cdot \frac{d^*_1}{|\pi|} = u + D \cdot r^* \mod q$.

The simulator $B$ declares failure in the following situations:

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

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

(iii) The knowledge extractor reveals vectors bin$(v^*) \in \{0, 1\}^{2m}$ and $s^* \in \mathbb{Z}_q^m$ with $r^* = r_{i^*}$, where bin$(v_{i^*})$ and $s_{i^*}$ are the vectors involved in the $t^*$-th $Q_{\text{join}}$ query i.e., id$^* = \widehat{id}$ and $r^* = r_{i^*}$.

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 id$^* = \widehat{id}$ belongs to user $i^* \in U^\alpha$, but $r^* \neq r_{i^*}$. $B$ can solve the given SIS instance by setting $h = S(d^*_1 - d_{i^*}), (S_0 + \sum_{k=1}^{\mu} \text{id}^kS_k)(d^*_2 - d_{i^*}) + (r_{i^*} - r^*)$ and by setup $A_{\widehat{i} \cdot} \cdot h = 0 \mod q$ with $||h|| \leq 2\sigma \sqrt{m} + 2\sigma \sqrt{m}$. Note that $r^* \neq r_{i^*}$, ensures that $h \neq 0$ with high probability. It gives, $(h'(0)^{\mu})^t$ is a short vector in $\Lambda^\perp_q(\widehat{A})$ and hence a solution of the given SIS instance.

Subcase II($r^* = r_{i^*}$ but (bin$(v^*), s^*) \neq (\text{bin}(v_{i^*}), s_{i^*})$):

Setup: The simulator $B$ generates the group public key $Y = (M_0, M_1, A, \{A_k\}_{k=0}^{\mu})$, $\mathbf{B, D, D_0, D_1, F, u, OTS, D\text{Sig}, H, H_0, H_2, \beta, \gamma, \sigma}$) faithfully as in the original protocol except the matrices $D_0, D_1 \leftarrow U(\mathbb{Z}_q^{2m \times 2m})$ which are set by $B$ as follows: It samples $\widehat{A}^* \leftarrow U(\mathbb{Z}_q^{n \times 2m})$, randomly selects $Q \leftarrow \mathbb{Z}_q^{2m \times 2m'}$ whose columns are sampled from $D_2 \in \mathbb{Z}_q^{2m \times 2m}$ and $D_1 \leftarrow D_0 \cdot Q \mod q$. The distribution of each of the matrices $D_0, D_1$ is statistically close to $U(\mathbb{Z}_q^{2m \times 2m})$. The simulator initializes the public state $St = (St_{\text{pub}}, St_{\text{trans}}) = (\emptyset, \epsilon)$.

Query Phase: In response to the $A$’s queries for $Q_{\text{pub}}, Q_{\text{read}}, Q_{\text{keyOA}}, Q_{\text{join}}$ and hash queries, the simulator $B$ answers as in the real protocol with the above setup.

Forgery: Let $(M^*, \sigma^*, t^*)$ be the forgery of $A$ for a user $U \notin U^\alpha$ where $\sigma^* = (VK^*, c_{\star}, I^*, \text{osig}^*, C^*_{R^*}, [t_1^*, t_2^*])$ such that $\text{FSGS.\text{Verify}}(\sigma^*, t^*, M^*, Y) = 1$ with $t^* \in [t_1, t_2]$. 


Using improved forking lemma [4], the simulator $B$ has access to a knowledge extractor. From the knowledge extractor, $B$ extracts $\bin(v^*) \in \{0,1\}^{2m}$, $s^* \in \mathbb{Z}_2^{2m}$, $r^* \in \{0,1\}^m$. The simulator $B$ declares failure if $r^* \neq r_\ast$.

