Efficient Dynamic Group Signature Scheme with Verifier Local Revocation and Time-Bound Keys using Lattices

Abhilash M H
Department of Computer Science & Engg.
National Institute of Technology
Warangal, India
Email: abhilashmh[AT]student.nitw.ac.in

B B Amberker
Department of Computer Science & Engg.
National Institute of Technology
Warangal, India
Email: bba[AT]nitw.ac.in

Abstract—**Revocation is an important feature of group signature schemes. *Verifier Local Revocation* (VLR) is a popular revocation mechanism that involves only verifiers in the revocation process. In VLR, a revocation list is maintained to store the information about revoked users. The verification cost of VLR based schemes is linearly proportional to the size of the revocation list. In many applications, the size of the revocation list grows with time, which makes the verification process expensive. In this paper, we propose a lattice based dynamic group signature using VLR and time bound keys to reduce the size of the revocation list to speed up the verification process. In the proposed scheme, an expiration date is fixed for signing key of each group member, and verifiers can find out (at constant cost) if a signature is generated using an expired key. Hence revocation information of members who are revoked before signing key expiry date (premature revocation) are kept in the revocation list, and other members are part of natural revocation. This leads to a significant saving on the revocation check by assuming natural revocation accounts for a large fraction of the total revocation. This scheme also takes care of non-forgeability of signing key expiry date.**

Keywords– Lattice based cryptography; dynamic group signatures; verifier local revocation; time bound keys

I. INTRODUCTION

The notion of group signatures was introduced by Chaum and Van Heyst [1]. In a group signature scheme, a set of potential signers (called members of group), each with their own secret key form a group. Each member in the group can anonymously issue a signature on behalf of the whole group i.e., the identity of the member is not revealed (anonymity). However, in cases of any disputes with a signature, the identity of the corresponding signer can be found using a tracing mechanism (traceability). Group membership is handled by a designated authority called group manager. Tracing of the signature is handled by an independent opening authority or the group manager itself.

The first rigorous security model for a static setting was given by Bellare, Micciancio, and Warinschi in [2]. In the static model, all the potential group members are fixed in the setup phase itself. Subsequently, Bellare, Shi, and Zhang [3] and Kiayias, Agelos and Yung [4] proposed a partially dynamic model, where users can join the group at any time (member registration), but once they have done so, they cannot leave the group. Also, there are schemes in which members can only be revoked at any time, but cannot join after the group setup phase. Even this model is considered partially dynamic. Whereas in dynamic model members can be registered and revoked from the group at any time.

So an important functionality of dynamic group signatures is *revocation*, which supports removing members from the group anytime. The most commonly used revocation mechanism is *Verifier Local Revocation* (VLR) proposed by Boneh and Shacham [5]. In the VLR scheme, a signer is not part of the revocation process, only a verifier is involved in the revocation process. A revocation list RL is maintained by the group manager which contains information about the revoked members. The group manager keeps a revocation token for each member of the group. Whenever a member is revoked, his revocation token is kept in RL. To check if a signer of signature is revoked or not, the verifier runs the revocation check procedure for all the revocation tokens in RL. Hence, the signature verification algorithm consists of two steps

- Validation check: To check whether the signature is generated by a member of the group or not.
- Revocation check: To check that the signed member is revoked or not (by using RL).

Group signature schemes with VLR are used in many real-life applications like online medical services [6], anonymous authentication in wireless sensor networks [7], anonymous online communication [8], remote biometric authentication [9], privacy preserving inter vehicle communications in vehicular ad hoc network (VANET) [10]. In the above applications, members will be part of the group for a certain period, after that they will be revoked. In such scenarios, RL size grows significantly with the passage of time. It is clearly observed that the computation cost of the revocation check is linearly proportional to the size of RL. This motivates us to reduce the size of RL, which in turn reduces the verification cost.

Chu, Liu, Huang, and Zhou [11] proposed a VLR based group signature scheme (using bilinear- pairing) to reduce the verification cost. They used the concept of time bound keys, where an expiration date τ is set to each member signing key. Along with proving membership of the group, a signer has to prove that his expiration date is not passed i.e., $t_c < \tau$, where t_c is the present date. Hence a member whose expiration date τ has passed cannot generate a signature that is successfully validated by the verification algorithm. This type of revocation is called "natural" revocation i.e, members whose signing key has expired. In case a signer needs to be revoked before the expiration date τ, his revocation tokens are kept in RL. This type of revocation is called "premature" revocation. Now, a verifier needs to run the revocation check only for the prematurely revoked members.

All the schemes discussed above becomes insecure once quantum computers are realized. It is conjectured that cryptographic schemes based on lattices are resistant against quantum computers. Unlike classical cryptography (based on hardness of discrete log problem or factoring problem), lattice based cryptography enjoys provable security based on worst-case hardness assumptions [12] [13] [14]. This necessitates the construction of secure and efficient cryptographic schemes using lattices.

Lattice based Group Signatures. In 2010, Gordon, Katz, and Vaikuntanathan [15] constructed the first lattice based group signature where both the signature size and group public key size are linear in number of group members. Camenisch, Jan and Neven [16] improved the work of [15] to achieve stronger anonymity using attribute token system. In [17] an efficient scheme was constructed where the sizes of both group public key and signature were proportional to $\log N$. However, all the above are static schemes, which do not support member registration or revocation. In 2014, [18] constructed the first group signature scheme with revocation using VLR mechanism, which achieves almost the same asymptotical efficiency as [17]. But this scheme does not support member registration. In [19] a static group signature scheme was constructed, which is efficient by a $\log N$ factor in group public key and signature size compared to previous schemes [18][17]. The work of [19] was improved in [20] [21] by including VLR based revocation to their construction. Still, this scheme is partially dynamic which does not support member registration. In [22] first VLR based dynamic group signature scheme was constructed, but the sizes of both public key and secret key are similar to [18], which is inefficient compared to [19] [20]. In [23] a signature scheme that facilitates only member registration was constructed but it does not support revocation.
In the above discussed VLR based schemes, verification cost is proportional to the size of RL. This leads to the following open questions: *i)* Is it possible to reduce the size of RL? *ii)* Is it possible to construct a lattice based dynamic group signature scheme using VLR and reduced key size?

A. Contribution

In this work, we address all the above issues by constructing a new lattice based dynamic group signature scheme using VLR. We use the concept of time bound keys similar to [11], to reduce the size of RL. Compared to the previous lattice based VLR group signature schemes, our scheme has the following advantages.

- Our scheme significantly reduces the size of the revocation list because of natural revocation.
- Vertification cost of our scheme is less because of the reduced size of the revocation list.
- The size of the public key and the secret key is efficient by $\log N$ compared to the schemes in [18] and [22].
- Unlike [11], our scheme takes care of non-forgeability of secret key expiration date τ i.e, no member with τ as signing key expiration date can generate a valid signature after τ has passed.

A detailed comparison our scheme and with the existing lattice based VLR group signature schemes is shown in the Table 1. Here, λ is the security parameter, $n = \mathcal{O}(\lambda)$, N is the maximum number of members in the group, m and q are the parameters polynomial in n and $l = \log N$. $|R_{pre}|$ denotes the number of prematurely revoked members, and $|RL|$ denotes the number of revoked members or size of the revocation list. We know, $|R_{pre}| << |RL|$ in many scenarios.

Table 1: **Comparision of Lattice based VLR group signature schemes**

Scheme	**[18]**	**[20]**	**[21]**	**[22]**	**Our Scheme**
gpk size	$O(nml \log q)$	$O(nm \log q)$	$O(nm \log q)$	$O(nml \log q)$	$O(nm \log q)$
sk size	$O(ml \log q)$	$O(m \log q)$	$O(m \log q)$	$O(ml \log q)$	$O(m \log q)$
Revocation List size	$\|RL\|$	$\|RL\|$	$\|RL\|$	$\|RL\|$	$\|R_{pre}\|$
Verification cost	$\|RL\|$	$\|RL\|$	$\|RL\|$	$\|RL\|$	$\|R_{pre}\|$
Partial Dynamic/ Dynamic	Partial Dynamic	Partial Dynamic	Partial Dynamic	Dynamic	Dynamic

II. PRELIMINARIES

A. Notations

All matrices are denoted by bold upper-case letters such as $\boldsymbol{A}$, and vectors are denoted by bold lower-case letters, such as $\boldsymbol{u}$. All the vectors are assumed to be in column form. For $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ and $\boldsymbol{B} \in \mathbb{Z}^{p\times m}$, row concatenation is denoted by $[\boldsymbol{A}||\boldsymbol{B}] \in \mathbb{Z}^{(n+p)\times m}$. For $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ and $\boldsymbol{B} \in \mathbb{Z}^{n\times p}$, column concatenation is denoted by $[\boldsymbol{A}|\boldsymbol{B}] \in \mathbb{Z}^{n\times(m+p)}$. $\boldsymbol{A}^{\mathrm{T}}$ indicates the transpose of matrix $\boldsymbol{A}$, and $\boldsymbol{A}^{-1}$ indicates the inverse of matrix $\boldsymbol{A}$. Denote the l_2 and l_∞ norm by $\|\cdot\|$ and $\|\cdot\|_\infty$, respectively. The norm of a matrix $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ with columns $(a_i)_{i=1}^{m}$ is defined as the norm of its longest column (i.e., $\|\boldsymbol{A}\| = max_i\|\boldsymbol{a}_i\|$). If the columns of $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m)$ are linearly independent, let $\widetilde{\boldsymbol{A}} = (\widetilde{\boldsymbol{a}_1}, \widetilde{\boldsymbol{a}_2}, \ldots, \widetilde{\boldsymbol{a}_m})$ denote the Gram-Schmidt Orthogonalization of vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m$ in the same order. For a finite set S, sampling a value x accoding to distribution D is denoted by $x \hookleftarrow D_S$. The set of integers $\{1, 2, \ldots, n\}$ is denoted by [n], for any integer $n \geq 1$.

B. VLR Group Signature Using Time Bound Keys

The construction of our scheme is similar to [23], and the concept of time bound keys from [11] is used in our scheme. A trusted third party runs a Setup algorithm to fix public parameters and group manager secret key. A user who wants to be a potential group member involves in an interactive join protocol with the group manager. After successful join, the member gets a secret key (or signing key) with an expiration date τ, a group membership certificate, and a revocation token. The member can generate signatures on messages as long as τ is not expired. It is necessary to hide τ from verifiers to maintain the privacy of the group members. To achieve this, the member selects an intermediate signature expiration date t (public) for each signature and proves in zero-knowledge that $t < \tau$. During the verification process, the verifier checks if the current date t_c is no later than the signature expiration date t. If that is true following condition holds $t_c \leq t < \tau$. Thus, the verifier is sure that the signature is generated by a non-expired signing key. However, a verifier can still check if signature is generated by an old (secret key expired) group member even after the expiration date τ.

Different date formats such as "YYYYMMDD" or "YYMMDD" or "YYMM" in integer can be used to represent a date. If we take "YYMMDD" date format, then $t = 210101$ means 01 January 2021, $\tau = 210311$ means 11 March 2021, it can be easily seen that $t < \tau$.

The definition of VLR group signature with time bound keys is taken from [11]. It consists of the following algorithms.

- **Setup**(λ, N, d)**:** On input security parameter $\lambda \in \mathbb{N}$, the maximum number of members in a group N and a date format d this algorithm and outputs the group public-key (gpk) and a group manager secret key ($gmsk$). Group manager receives the $gmsk$.

- **Join:** It is an interactive protocol between user i (denoted U_i) and group manager(denoted GM) . U_i takes as input gpk, a digital signature public key, secret key pair $(upk[i], usk[i])$. GM takes $gpk, gmsk, \tau_i$ (secret key expiration date for user U_i). At the end of the protocol U_i fixes its secret (signing) key sec_i, and obtains a membership certificate $cert_i$, and a revocation token grt_i from the GM. Transcripts of this communication are stored in *transcripts* database by GM. It is a private database containing transcripts of communication between GM and U_i of all the group members. Each transcript in *transcripts* also contains coin tosses used by the GM during execution of join algorithm.

- **Sign**$(gpk, \boldsymbol{m}, sec_i, cert_i, t)$**:** On input gpk, a message $\boldsymbol{m} \in \{0,1\}^*$, secret-key sec_i, membership certificate $cert_i$, signature expiration date t this algorithm generates the signature Σ on $\boldsymbol{m}$.

- **Verify**$(gpk, \boldsymbol{m}, \Sigma, t_c, RL)$**:** On input gpk, a message $\boldsymbol{m} \in \{0,1\}^*$, signature Σ, current date t_c and revocation list RL, this algorithm returns 1 if Σ is valid signature on $\boldsymbol{m}$, else return 0.

- **Open**$(gpk, \boldsymbol{m}, \Sigma, RL)$**:** On input a message $\boldsymbol{m} \in \{0,1\}^*$, a valid signature Σ on $\boldsymbol{m}$, RL, this algorithm outputs an index $i \in [N]$ or a special symbol $\perp$ in case of opening failure.

C. Correctness

Correctness guarantees that for a group signature Σ generated with signature expiration time t, by a non revoked signer i, whose secret key expiration date is τ_i and $t < \tau_i$. Σ is valid and the opening correctly identifies the signer i.

Definition 1. $\forall\ d, t, t_c, \tau_i, RL, \boldsymbol{m} \in \{0,1\}^*$, $(gpk, gmsk) \leftarrow Setup(\lambda, N, d)$, $(sec_i, cert_i, grt_i, \tau_i) \leftarrow Join$, *we have* $Verify(gpk, \boldsymbol{m}, \Sigma, t_c, RL) = 1 \iff grt_i \notin RL$ *and* $t_c \leq t < \tau_i$

D. Security Definitions

For security model, we follow the definitions of [24] instead of [11]. Because security definitions of [24] are tighter and they consider non-forgeability of expiration date τ for traceability.

A group signature scheme is secure if it is secure against traceability, selfless-anonymity and framing attacks. In all of these attacks, an adversary $\mathcal{A}$ is given access to oracles which share few variables. Both shared variables and oracles are explained below.

Variables:

- H_u: Set of honest members in the group
- C_u: Set of corrupted members in the group
- $Sigs$: All the signing queries made by $\mathcal{A}$ are stored in this set.

Oracles:

- $\boldsymbol{Add_{user}}$: This oracle allows $\mathcal{A}$ to add honest user to the group. On input (i, τ_i), this oracle computes $sec_i, cert_i, grt_i$ by running Join algorithm locally and i is added to H_u.

- $\boldsymbol{Q_{gm}}$: This oracle allows $\mathcal{A}$ to act as a corrupt user i, and engage in join protocol. After successful join $\mathcal{A}$ gets $sec_i, \tau_i, cert_i, grt_i$ and i is added to C_u.

- $\boldsymbol{Q_{user}}$: This oracle allows $\mathcal{A}$ to act as a corrupted group manager and engage in join protocol with a hones user i. After successful join i is added to H_u.

- $\boldsymbol{Get_{reg}}$: On input index i, this oracle returns $\perp$ if i does not exist. Otherwise it returns (grt_i, τ_i)

- $\boldsymbol{Revoke}$: This oracles allow $\mathcal{A}$ to revoke honest users. On input index i, this oracle adds grt_i to revocation list RL.

- $\boldsymbol{G_{sign}}$: On input $i, \boldsymbol{m},$ and t signing oracle returns the signature Σ on $\boldsymbol{m}$ with signature expiration date t. Further $(\boldsymbol{m}, \Sigma, t)$ is added to $Sigs$.

- $\boldsymbol{Get_{gmsk}}$: This oracle returns the $gmsk$ to $\mathcal{A}$.

- $\boldsymbol{Get_{usk}}$: On input index i, it returns $\perp$ if i does not exist. Otherwise it returns $(sec_i, cert_i)$ and adds i to C_u.

1) Traceability

In this attack, an adversary $\mathcal{A}$ is allowed to control a set of users in the group through Q_{gm} queries. Adversary is also allowed to observe the system during addition of users and signature generation through $Add_{user}, G_{sign}, Get_{reg}, Revoke$ queries. Finally, $\mathcal{A}$ has to produce the signature that is not opened to a user controlled by the $\mathcal{A}$ or forge the secret key expiration date. It is clearly explained in the below experiment.

Definition 2. *For security parameter λ and any PPT adversary $\mathcal{A}$, traceability experiment $Exp_{\mathcal{A}}^{trace}(\lambda)$ is defined as follows.*

```
$Exp_{\mathcal{A}}^{trace}(\lambda)$ :
  $(gpk, gmsk) \leftarrow Setup(\lambda, N, d)$;  $C_u := \emptyset$;  $Sigs := \emptyset$;
  $(m, \Sigma, t, RL) \leftarrow \mathcal{A}(Q_{gm}, Add_{user}, Get_{reg}, Revoke, G_{sign})$
  Return 1 if conditions $((1) \wedge (2)) \vee ((1) \wedge (3))$ are satisfied.
      (1)Verify$(gpk, m, \Sigma, t_c, RL) = 1 \wedge (m, \Sigma, t) \notin Sigs$
      (2)$i \leftarrow$ Open$(gpk, m, \Sigma, RL) \wedge (i = \perp \vee i \notin C_u \backslash RL)$
      (3)$\tau_i < t$
  Otherwise return 0;
```

We say that our scheme is traceable if the advantage

$Adv_{\mathcal{A}}^{trace}(\lambda) := |Pr[Exp_{\mathcal{A}}^{trace}(1^{\lambda})] = 1|$ is negligible for $\mathcal{A}$.

2) Non-frameability

In this attack, $\mathcal{A}$ can corrupt the group manager through Get_{gmsk} query and add a set of honest users to the group through Q_{user} queries. $\mathcal{A}$ is also allowed to make signing queries through G_{sign}, get secret key of honest users via Get_{usk} query, and get registration details via Get_{reg} query. Finally, adversary has to produce a signature that is not queried earlier using G_{sign} query and opened to a honest non-revoked user. It is clearly explained in the below experiment.

Definition 3. *For security parameter λ and any PPT adversary $\mathcal{A}$, non-frameability experiment $Exp_{\mathcal{A}}^{non-frame}(\lambda)$ is defined as follows.*

```
$Exp_{\mathcal{A}}^{non-frame}(\lambda)$ :
  $(gpk, gmsk) \leftarrow Setup(\lambda, N, d)$;
  $H_u := \emptyset$;  $C_u := \emptyset$;  $Sigs := \emptyset$;
   $(m, \Sigma, t, RL, i) \leftarrow \mathcal{A}(Q_{user}, Get_{gmsk}, Get_{usk}, Get_{reg},$
                                          $G_{sign})$ ;
  Return 1 if conditions $(1) \wedge (2) \wedge (3)$ are satisfied.
      (1)Verify$(gpk, m, \Sigma, gpk, t_c, RL) = 1$
      (2)$i \leftarrow$ Open$(gpk, m, \Sigma, RL) \wedge (i = \perp \vee i \notin C_u \backslash RL)$
      (3)$(i \in H_u) \wedge (i \notin C_u \backslash RL) \wedge ((m, \Sigma, t) \notin Sigs)$
  Otherwise return 0;
```

We say that our scheme is non-frameable if the advantage

$$Adv_{\mathcal{A}}^{non-fra}(\lambda) := |Pr[Exp_{\mathcal{A}}^{non-fra}(\lambda)] = 1|$$

is negligible for $\mathcal{A}$.

3) Selfless-Anonymity

In this attack, $\mathcal{A}$ operates in two stages: *play* and *guess*. In the *play* stage, $\mathcal{A}$ is allowed to introduce honest users using Add_{user} query, get the secret key of honest users via Get_{usk} query, make signing queries through $Gsign$, revoke users through *Revoke* queries. At the end of the first stage, $\mathcal{A}$ returns a challenge message and two non-revoked identities $(i_0, i_1) \in H_u$. $\mathcal{A}$ obtains a challenge signature which was signed by one of the users, i_0 or i_1, on the challenge message. In the *guess* stage, the adversary tries to guess the signer of the challenge signature. It is clearly explained in the below experiment.