We denote the event by $\text{fail}$ when the above situation occurs. When fail does not occur, $B$ solves the given $\text{SIS}$ instance by computing the vector $h = \bin(v^*) - \bin(v_{\ast}) + Q.(s^* - s_\ast) \in \mathbb{Z}_2^{2m}$. Observe that $h \in \Lambda_q^\perp(D_0)$ is a short vector and $D_0.h = 0 \mod q$. Further, $h \in \mathbb{Z}_2^{2m}$ is non-zero with high probability since $\bin(v^*) \neq \bin(v_{\ast})$. Note that, $||h||_2 \leq 2\sqrt{m} + 2\sqrt{m}^2\sigma^2$.

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

$\square$

**Theorem 4.2.** The group signature FSGS scheme described in Section 3 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 $A$ be any PPT adversary to our group signature scheme FSGS with non-negligible advantage $\epsilon$. We show that there exists a simulator $B$ that solves the SIS problem using the non-negligible advantage of $A$ i.e., given $\hat{A} = [A_1|A_2] \in \mathbb{Z}_q^{4n \times 4m}$, $\hat{A}_1, \hat{A}_2 \in \mathbb{Z}_q^{4n \times 2m}$, $\mu = 2n\log q$, a real number $\beta$, the simulator $B$ finds a vector $z \in \Lambda_q^\perp(\hat{A})$ such that $||z|| \leq \beta$ using $A$ as a subroutine.

- **Setup:** The simulator $B$ acts as the KGC and honest users while the adversary $A$ acts as the group manager. First, $B$ chooses $i^\ast \leftarrow U(\{1, 2, \ldots, q_0\})$ with a guess that the forged signature produced by $A$ reveals $\bin(v^*)$ coincides with $\bin(v_{i^*})$ of $i^\ast$-th user in $U^b$ and also chooses $\hat{t} \leftarrow U(\{1, 2, \ldots, T\})$ where $q_0$ is the cardinality of the set $U^b$ of honest users with $A$ as the dishonest group manager, $T = 2^l$ is allowable time periods and $N = 2^\mu$ is maximum number of group members with $\mu \geq \log l$. The simulator $B$ faithfully generates the group public key $\gamma = (M_0, M_1, A, \{A_k\}_{k=0}^l, B, D, D_0, D_1, F, u, OTS, DSig, H, H_0, H_2, \beta, \gamma, \sigma)$ as in the original protocol except the matrix $F \leftarrow U(\mathbb{Z}_q^{4n \times 2m})$ which is set by $B$ as $F = A \in \mathbb{Z}_q^{4n \times 2m}$. It also initializes the public state $St = (St_{\text{users}}, St_{\text{trans}}) = (\emptyset, \epsilon)$.

- **Query Phase:** The adversary $A$ queries the oracles $Q_{\text{pub}}, Q_{\text{KeyGM}}, Q_{\ast-\text{join}}, Q_{\text{sig}}, Q_{\text{corrupt}}, Q_{\text{write}}, Q_{\text{read}}$ and $Q_{\text{KeyOA}}$ which are simulated by $B$ honestly.

- **Forgery:** With access to all the above queries, $A$ outputs the forgery $(M^\ast, t^\ast, \sigma^*)$ for a user $U \in U^b$ where $\sigma^*$ is a forged signature on the message $M^\ast$ at time $t^\ast$. Parsing $\sigma^*$ as $\sigma^* = (V^*, c_{v^*}, \Pi^*, osig^*, C_{R^*}, [t^*_1, t^*_2])$ such that $\text{FSGS.Verify}(\sigma^*, t^*, M^\ast, \gamma) = 1$ with $t^* \in [t^*_1, t^*_2]$.

The simulator $B$ declares failure if $v^* \neq v_{i^*}$. The opening of $\sigma^*$ reveals $\bin(v^*) = \bin(v_{i^*}) \in \{0,1\}^{2m}$ and as $B$ has all the collection of short $z_i \in \mathbb{Z}_2^{2m}$ for each user $U_i \in U^b$ with $||z_i|| \leq 2\sqrt{\hat{m}}$, the simulator $B$ can find a $z_{i^*}$ such that $v_{i^*} = F \cdot z_{i^*} \mod q$.