Definition 4. *For security parameter λ and any PPT adversary $\mathcal{A}$, selfless-anonymity experiment $Exp_{\mathcal{A}}^{anon\text{-}b}(\lambda)$ is defined as follows.*

```
 $Exp_{\mathcal{A}}^{anon\text{-}b}(\lambda)$
 $(gpk, gmsk) \leftarrow Setup(\lambda, N, d)$;
 $(aux, m, i_0, i_1) \leftarrow \mathcal{A}(play : Add_{user}, Get_{usk}, G_{sign},$
                              $Revoke)$;
 If $i_0 \in C_u$ or $i_1 \in C_u$, then return 0;
 $b \xleftarrow{\$} \{0,1\}$,  $\sigma \leftarrow Sign(gpk, m, sec_{i_b}, cert_{i_b}, t)$;
 $b' \leftarrow \mathcal{A}(guess, aux, \sigma; Q_{gm}, G_{sign}, Get_{usk}, Get_{reg})$;
 If $b = b'$, then return 1;
 return 0;
```

We say that our scheme is selfless-anonymous if the advantage

$Adv_{\mathcal{A}}^{anon\text{-}b}(\lambda) := |Pr[Exp_{\mathcal{A}}^{anon\text{-}b}(\lambda)] - \frac{1}{2}|$ is negligible for $\mathcal{A}$.

E. Lattices and Discrete Gaussian

Let $(\boldsymbol{b}_i)_{i \leq n}$ be a set of linearly independent basis vectors belonging to $\mathbb{R}^n$, the lattice Λ is the set of all integer linear combination of basis vector i.e.,

$$\Lambda(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n) = \{x_i \boldsymbol{b}_i : x_i \in \mathbb{Z}\}$$

We work on $q - ary$ lattices, which are defined below.

Definition 5. *For a prime $q \geq 2$, matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ where $m \geq n \geq 1$, define following three lattice,*

$$\Lambda_q(\boldsymbol{A}) = \{\boldsymbol{e} \in \mathbb{Z}^m : \exists \boldsymbol{x} \in \mathbb{Z}_q^n \text{ s.t. } \boldsymbol{A}^T \boldsymbol{x} = e \mod q\}$$

$$\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \mod q\}$$

For any $\boldsymbol{u} \in \mathbb{Z}_q^n$,

$$\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{Z}^m : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{u} \mod q\}$$

For a lattice Λ, a vector $\boldsymbol{c} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}^+$ define the function, $\rho_{\sigma,c}(\boldsymbol{x}) = exp(-\pi \frac{||\boldsymbol{x}-\boldsymbol{c}||^2}{\sigma^2})$. The *discrete Gaussian distribution* over the lattice Λ with centre c and parameter σ is defined as, $D_{\Lambda,\sigma,c}(\boldsymbol{x}) = \frac{\rho_{\sigma,c}(\boldsymbol{x})}{\rho_{\sigma,c}(\Lambda)}$, where $\boldsymbol{x} \in \Lambda$ and $\rho_{\sigma,c}(\Lambda) = \sum_{\boldsymbol{x} \in \Lambda} \rho_{\sigma,c}(\boldsymbol{x})$. We use $D_{\Lambda,\sigma}(\boldsymbol{x})$ to indicate the distribution centered in c=0.
In the following lemma, we review several well-known facts about discrete Gaussian distribution:

Lemma 1 ([25], Lemma 2.11, [13], Lemma 2.10). *Let n and $q \geq 2$ be integers, $m > 2n \log q$ and $\sigma > \omega(\sqrt{\log m})$.*

1. *For all but a $2q^{-n}$ fraction of all $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, for $\boldsymbol{x} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$, the distriution of $\boldsymbol{u} = \boldsymbol{Ax}$ is statistically close to unifrom over $\mathbb{Z}_q^n$. Moreover, the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{u}$ is $D_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma}$.*

2. *For $\beta = \lceil \sigma \log m \rceil$, and $\boldsymbol{x} \hookleftarrow D_{\mathbb{Z}^m,\sigma}$, $Pr[\|\boldsymbol{x}\|_\infty > \beta]$ is negligible.*

3. *The min-entropy of $D_{\mathbb{Z}^m,\sigma}$ is at least $m-1$.*

From the above lemma, it is clear that samples obtained from discrete Gaussian distribution over a lattice Λ are short with overwhelming probability. This fact is used extensively in our construction.

Lemma 2 ([26], Theorem 3.2)**.** *GenTrap is a PPT algorithm on input integers $m \geq n \geq 1$ and $q \geq 2$ such that $m \geq \Omega(n \log q)$ outputs a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and a basis $\boldsymbol{T_A}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$. The distribution of $\boldsymbol{A}$ is within statistical distance $2^{-\Omega(n)}$ to uniform distribution from $\mathbb{Z}_q^{n\times m}$ and $\|\tilde{\boldsymbol{T}}_{\boldsymbol{A}}\| \leq \mathcal{O}(\sqrt{n \log q})$*

Using *GenTrap* algorithm one can obtain a trapdoor (short basis) for lattice generated by a matrix.

Lemma 3 ([27], Lemma 3)**.** *ExtRandBasis is a PPT algorithm on input a matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m'}$, $\boldsymbol{A}_1 = [\boldsymbol{A}|\boldsymbol{B}] \in \mathbb{Z}_q^{n\times(m+m')}$ whose first m columns span $\mathbb{Z}_q^n$, a basis $\boldsymbol{T_A} \in \mathbb{Z}^{m\times m}$ of a lattice $\Lambda_q^{\perp}(\boldsymbol{A})$, a real $\sigma \geq \|\tilde{\boldsymbol{T}}_{\boldsymbol{A}}\|\omega(\sqrt{\log m})$ outputs a random basis $\boldsymbol{T}_{\boldsymbol{A}_1}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_1)$ satisfying $\|\tilde{\boldsymbol{T}}_{\boldsymbol{A}_1}\| \leq \|\tilde{\boldsymbol{T}}_{\boldsymbol{A}}\|$*

Using *ExtRandBasis* algorithm, one can obtain a trapdoor for any $\boldsymbol{A}_1$ whose left $n \times m$ submatrix is $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, with a known trapdoor $\boldsymbol{T_A}$.

Lemma 4 ([13], Theorem 4.1)**.** *SampleD is a PPT algorithm that takes a basis $\boldsymbol{T_A} \in \mathbb{Z}^{m\times m}$ of a lattice Λ, a real $\sigma \geq \|\tilde{\boldsymbol{T}}_{\boldsymbol{A}}\|\omega(\sqrt{\log m})$ and a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$ outputs a vector $\boldsymbol{x} \in \Lambda$ sampled according to $D_{\mathbb{Z}^m,\sigma}$ such that $\boldsymbol{Ax} = \boldsymbol{u}$*

Using *SampleD* algorithm, one can obtain a lattice vector sampled according to the discrete Gaussian distribution using trapdoor of the lattice.

Lemma 5 ([17], Lemma 5, [19] Proposition 6)**.** *SuperSamp is a PPT algorithm on input integers $m \geq n \geq 1$ and $q \geq 2$ such that $m \geq \Omega(n \log q)$, as well as $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{C} \in \mathbb{Z}_q^{n\times n}$ outputs an almost uniform matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m}$ such that $\boldsymbol{AB}^T = \boldsymbol{C}$ and a basis $\boldsymbol{T_B}$ of $\Lambda_q^{\perp}(\boldsymbol{B})$ satisfying $\|\boldsymbol{T_B}\| \leq m^{1.5}\omega(\sqrt{\log m})$ and $\|\tilde{\boldsymbol{T}}_{\boldsymbol{B}}\| \leq m\omega(\sqrt{\log m})$*

F. Lattice-Related Computational Problems

The security of our scheme relies on the hardness of two lattice problems: Learning With Errors (LWE) and Short Integer Solutions (SIS). A variant of SIS is called Inhomogeneous Short Integer Solutions ($ISIS$).

Definition 6. *Learning With Errors($LWE_{n,q,\chi}$) [28] Let χ be a distribution over $\mathbb{Z}$ and let $n, m \geq 1, q \geq 2$. For a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, $A_{\boldsymbol{s},\chi}$ is a distribution of $(\boldsymbol{a}, \boldsymbol{a}^T\boldsymbol{s}+e)$ over $(\mathbb{Z}_q^n, \mathbb{Z}_q)$, where $\boldsymbol{a} \hookleftarrow \mathbb{Z}_q^n, e \hookleftarrow \chi$. The $LWE_{n,q,\chi}$ problem asks to distinguish m samples chosen according to $A_{\boldsymbol{s},\chi}$ ($\boldsymbol{s}$ chosen uniformly) and m samples chosen according to uniform distribution over $(\mathbb{Z}_q^n, \mathbb{Z}_q)$.*

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

Definition 7. *Short Integer Solutions($SIS_{n,m,q,\beta}$) [29, 28] Let m, β, q be parameters polynomial in n. Given m uniformly random vectors $\boldsymbol{a}_i \in \mathbb{Z}_q^n$, forming the columns of a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, find a nonzero vector $\boldsymbol{x} \in \mathbb{Z}^m$ such that $\|\boldsymbol{x}\|_\infty \leq \beta$ and $\boldsymbol{Ax} = 0$.*

For any $m, \beta = poly(n)$ and for any $q \geq \sqrt{n}\beta$, solving $SIS_{n,m,q,\beta}$ problem with non-negligible probability is at least as hard as solving $SIVP_\gamma$ problem, for some $\gamma = \mathcal{O}(\beta\sqrt{n})$[13].

G. **0/1-**Encoding

To hide member secret key expiration date τ, 0/1-Encoding technique is used. This encoding technique reduces the *greater than* predicate to the *set intersection* predicate [30]. Let $x = \{x_k x_{k-1} \ldots x_0\} \in \{0,1\}^k$ be a k bit binary string. The 0-*Encoding* of x denoted 0-*Enc*(x) is the set of binary strings defined as below,

$$0\text{-}Enc(x) = \{x_k x_{k-1} \ldots x_{i+1}1 | x_i = 0, 1 \leq i \leq k\}$$

The 1-*Encoding* of x, denoted 1-*Enc*(x) is the set of binary strings defined as below,

$$1\text{-}Enc(x) = \{x_k x_{k-1} \ldots x_i | x_i = 1, 1 \leq i \leq k\}$$

In 0-*Enc*(x)/1-*Enc*(x) bit length of each binary strings is different and it varies from 1 to k. Also both the encoding have at most k elements.
If we encode x into 1-*Enc*(x) and y into 0-*Enc*(y), we can see that

$$x > y \iff 1\text{-}Enc(x) \text{ and } 0\text{-}Enc(y) \text{ has a common element}$$

Lets take an example. Let $x = 22 = 10110_2$ and $y = 17 = 10001_2$ of bit length 5(if needed we fill leading zero's to make both binary strings of same length), then 1-*Enc*$(x) = \{1011, 101, 1\}$ and 0-*Enc*$(y) = \{11, 101, 1001\}$. Since 1-*Enc*$(x) \cap$ 0-*Enc*$(y) = \{101\}$ we can say $x > y$. If $x = 17 = 10001_2$ and $y = 22 = 10110_2$, then 1-*Enc*$(x) = \{10001, 1\}$ and 0-*Enc*$(y) = \{11, 10111\}$. Since 1-*Enc*$(x) \cap$ 0-*Enc*$(y) = \emptyset$ we can say $x \leq y$.

Theorem 1 ([30], Theorem 1)**.** *x is greater than y if and only if* 1-*Enc*(x) *and* 0-*Enc*(y) *have a common element.*

III. PROPOSED SCHEME

Security parameter is denoted by $\lambda > 0$ everywhere and the maximum number of members in a group as $N = 2^l \in poly(\lambda)$. Then choose lattice parameter $n = \mathcal{O}(\lambda)$, prime modulus $q = \mathcal{O}(ln^3)$, dimension $m \geq \Omega(n \log q)$, Gaussian parameter $\sigma = \Omega(\sqrt{n \log q} \log n)$, infinity norm bound $\beta = \sigma\omega(\log m)$, $\acute{m} = 2n\lceil \log q \rceil$. Following are the random oracles used in our scheme, $\mathrm{H} : \{0,1\}^* \to \{0,1,2\}^t$, $\mathrm{H}_1 : \{0,1\}^* \to \{0,1\}^{2m}$, $\mathrm{H}_2 : \{0,1\}^* \to \mathbb{Z}_q^n$, and $\mathrm{H}_3 : \{0,1\}^* \to \mathbb{Z}_q^{m\times n}$.

A. Setup(λ, N, d)

- Generate three instances of hard random lattices $(\boldsymbol{A}, \boldsymbol{T_A})$, $(\boldsymbol{C}, \boldsymbol{T_C})$, and $(\boldsymbol{P'}, \boldsymbol{T_{P'}})$ using $GenTrap(n,m,q)$ algorithm. Sample $\boldsymbol{P}$ uniformly over $\mathbb{Z}_q^{n\times n}$ and run $Supersamp(n,m,q,\boldsymbol{C},\boldsymbol{P})$ to obtain $(\boldsymbol{B}, \boldsymbol{T_B})$ such that $\boldsymbol{C}\boldsymbol{B}^{\mathrm{T}} = \boldsymbol{P} \mod q$.
- Sample the following matrices uniformly, $\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3$ over $\mathbb{Z}_q^{n\times m}$, $\boldsymbol{D}$ over $\mathbb{Z}_q^{n\times \acute{m}}$, $\boldsymbol{D}_0$ over $\mathbb{Z}_q^{2n\times 2\acute{m}}$, $\boldsymbol{D}_1$ over $\mathbb{Z}_q^{2n\times 2m}$, $\boldsymbol{F}$ over $\mathbb{Z}_q^{4n\times 4m}$. Sample the vectors $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{m'}, \boldsymbol{u})$ uniformly over $\mathbb{Z}_q^n$.
- $gpk = (\boldsymbol{A}, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{D}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{P}, \boldsymbol{P'}, \boldsymbol{F}, \{\boldsymbol{b}_j\}_{j=1}^{m'}, \boldsymbol{u})$ and $gmsk = (\boldsymbol{T_A}, \boldsymbol{T_C}, \boldsymbol{T_{P'}})$

B. Join(U_i,GM)

Join is an interactive protocol between user U_i and group manager GM.

- User U_i who possess a key-pair $(upk[i], usk[i])$ of a digital signature scheme and performs the following steps:
  - Sample $\boldsymbol{z}_i$ according to $D_{\mathbb{Z}^{4m},\sigma}$ and compute $\boldsymbol{v}_i = \boldsymbol{F}\boldsymbol{z}_i \mod q \in \mathbb{Z}_q^{4n}$. The binary representation of $\boldsymbol{v}_i$, denoted by $bin(\boldsymbol{v}_i)$ consists of $4n\lceil\log q\rceil = 2\cdot 2n\lceil\log q\rceil = 2\acute{m}$ bits.
  - Generate the digital signature sig_i on syndrome $\boldsymbol{v}_i$ using $usk[i]$ and send $(\boldsymbol{v}_i, sig_i, upk[i])$ to GM.
- Upon receiving $(\boldsymbol{v}_i, sig_i, upk[i])$, GM checks whether sig_i is a valid signature using $upk[i]$. If it is a valid signature, GM proceeds and performs following steps:
  - Choose an identity $i \in [N]$ and compute an identity dependent matrix $\boldsymbol{A}_i$ as
$$\boldsymbol{A}_i = [\boldsymbol{A}|\boldsymbol{A}_1 + i\boldsymbol{A}_2] \in \mathbb{Z}_q^{n\times 2m}$$
and compute short basis $\boldsymbol{T}_{\boldsymbol{A}_i}$ for $\Lambda_q^{\perp}(\boldsymbol{A}_i)$ using $ExtRandBasis(\boldsymbol{A}_i, \boldsymbol{T}_A, [\boldsymbol{A}_1 + i\boldsymbol{A}_2], \sigma)$.
  - $\tau_i \in \mathbb{N}$ is the secret-key expiration date for U_i. Let $\tau_i' = [\tau_{ia}'||\tau_{ib}'] = \mathrm{H}_1(\tau_i) \in \{0,1\}^{2m}$, where $\tau_{ia}', \tau_{ib}' \in \{0,1\}^m$. Compute $\boldsymbol{C}_i\tau_i' = \boldsymbol{f}_i \mod q \in \mathbb{Z}_q^n$, where $\boldsymbol{C}_i = [\boldsymbol{C}|\boldsymbol{C}_1 + i\boldsymbol{C}_2] \in \mathbb{Z}_q^{n\times 2m}$.
  - Choose $\boldsymbol{s}_i \hookleftarrow D_{\mathbb{Z}^{2m},\sigma}$ and compute $\boldsymbol{d}_i = [\boldsymbol{d}_{i1}||\boldsymbol{d}_{i2}] \in \mathbb{Z}^{2m}$ using $SampleD(\boldsymbol{T}_{\boldsymbol{A}_i}, \boldsymbol{u} + \boldsymbol{D}bin(\boldsymbol{w}_i) + \boldsymbol{f}_i, \sigma)$ algorithm such that
$$\boldsymbol{A}_i\boldsymbol{d}_i = \boldsymbol{u} + \boldsymbol{D}bin(\boldsymbol{w}_i) + \boldsymbol{f}_i \mod q \text{ and } \|\boldsymbol{d}_i\|_\infty \le \beta \quad (1)$$
where, $\boldsymbol{w}_i = \boldsymbol{D}_0 bin(\boldsymbol{v}_i) + \boldsymbol{D}_1\boldsymbol{s}_i \mod q \in \mathbb{Z}_q^{2n}$.
  - Let $bin(\tau_i) \in \{0,1\}^{m'}$, then 1-$Enc(\tau_i) = \{\tau_{i1}, ..., \tau_{ir}\}$, where $1 \le r \le m'$. Let j be the length of binary string τ_{ij} in 1-$Enc(\tau_i)$ set. By definition j is different for each binary string in encoding set . A fixed vector $\boldsymbol{b}_j$ is used for j length τ_{ij}.
For each $j \in [r]$, compute $\mathrm{H}_2(\tau_{ij}) = \boldsymbol{\tau}'_{ij} \in \mathbb{Z}_q^n$ and generate
    - $\boldsymbol{x}_j \leftarrow SampleD(\boldsymbol{T}_\mathbf{C}, \boldsymbol{b}_j - \boldsymbol{P}\boldsymbol{\tau}'_{ij}, \sigma)$ such that
$$\boldsymbol{C}\boldsymbol{x}_j + \boldsymbol{P}\boldsymbol{\tau}'_{ij} = \boldsymbol{b}_j \text{ and } \|\boldsymbol{x}_j\|_\infty \le \beta \quad (2)$$
    - $\boldsymbol{x}'_j \leftarrow SampleD(\boldsymbol{T}_{\mathbf{P}'}, \boldsymbol{\tau}'_{ij}, \sigma)$ such that
$$\boldsymbol{P}'\boldsymbol{x}'_j = \boldsymbol{\tau}'_{ij} \text{ and } \|\boldsymbol{x}'_j\|_\infty \le \beta \quad (3)$$
  - Revocation token, grt_i=$\boldsymbol{A}\boldsymbol{d}_{i1} \mod q$ and send $cert_i$=$(i, \boldsymbol{d}_i, \boldsymbol{s}_i, \boldsymbol{f}_i, \tau_i, grt_i, \{\boldsymbol{x}_j, \boldsymbol{x}'_j\}_{j=1}^r)$ to U_i
- Upon receiving $cert_i$, user U_i checks whether eq. (2) and eq. (3) are satified for all $\boldsymbol{x}_j, \boldsymbol{x}'_j$ respectively. It also checks if eq. (1) is satisfied along with $\|\boldsymbol{d}_i\|_\infty \le \beta$, and $\|\boldsymbol{s}_i\|_\infty \le \beta$. If all the conditions are satified U_i sets its secret (signing) key as $sec_i = \boldsymbol{z}_i$ otherwise abort.
- $transcript_i = (cert_i, i, \boldsymbol{v}_i, sig_i, grt_i, upk[i])$ stored in $transcripts$ database by GM.

C. Sign$(gpk, \boldsymbol{m}, sec_i, cert_i, t, RL)$

- t is the intermediate signature expiration date such that $t < \tau_i$ and $bin(t) \in \{0,1\}^{m'}$. Compute 0-$Enc(t) = \{t_1, ..., t_{r'}\}$ where $1 \le r' \le m'$.
- Find the binary string for which $\tau_{ij} = t_j$ i.e., 1-$Enc(\tau_i) \cap$ 0-$Enc(t) = \tau_{ij} = t_j$, let j indicate the length of binary string τ_{ij}/t_j.