Now, $B$ finds another short vector $\tilde{z}$ such that $v_{i^*} = F \cdot \tilde{z} \mod q$ by constructing a knowledge extractor for $\Pi^*$ by invoking improved forking lemma [4]. Thus $B$ has

$$F \cdot z_{i^*} = F \cdot \tilde{z} \mod q \Rightarrow F(z_{i^*} - \tilde{z}) = 0 \mod q \Rightarrow \hat{A}(z_{i^*} - \tilde{z}) = 0 \mod q \text{ (by Setup)} \Rightarrow z_{i^*} - \tilde{z} \in \Lambda_q^\perp(\hat{A})$$

with $||z_{i^*} - \tilde{z}|| \leq ||z_{i^*}|| + ||\tilde{z}|| \leq 2\sigma \sqrt{\hat{m}} + 2\sigma \sqrt{\hat{m}} = 4\sigma \sqrt{\hat{m}}$ is a short vector of $\Lambda_q^\perp(\hat{A})$ i.e., a short solution to the given SIS instance. $\square$
Theorem 4.3. The FSGS scheme described in Section 3 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 $E_i = Adv^{anon}(A)$ in Game $i$ for $i = 0, 1, \ldots, 6$. The game transition is described below.

- **Game 0**: Let $A$ be the adversary and $B$ be the simulator. This is the original game. The public parameters and query phase are respectively honestly generated and answered.

  **Play Phase**: With access to all the above queries, $A$ sends $(M^*, t^*, (\text{sec}_{i_1}, t^*\to t^*_1, \text{cert}_{i_1}, t^*\to t^*_1))$ with $b \in \{0, 1\}$ to $B$. The challenger $B$ aborts if either $([\text{cert}_{i_1}, t^*\to t^*_1]=y \text{ sec}_{i_1}, t^*\to t^*_1]$ or $[\text{cert}_{i_0}, t^*\to t^*_0 = \text{cert}_{i_1}, t^*\to t^*_1]$.

  **Challenge Phase**: In this phase, the challenger $B$ flips a coin and selects a bit $d \in \{0, 1\}$ to sign the message $M^*$ at time $t^*$. The challenger $B$ computes the challenge signature $\text{gsig}^* \leftarrow \text{FSGS.Sign}(\text{sec}_{i_0}, t^*\to t^*_0, \text{cert}_{i_0}, t^*\to t^*_0, Y, M^*)$ and sends it to $A$ where $\text{gsig}^* = (\text{VK}^*, c_{\text{v}}, \Pi^*, \text{osig}^*, C_{\text{R}}, [t_1, t_2])$ such that $\text{FSGS.Verify}(\sigma^*, t^*, M^*, Y) = 1$ with $t^* \in [t_1, t_2]$.

  **Guess Phase**: With access to the queries $Q_{\text{pub}}, Q_{\text{keyGM}}, Q_{\text{open}}[\langle M^*, \sigma^*, t^* \rangle], Q_{\text{read}}, Q_{\text{write}}$, the adversary $A$ 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 $B$ generates a OTS key pair $(\text{VK}^*, \text{SK}^*)$ before query phase starts. This is used by $B$ in the challenge query. The challenger $B$ aborts if following two cases occur:
  - The adversary $A$ asks $Q_{\text{open}}$ query to $B$ for the signature $\text{gsig} = (\text{VK}, c_{\text{v}}, \Pi, \text{osig}, C_{\text{R}}, [t_1, t_2])$ with $\text{VK} = \text{VK}^*$.
  - The adversary $A$ produces a signature $\text{gsig} = (\text{VK}, c_{\text{v}}, \Pi, \text{osig}, C_{\text{R}}, [t_1, t_2])$ with $\text{VK} = \text{VK}^*$ after knowing the challenge signature $\text{gsig}^*$.