  Sample a vector $\boldsymbol{e}_j \leftarrow D_{\mathbb{Z}^m,\sigma}$ and compute $\boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{e}_j$. Generate the commitments $\boldsymbol{b}'_j$, $\boldsymbol{b}''_j$, and $\boldsymbol{b}'''_j$
$$\boldsymbol{b}'_j = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}'\boldsymbol{x}'_j + \boldsymbol{y}_j \mod q \quad (4)$$
$$\boldsymbol{b}''_j = \boldsymbol{C}\boldsymbol{e}_j \mod q \quad (5)$$
$$\boldsymbol{b}'''_j = \boldsymbol{C}_3\boldsymbol{y}_j \mod q \quad (6)$$
- Let y_1 is uniformly chosen over $\{0,1\}^m$ and compute $\boldsymbol{A}_3 = \mathrm{H}_3(\boldsymbol{m}, gpk, t, y_1)$. Generate the commitment for grt_i as
$$\boldsymbol{b} = \boldsymbol{A}_3 grt_i + \boldsymbol{e}' \mod q \quad (7)$$
where $\boldsymbol{e}'$ is chosen according to $D_{\mathbb{Z}^m,\sigma}$.
- Generate a Non-Interactive Zero Knowledge (NIZK) protocol Π to prove $i \in [N], \boldsymbol{z}_i, \boldsymbol{d}_{i1}, \boldsymbol{d}_{i2}, \boldsymbol{s}_i, \boldsymbol{x}_j, \boldsymbol{x}'_j, \boldsymbol{e}'$ has infinity norm bound β, equations (1),(2), (3) and (4) are satisfied, and $\boldsymbol{b}, \boldsymbol{b}''_j, \boldsymbol{b}'''_j$ are the correct committement of $grt_i, \boldsymbol{e}_j, \boldsymbol{y}_j$ respectively. This can be generated by running protocol in A., t number of times and convert it into non-interactive using Fiat-Shamir heuristic [31] i.e., $\Pi = (CMT, CH, RSP)$ where,
$$CH = (ch_1, \ldots, ch_t) = \mathrm{H}(CMT, m, t, j, \boldsymbol{b}, \boldsymbol{b}'_j, \boldsymbol{b}''_j, \boldsymbol{b}'''_j)$$

Output the signature, $\Sigma = (\Pi, t, j, \boldsymbol{b}, \boldsymbol{b}'_j, \boldsymbol{b}''_j, \boldsymbol{b}'''_j, y_1)$

D. Verify$(gpk, \boldsymbol{m}, \Sigma, t_c, RL)$

- Parse the signature $\Sigma = (\Pi, t, j, \boldsymbol{b}, \boldsymbol{b}'_j, \boldsymbol{b}''_j, \boldsymbol{b}'''_j, y_1)$. Return 1 if all of the below conditions are satisfied otherwise return 0.

  1. Current date t_c is less than the signature expiration date t i.e., $t_c < t$
  2. Let 0-$Enc(t) = \{t_1, ..., t_{r'}\}$, where $1 \le r' \le m'$, and t_j be the binary string of length j in 0-$Enc(t)$. Compute $t'_j = \mathrm{H}_2(t_j) \in \mathbb{Z}_q^n$. Check $\boldsymbol{C}_3(\boldsymbol{b}'_j - \boldsymbol{B}^{\mathrm{T}} t'_j) = \boldsymbol{b}'''_j$ and $\boldsymbol{b}_j - \boldsymbol{C}\boldsymbol{b}'_j + \boldsymbol{b}''_j = 0$. Previous two conditions ensures $t < \tau_i$.
  3. Calculate $\boldsymbol{A}_3 = \mathrm{H}(\boldsymbol{m}, gpk, t, y_1)$. For each $\boldsymbol{u}_j \in RL$, compute $\boldsymbol{c}_j = b - \boldsymbol{A}_3\boldsymbol{u}_j$ and return invalid if any $||\boldsymbol{c}_j||_\infty \le \beta$.
  4. Protocol Π is valid.

E. Open$(gpk, \boldsymbol{m}, \Sigma, RL)$

Like most of the VLR group signatures, we use *implicit tracing algorithm* to trace the identity of a signed users. The implicit tracing algorithm works as follows,
For each $i \in [N]$, run $Verify(gpk, \boldsymbol{m}, \Sigma, t_c, RL{=}grt_i)$ and return the first index $i \in [N]$ for which it returns 0. Otherwise return $\perp$.

The correctness of our scheme is proved below.

Theorem 2. *$\forall\ d, t, t_c, \tau_i, RL, \boldsymbol{m} \in \{0,1\}^*$, $(gpk, gmsk){\leftarrow}Setup(\lambda, N, d)$, $(sec_i, cert_i, grt_i, \tau_i){\leftarrow}Join$. We have, $Verify(gpk, \boldsymbol{m}, \Sigma, t_c, RL) = 1 \iff grt_i \notin RL$ and $t_c \le t < \tau_i$*

Proof: (IF Condition)
Assume, Verify algorithm described in Section (D.) returns 1. We need to prove that $grt_i \notin RL$ and $t_c \le t < \tau_i$. If Verify algorithm returns 1 indicates that all the checks in that algorithm are valid. In verify algorithm step 1 ensures that $t_c < t$ and step 2 indicates that $t < \tau_i$. In step 3, infinity norm of all $\boldsymbol{c}_j$ is greater that β indicating that $grt_i \notin RL$.
(ONLY IF condition)
Assume, $grt_i \in RL$ and $t_c \le t < \tau_i$. We need to prove that the Verify algorithm described in Section returns 1 with high probability. Verify algorithm returns 1 iff all the steps in that algorithm are valid. Since $t_c \le t < \tau_i$, step 1 and step 2 in verify algorithm are valid. Let $\boldsymbol{s}_j = grt_i - \boldsymbol{u}_j$ for each $\boldsymbol{u}_j \in RL$. Since $grt_i \notin RL$, all $\boldsymbol{s}_j$ are non-zero vectors. By Lemma 4 in [18], $Pr[\|\boldsymbol{A}_3\boldsymbol{s}_j\|_\infty \le 2\beta] \le negl(n)$. We know, for each $\boldsymbol{u}_j \in RL$

$$\begin{aligned} \boldsymbol{c}_j &= \boldsymbol{b} - \boldsymbol{A}_3\boldsymbol{u}_j \\ &= \boldsymbol{A}_3 grt_j + \boldsymbol{e}' - \boldsymbol{A}_3\boldsymbol{u}_j \\ &= \boldsymbol{A}_3\boldsymbol{s}_j + \boldsymbol{e}' \end{aligned}$$

Therefore, $\|\boldsymbol{A}_3\boldsymbol{s}_j\|_\infty \le \|\boldsymbol{c}_j\|_\infty + \|\boldsymbol{e}'\|_\infty$. Applying Lemma 4 in [18], we obtain $\|\boldsymbol{c}_j\|_\infty > \beta$. So, step 3 in verify algorithm is valid. By completeness property of protocol Π, step 4 returns valid. Therefore, verify algorithm returns 1 with high probability.

IV. SECURITY ANALYSIS

In this section we show our scheme is traceable, non-freamable and selfless-anonymous. The security of our scheme is based on the hardness assumption of LWE and SIS problem.

A. Traceability

Theorem 3. *Our scheme is secure against traceability attacks based on the hardness of SIS assumption.*

proof: Assume, there exists an adversary $\mathcal{A}$ breaking the security of our scheme against traceability with non-negligible probability. We construct an algorithm $\mathcal{B}$ that solves SIS instance $\bar{\boldsymbol{A}} = [\bar{\boldsymbol{A}}_1||\bar{\boldsymbol{A}}_2||\bar{\boldsymbol{A}}_3] \in \mathbb{Z}_q^{n\times 3m}$ with non-negligible probability. A ***coin*** is uniformly chosen over $\{1,2\}$ and $i^* \xleftarrow{\$} [N]$ and τ_{i^*} be the secret key expiration time of user i^*.

$\boldsymbol{coin = 1}$: This case handles conditions (1)$\wedge$(2) of traceability experiment.

- *Setup:*
  - Assign the matrix $\boldsymbol{A}{=}\bar{\boldsymbol{A}}_1, \boldsymbol{C}{=}\bar{\boldsymbol{A}}_3$. Run GenTrap$(n, m, q)$ three times to obtain $(\boldsymbol{A}_2, \boldsymbol{T}_{\boldsymbol{A}_2})$, $(\boldsymbol{C}_2, \boldsymbol{T}_{\boldsymbol{C}_2})$ and $(\boldsymbol{P}', \boldsymbol{T}'_{\boldsymbol{P}})$.
  - Sample the following matrices uniformly, $\boldsymbol{D}_0$ over $\mathbb{Z}_q^{2n\times 2\acute{m}}$, $\boldsymbol{D}_1$ over $\mathbb{Z}_q^{2n\times 2m}$, $\boldsymbol{F}$ over $\mathbb{Z}_q^{4n\times 4m}$, $\boldsymbol{C}_3, \boldsymbol{P}$ over $\mathbb{Z}_q^{n\times m}$, $\boldsymbol{R}, \boldsymbol{R}'$ over $\{-1,1\}^{m\times m}$, $\boldsymbol{R}''$ over $\{-1,1\}^{m\times \acute{m}}$.
    Sample vectors $\{\boldsymbol{b}_j\}_{j=1}^{m'}$ uniformly over $\mathbb{Z}_q^n$.
  - Run $SuperSamp(n, m, q, \boldsymbol{C}, \boldsymbol{P})$ to get $(\boldsymbol{B}, \boldsymbol{T}_{\boldsymbol{B}})$.
  - Compute $\boldsymbol{A}_1 = \boldsymbol{A}\boldsymbol{R} - i^*\boldsymbol{A}_2$, $\boldsymbol{C}_1 = \boldsymbol{C}\boldsymbol{R}' - i^*\boldsymbol{C}_2$, $\boldsymbol{D} = \bar{\boldsymbol{A}}_2\boldsymbol{R}'', \boldsymbol{u} = \bar{\boldsymbol{A}}_2\boldsymbol{e} \mod q$ where vector $\boldsymbol{e}$ is chosen according to $D_{\mathbb{Z}^m,\sigma}$
  - $gpk{=}(\boldsymbol{A}, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{D}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{P}, \boldsymbol{P}', \boldsymbol{F}, \{\boldsymbol{b}_j\}_{j=1}^{m'}, \boldsymbol{u})$ and $gmsk{=}(\boldsymbol{T}_{\boldsymbol{A}_2}, \boldsymbol{T}_{\boldsymbol{C}_2}, \boldsymbol{T}_{\boldsymbol{P}'})$.

- *Queries:*
  - Q_{gm} :Adversary $\mathcal{A}$ initiates join protocol by sending $(\boldsymbol{v}_n, sign_n, upk[n])$, where $sign_n$ is a valid digital signature on $\boldsymbol{v}_n$ using $upk[n]$ and $\boldsymbol{v}_n = \boldsymbol{F}\boldsymbol{z}_n$.
    $\mathcal{B}$ Increment n by 1 and fixes a secret key expiration time τ_n. $\mathcal{B}$ computes $\boldsymbol{A}_n = [\boldsymbol{A}|\boldsymbol{A}_1 + n\boldsymbol{A}_2]$, $\boldsymbol{C}_n = [\boldsymbol{C}|\boldsymbol{C}_1 + n\boldsymbol{C}_2]$, $H_1(\tau_n) = \tau'_n$, $\boldsymbol{C}_n\tau'_n = \boldsymbol{f}_n$. Using $\boldsymbol{T}_{\boldsymbol{A}_2}$ obtain the vector $\boldsymbol{d}_n = [\boldsymbol{d}_{n,1}||\boldsymbol{d}_{n,2}]$ such that $\boldsymbol{A}_n\boldsymbol{d}_n = \boldsymbol{u} + \boldsymbol{D}bin(\boldsymbol{D}_0 bin(\boldsymbol{v}_n) + \boldsymbol{D}_1\boldsymbol{s}_n) + \boldsymbol{f}_n \mod q$ where $\boldsymbol{s}_n$ is chosen according to $D_{\mathbb{Z}^{2m},\sigma}$.
    Let 1-$ENC(\tau_n){=}\{\tau_{n1}, \tau_{n2}, \ldots, \tau_{nr}\}$, for each $\boldsymbol{\tau}'_{nj}{=}\mathrm{H}_2(\tau_{nj})$, compute $\boldsymbol{x}_j$ using $\boldsymbol{T}_{\mathbf{C}}$ and $\boldsymbol{x}'_j$ using $\boldsymbol{T}'_{\mathbf{P}}$. Set $grt_n = \boldsymbol{A}\boldsymbol{d}_{n,1}$, send $cert_n{=}(n, \boldsymbol{d}_n, \boldsymbol{s}_n, \boldsymbol{f}_n, \tau_n, grt_n, \{\boldsymbol{x}_j, \boldsymbol{x}'_j\}_{j=1}^{r})$ to $\mathcal{A}$, $gsk_n = \boldsymbol{z}_n$ and add n to the set C_u.
  - G_{sign}: If $i \notin C_u$ or $i = i^*$ abort. Otherwise select an intermediate signature expiration date t such that $t < \tau_i$, then generate the signature Σ on message m with signature expiration date t using sec_i. Add $(\boldsymbol{m}, \Sigma, t)$ to $Sigs$

– *Revoke* : If $i = i^*$ abort, otherwise $\mathcal{B}$ search in *transcripts* list for i to get revocation token grt_i, add grt_i to RL.

– Get_{reg} : On input index i, return (grt_i, τ_i) by searching in *transcript* database for i.

- *Forgery:* $\mathcal{A}$ outputs $(\boldsymbol{m}^*, \Sigma^*, RL^*)$ such that $\text{Verify}(gpk, \boldsymbol{m}^*, \Sigma^*, t_c{}^*, RL^*)$=1. If $\text{Open}(gpk, \boldsymbol{m}^*, \Sigma^*, RL^*)$=$k{\in}C_u$ and $k{\neq}i^*$ abort. Otherwise, parse Π^*=(CMT, CH, RSP) and Σ=$(\Pi^*, t, j, \boldsymbol{b}^*, \boldsymbol{b}'^*_j, \boldsymbol{b}''^*_j, y^*_1)$. $\mathcal{A}$ must have queried the random oracle H on input $(CMT^*, \boldsymbol{m}^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}'^*_j, \boldsymbol{b}''^*_j, \boldsymbol{b}'''^*_j)$ with high probability. Otherwise,

$$Pr[\{ch\}^t_{i=1} = \text{H}(CMT^*, \boldsymbol{m}^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}'^*_j, \boldsymbol{b}''^*_j, \boldsymbol{b}'''^*_j)] \le \frac{1}{3^t}$$

Therefore with $\epsilon - 3^{-t}$ probability, there exists an index $\kappa^* \le Q_{\text{H}}$. At this stage, algorithm $\mathcal{B}$ runs $\mathcal{A}$ with same input and random tape as in original execution. Pick κ^* as the target point and replay $\mathcal{A}$ many times with the same random tape and input. Each time, first $\kappa^* - 1$ queries are answered as $r_1, ..., r_{\kappa^*-1}$ and from κ^*th query the answers are uniformly chosen from $\{1, 2, 3\}^t$. The Improved Forking Lemma [32] implies that, with probability greater than $\frac{1}{2}$, $\mathcal{B}$ can obtain a 3-fork involving tuple $(CMT^*, \boldsymbol{m}^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}'^*_j, \boldsymbol{b}''^*_j, \boldsymbol{b}'''^*_j)$. Let the answers of $\mathcal{B}$ with respect to the 3-fork branches be

$$r^{(1)}_{\kappa^*} = (ch^{(1)}_1, ...ch^{(1)}_t); r^{(2)}_{\kappa^*} = (ch^{(2)}_1, ...ch^{(2)}_t); r^{(3)}_{\kappa^*} = (ch^{(3)}_1, ...ch^{(3)}_t)$$

$$Pr[\exists k \in [t] : \{ch^{(1)}_k, ch^{(2)}_k, ch^{(3)}_k\} = \{1, 2, 3\}] = (1 - (\frac{7}{9})^t)$$

If such k exists, parse the 3-forgeries corresponding to 3-fork branches to obtain $(\text{RSP}^{(1)}_k, \text{RSP}^{(2)}_k, \text{RSP}^{(3)}_k)$. Given three different challenges and three valid responses for same commitment CMT_k, using witness extraction procedure, extract the witness $(k, \boldsymbol{d}_k = [\boldsymbol{d}_{k1}||\boldsymbol{d}_{k2}], \boldsymbol{z}_k, \boldsymbol{s}_k, \boldsymbol{f}_k)$. Algorithm $\mathcal{B}$ aborts if $k \neq i^*$. We know $\boldsymbol{A}_k\boldsymbol{d}_k = \boldsymbol{u} + \boldsymbol{D}bin(\boldsymbol{w}_k) + \boldsymbol{f}_k \mod q$ where $\boldsymbol{w}_k = \boldsymbol{D}_0 bin(\boldsymbol{v}_k) + \boldsymbol{D}_1\boldsymbol{s}_k$.

$$\begin{aligned}
&\boldsymbol{A}_k\boldsymbol{d}_k = \boldsymbol{u} + \boldsymbol{D}bin(\boldsymbol{w}_k) + \boldsymbol{f}_k \mod q\\
&[\bar{\boldsymbol{A}}_1|\bar{\boldsymbol{A}}_1\boldsymbol{R}][\boldsymbol{d}_{k1}||\boldsymbol{d}_{k2}] = \bar{\boldsymbol{A}}_2\boldsymbol{e} + \bar{\boldsymbol{A}}_2\boldsymbol{R}''bin(\boldsymbol{w}_k) +\\
&[\bar{\boldsymbol{A}}_3|\bar{\boldsymbol{A}}_3\boldsymbol{R}'][\tau'_{ka}||\tau'_{kb}] \mod q\\
&\bar{\boldsymbol{A}}_1(\boldsymbol{d}_{k1} + \boldsymbol{R}\boldsymbol{d}_{k2}) - \bar{\boldsymbol{A}}_2(\boldsymbol{R}''bin(\boldsymbol{w}_j) + \boldsymbol{e})\\
&- \bar{\boldsymbol{A}}_3(\tau'_{ka} + \boldsymbol{R}'\tau'_{kb}) = 0 \mod q
\end{aligned}$$

Let $\bar{\boldsymbol{x}} = (\boldsymbol{d}_{k1} + \boldsymbol{R}\boldsymbol{d}_{k2}|| - (\boldsymbol{R}''bin(\boldsymbol{w}_k) + \boldsymbol{e})|| - (\tau'_{ka} + \boldsymbol{R}'\tau'_{kb}))$. Therefore, $\bar{\boldsymbol{A}}\bar{\boldsymbol{x}} = 0 \mod q$ and $\|\bar{\boldsymbol{x}}\| \le \sqrt{m((5m + 1)\beta^2)}$.

$\boldsymbol{coin = 2}$: This case handles conditions (1)∧(3) of traceability experiment.

- *Setup:*

  – Run GenTrap(n, m, q) two times to obtain $(\boldsymbol{A}, \boldsymbol{T}_{\boldsymbol{A}})$, $(\boldsymbol{C}_2, \boldsymbol{T}_{\boldsymbol{C}_2})$.

  – Sample the following matrices uniformly, $\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{C}_1, \boldsymbol{C}_3$ over $\mathbb{Z}^{n\times m}_q$, $\boldsymbol{D}$ over $\mathbb{Z}^{n\times \acute{m}}_q$, $\boldsymbol{D}_0$ over $\mathbb{Z}^{2n\times 2\acute{m}}_q$, $\boldsymbol{D}_1$ over $\mathbb{Z}^{2n\times 2m}_q$, $\boldsymbol{F}$ over $\mathbb{Z}^{4n\times 4m}_q$, $\boldsymbol{R}'$ over $\{-1, 1\}^{m\times n}$. An invertible matrix $\boldsymbol{P}$ over $\mathbb{Z}^{n\times n}_q$.
  Sample the vector $\boldsymbol{u}$ uniformly over $\mathbb{Z}^n_q$.