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 $A$’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 $G_0^* \in \mathbb{Z}_q^{n \times 2m}$ uniformly and define $H_0(\text{VK}^*) = G_0^*$. Observe that the distribution of $G_0^*$ is still statistically close to that in the real game. To answer other queries for $\text{VK} \neq \text{VK}^*$, the challenger $B$ selects a small norm matrix $K_{\text{VK}} \leftarrow D_{\mathbb{Z}_q^{m \times n}}^{2m}$ and defines $H_0(\text{VK}) = B \cdot K_{\text{VK}}$. The matrix $K_{\text{VK}}$ is stored for further usage. If any query is repeated for $\text{VK}$, the stored value will be produced. Note that in the real protocol, $K_{\text{VK}}$ is generated using the trapdoor $T_B$. As SIS is hard, finding $K_{\text{VK}}$ is difficult without $T_B$ and hence difficult to decide whether $K_{\text{VK}}$ is generated using $T_B$ or uniformly chosen, for all fixed choices of $B$. Thus $H_0(\text{VK})$ is statistically close to uniform. Also distinguishing between the pair $(B, B \cdot K_{\text{VK}})$ generated using $T_B$, and the uniformly generated pair $(B, H_0(\text{VK}))$ 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, $B$ samples a uniformly random matrix $B^* \in \mathbb{Z}_q^{n \times m}$ to use in place of $B$ to respond to $H_0$-queries. Hence to answer $Q_{\text{open}}$ queries, $B$ does not use $T_B$ but recalls the stored $K_{\text{VK}}$ 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 [9]. Thus, $B$ simulates the zero knowledge argument of knowledge $\Pi = (\{\text{COM}_k\}_{k=1}^s, \text{Ch}, \{\text{RSP}_k\}_{k=1}^s)$ where $\text{Ch} = H(M, \text{VK}, c_v, \{\text{COM}_k\}_{k=1}^s) \in \{1, 2, 3\}^s$ 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 $\Pi$ 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 $c_{v^*_d}$ of the challenge phase. The challenger $B$ does not use the real GPV-IBE for encryption of $\text{bin}(v^*_d)$ but returns truly random ciphertext by setting $c_{v^*_d} = \left[ z_1, z_2 + \text{bin}(v^*_d)q/2 \right] \mod q$ where $\text{bin}(v^*_d) = F \cdot z^*_d$, and $z_1 \leftarrow U(\mathbb{Z}_q^m)$, $z_2 \leftarrow U(\mathbb{Z}_q^{2m})$ are randomly chosen. Note that in Game 0 to Game 5, $c_{v^*_d}$ is set as $c_{v^*_d} = \left[ G_0^* e_0 + e_1, B^* e_0 + e_1 \right]$ where $e_0 \leftarrow \chi^n$, $x_1 \leftarrow \chi^n$, $x_2 \leftarrow \chi^{2m}$ and $\chi$ is a $\gamma$ bounded distribution. If $A$ can distinguish between $B^* e_0 + x_1$ from $z_1$ and $G_0^* e_0 + x_2$ from $z_2$, which would break the decisional LWE assumption. Thus Game 4 and Game 5 are computationally indistinguishable under the security of GPV-IBE. Hence, $|E_4 - E_5| = \epsilon_5$ where $\epsilon_5 > 0$ is negligible.

• **Game 6**: This is the final game where $B$ uniformly selects $z'_1 \leftarrow U(\mathbb{Z}_q^m)$, $z'_2 \leftarrow U(\mathbb{Z}_q^{2m})$ and sets $c_{v^*_d} = \left[ z'_1, z'_2 \right]$ instead of using $\text{bin}(v^*_d)$. Observe that $c_{v^*_d}$ follows the same distribution as in Game 5. Also $c_{v^*_d}$ does not depend on $d \in \{0, 1\}$, giving zero advantage to the adversary $A$. Thus, $|E_5 - E_6| = |E_5| = \epsilon_6$ where $\epsilon_6 > 0$ is negligible. Thus the advantage of the adversary is $|\text{Adv}(A)| \leq |E_0 - E_1| + |E_1 - E_2| + |E_2 - E_3| + |E_3 - E_4| + |E_4 - E_5| + |E_5 - E_6| = \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + \epsilon_5 + \epsilon_6 < \epsilon$. Hence the result.