  – For each $j \in [m']$ assign $\boldsymbol{b}_j = \bar{\boldsymbol{A}}_3\boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is sampled over $D_{\mathbb{Z}^m,\sigma}$. Assign $\boldsymbol{C} = \bar{\boldsymbol{A}}_1$

  – Choose a binary matrix $\boldsymbol{R}_1 \in \{0, 1\}^{n\times n}$ such that its inverse is also a binary matrix. Let $\boldsymbol{R} = [\boldsymbol{R}_1|0^{m-n}] \in \{0, 1\}^{n\times m}$, $\boldsymbol{Q} = {\boldsymbol{P}^{-1}}^{\text{T}}{\boldsymbol{R}'}^{\text{T}}\bar{\boldsymbol{A}}^{\text{T}}_2$. Run *SuperSamp*$(n, m, q, \boldsymbol{R}, \boldsymbol{Q})$ to get $(\boldsymbol{P}', \boldsymbol{T}_{\boldsymbol{P}'})$.

  – $gpk = (\boldsymbol{A}, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{D}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{P}, \boldsymbol{P}', \boldsymbol{F}, \{\boldsymbol{b}_j\}^{m'}_{j=1}, \boldsymbol{u})$ and $gmsk$=$(\boldsymbol{T}_{\boldsymbol{A}}, \boldsymbol{T}_{\boldsymbol{C}_2}, \boldsymbol{T}_{\boldsymbol{P}'})$.

- *Queries:*

  – Q_{gm} : Adversary $\mathcal{A}$ initiates join protocol by sending $(\boldsymbol{v}_n, sig_n, upk[n])$, where sig_n is a valid digital signature on $\boldsymbol{v}_n$ using $upk[n]$ and $\boldsymbol{v}_n = \boldsymbol{F}\boldsymbol{z}_n$.
  $\mathcal{B}$ Increment n by 1 and fixes a secret key expiration time τ_n. $\mathcal{B}$ computes $\boldsymbol{A}_n = [\boldsymbol{A}|\boldsymbol{A}_1 + n\boldsymbol{A}_2]$, $\boldsymbol{C}_n = [\boldsymbol{C}|\boldsymbol{C}_1 + n\boldsymbol{C}_2]$, $H_1(\tau_n) = \tau'_n$, $\boldsymbol{C}_n\tau'_n = \boldsymbol{f}_n$. Using $\boldsymbol{T}_{\boldsymbol{A}}$ obtain the vector $\boldsymbol{d}_n = [\boldsymbol{d}_{n,1}||\boldsymbol{d}_{n,2}]$ such that $\boldsymbol{A}_n\boldsymbol{d}_n = \boldsymbol{u} + \boldsymbol{D}(bin(\boldsymbol{D}_0 bin(\boldsymbol{v}_n) + \boldsymbol{D}_1 bin(\boldsymbol{s}_n)) + \boldsymbol{f}_n \mod q$ where $\boldsymbol{s}_n$ is chosen according to $D_{\mathbb{Z}^{2m},\sigma}$. Let 1-*ENC*$(\tau_n) = \{\tau_{n1}, \tau_{n2}, \ldots, \tau_{nr}\}$. For each $\boldsymbol{\tau}'_{nj} = \text{H}_2(\tau_{nj})$, sample $\boldsymbol{x}_j$ according to $D_{\mathbb{Z}^m,\sigma}$ and using $\boldsymbol{T}_{\boldsymbol{P}'}$ sample $\boldsymbol{x}'_j$ such that $\boldsymbol{C}\boldsymbol{x}_j + \boldsymbol{P}\boldsymbol{\tau}'_{ij} = \boldsymbol{b}_j$ and $\boldsymbol{P}'\boldsymbol{x}'_j = \boldsymbol{\tau}'_{ij}$. Set $grt_n = \boldsymbol{A}\boldsymbol{d}_{n,1}$, send $cert_n = (n, \boldsymbol{d}_n, \boldsymbol{s}_n, \boldsymbol{f}_n, \tau_n, grt_n, \{\boldsymbol{x}_j, \boldsymbol{x}'_j\}^r_{j=1})$ to $\mathcal{A}$, $gsk_n = \boldsymbol{z}_n$ and add n to the set C_u.

  – G_{sign} : If $i \notin C_u$ `or` $i = i^*$ abort. Otherwise select an intermediate signature expiration date t such that $t < \tau_i$ generate the signature Σ on message m with signature expiration time t using sec_i . Add $(\boldsymbol{m}, \Sigma, t)$ to *Sigs*.

  – *Revoke* : If $i = i^*$ abort, otherwise $\mathcal{B}$ search in *transcripts* list for i to get revocation token grt_i, add grt_i to RL.

  – Get_{reg} : On input index i, return (grt_i, τ_i) by searching in *transcript* list for i.

- *Forgery:* $\mathcal{A}$ outputs $(\boldsymbol{m}^*, \Sigma^*, RL^*)$ such that $\text{Verify}(gpk, \boldsymbol{m}^*, \Sigma^*, t_c{}^*, RL^*)$=1. If $\text{Open}(gpk, \boldsymbol{m}^*, \Sigma^*, RL^*)$=$k{\in}C_u$ or $k{\neq}i^*$ abort. Otherwise, parse Π^*=(CMT, CH, RSP) and Σ^*=$(\Pi^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}'^*_j, \boldsymbol{b}''^*_j, \boldsymbol{b}'''^*_j, y^*_1)$. $\mathcal{A}$ must have queried the random oracle H on input $(CMT^*, \boldsymbol{m}^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}'^*_j, \boldsymbol{b}''^*_j, \boldsymbol{b}'''^*_j)$ with high probability. Otherwise,

$$Pr[\{ch\}^t_{i=1} = \text{H}(CMT^*, \boldsymbol{m}^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}'^*_j, \boldsymbol{b}''^*_j, \boldsymbol{b}'''^*_j)] \le \frac{1}{3^t}$$

Therefore with $\epsilon - 3^{-t}$ probability, there exists an index $\kappa^* \le Q_{\text{H}}$. At this stage, algorithm $\mathcal{B}$ runs

$\mathcal{A}$ with same input and random tape as in original execution. Pick κ^* as the target point and replay $\mathcal{A}$ many times with the same random tape and input. Each time, first $\kappa^* - 1$ queries are answered as $r_1, ..., r_{\kappa^*-1}$ and from $\kappa^* th$ query the answers are uniformly chosen from $\{1,2,3\}^t$. The Improved Forking Lemma [32] implies that, with probability greater than $\frac{1}{2}$, $\mathcal{B}$ can obtain a 3-fork involving tuple $(CMT^*, \boldsymbol{m}^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}_j'^*, \boldsymbol{b}_j''^*, \boldsymbol{b}_j'''^*)$. Let the answers of $\mathcal{B}$ with respect to the 3-fork branches be

$$r_{\kappa^*}^{(1)} = (ch_1^{(1)}, ...ch_t^{(1)}); r_{\kappa^*}^{(2)} = (ch_1^{(2)}, ...ch_t^{(2)});$$
$$r_{\kappa^*}^{(3)} = (ch_1^{(3)}, ...ch_t^{(3)})$$

$$Pr[\exists k \in [t] : \{ch_k^{(1)}, ch_k^{(2)}, ch_k^{(3)}\} = \{1,2,3\}]$$
$$= (1 - (\frac{7}{9})^t)$$

If such j exists, parse the 3-forgeries corresponding to 3-fork branches to obtain $(\mathrm{RSP}_k^{(1)}, \mathrm{RSP}_k^{(2)}, \mathrm{RSP}_k^{(3)})$. Given three different challenges and three valid responses for same commitment CMT_k, using witness extraction procedure, the witness $(k, \boldsymbol{x}_j, \boldsymbol{x}_j')$ can be extracted. Algorithm $\mathcal{B}$ aborts if $j \neq i^*$. We know

$$\boldsymbol{C}\boldsymbol{x}_j + \boldsymbol{P}\boldsymbol{\tau}_{kj}' = \boldsymbol{b}_j$$
$$\bar{\boldsymbol{A}}_1\boldsymbol{x}_j + \bar{\boldsymbol{A}}_2\boldsymbol{R}'\boldsymbol{R}^{\mathrm{T}^{-1}}\boldsymbol{x}_j' = \bar{\boldsymbol{A}}_3\boldsymbol{e}_j$$
$$\bar{\boldsymbol{A}}_1\boldsymbol{x}_j + \bar{\boldsymbol{A}}_2\boldsymbol{R}'\boldsymbol{R}^{\mathrm{T}^{-1}}\boldsymbol{x}_j' - \bar{\boldsymbol{A}}_3\boldsymbol{e}_j = 0$$

Let $\bar{\boldsymbol{x}}$=$(\boldsymbol{x}_j||\boldsymbol{R}'\boldsymbol{R}^{\mathrm{T}^{-1}}\boldsymbol{x}_j'|| - \boldsymbol{e}_j)$. Therefore, $\bar{\boldsymbol{A}}\bar{\boldsymbol{x}} = 0 \mod q$ and $\|\bar{\boldsymbol{x}}\| \leq \beta\sqrt{2m + m^2n^2}$.

B. Framing attacks

Theorem 4. *Our scheme is secure against framing attacks based on the hardness of SIS assumption.*

proof: Let $\mathcal{A}$ be an adversary that generates a forgery $(\boldsymbol{m}^*, \Sigma^*, RL^*)$ which opens to the honest user i^* who did not sign the message $\boldsymbol{m}^*$. We construct an algorithm $\mathcal{B}$ that solves an instance of SIS assumption i.e., given a matrix $\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{4n\times 4m}$ as input, algorithm $\mathcal{B}$ finds a vector $\bar{\boldsymbol{x}}$ such that $\bar{\boldsymbol{A}}\bar{\boldsymbol{x}} = 0 \mod q$ and $\|\bar{\boldsymbol{x}}\| \leq 2\beta\sqrt{m}$.
Steps:
Setup: Obtain $(gpk, gmsk)$ using Setup(λ, N, d) (described in section A.) with one modification. Instead of uniformly choosing $\boldsymbol{F} \in \mathbb{Z}_q^{4n\times 4m}$, we assign $\boldsymbol{F} = \bar{\boldsymbol{A}}$.
Queries:

- Get_{gmsk} : return $gmsk$ to $\mathcal{A}$
- Q_{user} : Here adversary $\mathcal{A}$ corrupts group manager and engage in join protocol with any honest user i. At each query, $\mathcal{B}$ runs join protocol on behalf of honest user. After successful join i is added to H_u set.
- G_{sign} : If $\mathcal{A}$ requests for the signature on message $\boldsymbol{m}$ of user i and $i \in H_u$. Select an intermediate signature expiry date t such that $t < \tau_i$, recall $(cert_i, sec_i,)$ and generate signature using sign$(gpk, \boldsymbol{m}, sec_i, cert_i, t, RL)$ algorithm. Add $(\boldsymbol{m}, \Sigma, t)$ to $Sigs$.
- Get_{reg} : On input index i, return (grt_i, τ_i) by searching in *transcript* database for i.
- Get_{usk} : On input index i, return the $(cert_i, sec_i)$. Add i to C_u set.