5. Conclusion

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

Acknowledgments

We would like to thank the anonymous reviewers for their valuable suggestions and comments, which considerably improved the quality of our paper.
References

[1] S. Agrawal, D. Boneh and X. Boyen, Efficient lattice (H)IBE in the standard model, in Advances in cryptology—EUROCRYPT 2010, Lecture Notes in Comput. Sci., Springer, Berlin, 6110 (2010), 533–572.

[2] M. Ajtai, Generating hard instances of lattice problems (extended abstract), in Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, ACM, New York, (1996), 99–108.

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

[4] E. Brickell, D. Pointcheval, S. Vaudenay and M. Yung, Design validations for discrete logarithm based signature schemes, in Public Key Cryptography, Lecture Notes in Comput. Sci., Springer, Berlin, 1751 (2000), 276–292.

[5] J. Camenisch, G. Neven and M. Rücker, Fully anonymous attribute tokens from lattices, in International Conference on Security and Cryptography for Networks, Springer, 2012, 57–75.

[6] D. Cash, D. Hofheinz, E. Kiltz and C. Peikert, Bonsai trees, or how to delegate a lattice basis, in Advances in cryptology—EUROCRYPT 2010, Lecture Notes in Comput. Sci., Springer, Berlin, 6110 (2010), 523–552.

[7] C. Gentry, C. Peikert and V. Vaikuntanathan, Trapdoors for hard lattices and new cryptographic constructions, in STOC’08, ACM, New York, (2008), 197–206.

[8] S. D. Gordon, J. Katz and V. Vaikuntanathan, A group signature scheme from lattice assumptions, in Advances in cryptology—ASIACRYPT 2010, Lecture Notes in Comput. Sci., Springer, Berlin, 6477 (2010), 395–412.

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

[10] F. Laguillaumie, A. Langlois, B. Libert and D. Stehlé, Lattice-based group signatures with logarithmic signature size, in Advances in cryptology—ASIACRYPT 2013. Part II, Lecture Notes in Comput. Sci., Springer, Heidelberg, 8270 (2013), 41–61.

[11] B. Libert, S. Ling, F. Mouhartem, K. Nguyen and H. X. Wang, Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions, in Advances in Cryptology—ASIACRYPT 2016. Part II, Lecture Notes in Comput. Sci., Springer, Berlin, 10032 (2016), 373–403.

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

[13] S. Ling, K. Nguyen, D. Stehlé and H. X. Wang, Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications, in Public Key Cryptography—PKC 2013, Lecture Notes in Comput. Sci., Springer, Heidelberg, 7778 (2013), 107–124.

[14] S. Ling, K. Nguyen and H. X. Wang, Group signatures from lattices: simpler, tighter, shorter, ring-based, in Public-key Cryptography—PKC 2015, Lecture Notes in Comput. Sci., Springer, Heidelberg, 9020 (2015), 427–449.

[15] S. Ling, K. Nguyen, H. X. Wang and Y. H. Xu, Forward-secure group signatures from lattices, International Conference on Post-Quantum Cryptography, 11505 (2019), 44–64, arXiv:1801.08323.

[16] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, Journal of the ACM (JACM), 56 (2009), Art. 34, 40 pp.

Received October 2018; revised March 2019.

E-mail address: kansal@maths.iitkgp.ernet.in
E-mail address: ratna@maths.iitkgp.ernet.in
E-mail address: sourav@maths.iitkgp.ernet.in