Forgery: Let $\mathcal{A}$ outputs $(m^*, \Sigma^*, RL^*, i^*)$ such that Verify$(gpk, m^*, \Sigma^*, t_c^*, RL^*)$=1 with non-negligible probability ϵ. Let $\Sigma^* = (\Pi^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}_j'^*, \boldsymbol{b}_j''^*, \boldsymbol{b}_j'''^*, y_1^*)$.

parse Π^*=(CMT, CH, RSP), Adversary $\mathcal{A}$ must have queried the random oracle H on input $(CMT^*, m^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}_j'^*, \boldsymbol{b}_j''^*, \boldsymbol{b}_j'''^*)$ with high probability. Otherwise,

$$Pr[\{ch\}_{i=1}^t = \mathrm{H}(CMT^*, m^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}_j'^*, \boldsymbol{b}_j''^*, \boldsymbol{b}_j'''^*)]$$
$$\leq \frac{1}{3^t}$$

Therefore with $\epsilon - 3^{-t}$ probability, there exists an index $\kappa^* \leq Q_{\mathrm{H}}$. At this stage, algorithm $\mathcal{B}$ runs $\mathcal{A}$ with same input and random tape as in original execution. Pick κ^* as the target point and replay $\mathcal{A}$ many times with the same random tape and input. Each time, first κ^*-1 queries are answered as $r_1, ..., r_{\kappa^*-1}$ and from $\kappa^* th$ query the answers are uniformly chosen from $\{1,2,3\}^t$. The Improved Forking Lemma [32] implies that, with probability greater than $\frac{1}{2}$, $\mathcal{B}$ can obtain a 3-fork involving tuple $(CMT^*, m^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}_j'^*, \boldsymbol{b}_j''^*, \boldsymbol{b}_j'''^*)$. Let the answers of $\mathcal{B}$ with respect to the 3-fork branches be

$$r_{\kappa^*}^{(1)} = (ch_1^{(1)}, ...ch_t^{(1)}); r_{\kappa^*}^{(2)} = (ch_1^{(2)}, ...ch_t^{(2)});$$
$$r_{\kappa^*}^{(3)} = (ch_1^{(3)}, ...ch_t^{(3)})$$

$$Pr[\exists k \in [t] : \{ch_k^{(1)}, ch_k^{(2)}, ch_k^{(3)}\} = \{1,2,3\}]$$
$$= (1 - (\frac{7}{9})^t)$$

If such k exists, parse the 3-forgeries corresponding to 3-fork branches to obtain $(\mathrm{RSP}_k^{(1)}, \mathrm{RSP}_k^{(2)}, \mathrm{RSP}_k^{(3)})$. Given three different challenges and three valid responses for same commitment CMT_k, using witness extraction procedure, we obtain the witness $(k, \boldsymbol{d}_k, \boldsymbol{z}_k, \boldsymbol{s}_k, \boldsymbol{f}_k)$.
We consider the cases where $\mathcal{A}$ returns 1 in framing experiment $Exp_{\mathcal{A}}^{non-frame}$ and show that $\mathcal{B}$ solves SIS instance.

Open algorithm obtains the vector $\boldsymbol{v}_{k^*}$. Recall $\boldsymbol{z}_{i^*}$ when answering Q_{user} query such that $\boldsymbol{F}\boldsymbol{z}_{k^*} = \boldsymbol{v}_{k^*}$. We know $\boldsymbol{v}_{k^*} = \boldsymbol{F}\boldsymbol{z}_k$, where $\boldsymbol{z}_k$ is obtained using witness extraction procedure. Due to the statistical witness indistinguishability of stern extension protocol, $\boldsymbol{z}_k \neq \boldsymbol{z}_{i^*}$ with high probability. Let $\bar{\boldsymbol{x}} = \boldsymbol{z}_k - \boldsymbol{z}_{i^*}$. We know $\boldsymbol{F}\bar{\boldsymbol{x}} = 0 \mod q$ and $\|\bar{\boldsymbol{x}}\| \leq 2\beta\sqrt{m}$. Hence $\bar{\boldsymbol{x}}$ is a solution to SIS instance

C. Anonymity Attack

Theorem 5. *Our scheme is secure against anonymity attacks based on the zero knowledge property of NIZK protocol* Π, *and hardness of LWE assumption.*

Game $G^{(b)}$**:** This game is same as original anonymity game. In this game, challenger $\mathcal{B}$ runs setup algorithm to generate $(gpk, gmsk)$ and gives gpk to adversary $\mathcal{A}$.

Challenger $\mathcal{B}$ replies to all the queries of the adversary. After polynomial number of queries , $\mathcal{A}$ sends a challenge message $\boldsymbol{m}^*$, two legitimate identities (i_0, i_1) such that $i_0 \neq i_1$. Challenger $\mathcal{B}$ selects an identity i_b, where b is uniformly choosen over $\{0,1\}$ and generates the challenge signature $\Sigma^* = (\Pi^*, t^*, j^*, \boldsymbol{b}^*, \boldsymbol{b}_j'^*, \boldsymbol{b}_j''^*, \boldsymbol{b}_j'''^*, y_1^*)$. Finally, $\mathcal{A}$ outputs the bit $b' \in \{0,1\}$.

Game $G_1^{(b)}$**:** This game is similar to $G^{(b)}$, except the protocol Π^* is replaced by a simulator output Sim^*. The transcripts of the protocol Π^*, is simulated using the simulator of Π^*, the two transcripts Π^* and Sim^* are statistically indistinguishable because of the statistical zero knowledge property of Π^*. Hence Challenge signature generated in this game is computationally indistinguishable from the signature generated in game $G^{(b)}$.

Game $G_2^{(b)}$**:** This game is similar to $G_1^{(b)}$, except the steps to generate commitment b, b_j' are changed. In game $G_1^{(b)}$, b and b_j' are computed as $b = \boldsymbol{A}_3 grt_i + \boldsymbol{e}_1$ and $b_j' = \boldsymbol{B}^{\mathrm{T}} \boldsymbol{P}' \boldsymbol{x}_j + \boldsymbol{y}_j$ respectively. Instead, they are computed as $b = \boldsymbol{A}_3 \boldsymbol{s} + \boldsymbol{e}_1$ and $b_j' = \boldsymbol{B}^{\mathrm{T}} \boldsymbol{s}_1 + \boldsymbol{y}_j$, where $\boldsymbol{s}$ and $\boldsymbol{s}_1$ are uniformly chosen over $\mathbb{Z}_q^n$. The signature Σ^* generated in this game is statistically close to the signature generated in game $G_1^{(b)}$ because of Lemma 1.

Game $G_3^{(b)}$**:** This game is similar to $G_2^{(b)}$, except the steps to genrate commitment b_j'', and b_j''' are changed. In game $G_2^{(b)}$, b_j'' and b_j''' are computed as $b_j'' = \boldsymbol{C}\boldsymbol{e}_j$ and $b_j''' = \boldsymbol{D}\boldsymbol{y}_j$ respectively. Instead, b_j'' and b_j''' are uniformly chosen over $\mathbb{Z}_q^n$. The signature Σ^* generated in this game is statistically close to the signature generated in game $G_1^{(b)}$ because of Lemma 1.

Game G_4**:** This game is similar to $G_3^{(b)}$, except the steps to genrate commitment b, b_j' are changed. In game $G_3^{(b)}$, b and b_j' are computed as $b = \boldsymbol{A}_3 \boldsymbol{s} + \boldsymbol{e}_1$ and $b_j' = \boldsymbol{B}^{\mathrm{T}} \boldsymbol{s}_1 + \boldsymbol{y}_j$ respectively. Instead, b and b_j' are uniformly chosen over $\mathbb{Z}_q^m$. Signature Σ^* generated in this game is statistically close to signature generated in game $G_3^{(b)}$ because of the hardness of decision version of LWE.

We know the challenge signature Σ^* generated in the game *Game* G_4 is independent of the both the user i_0 and i_1, and the challenge signature Σ^* in the original anonimity game is computationally indistinguishable with the challenge signature Σ^* generated in the game *Game* G_4. Therefore advantage of the adversary $\mathcal{A}$ is negligible.

V. CONCLUSION

We have presented a new lattice based dynamic group signature using VLR by fixing an expiry date for each signing key. With this feature, any group member whose signing key has expired is naturally revoked and he cannot generate a valid group signature after that. Revocation tokens of only prematurely revoked members are kept in the revocation list. This results in a significantly smaller revocation list improving the efficiency of the verification process. Our scheme is particularly efficient in scenarios where the fraction of premature revocation is very less compared to natural revocation. Our scheme is proved to be secure based on the hardness of LWE and SIS assumptions in the random oracle model.

REFERENCE

[1] E. Chaum, David & Van Heyst, Group signatures, in: Workshop on the Theory and Application of of Cryptographic Techniques, Springer, 1991, pp. 257–265.

[2] M. Bellare, D. Micciancio, B. Warinschi, Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions, in: International conference on the theory and applications of cryptographic techniques, Springer, 2003, pp. 614–629.

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

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

[5] D. Boneh, H. Shacham, Group signatures with verifier-local revocation, in: Proceedings of the 11th ACM conference on Computer and communications security, 2004, pp. 168–177.

[6] A. Mehmood, I. Natgunanathan, Y. Xiang, H. Poston, Y. Zhang, Anonymous authentication scheme for smart cloud based healthcare applications, IEEE access 6 (2018) 33552–33567.

[7] A. Sudarsono, M. U. H. Al Rasyid, An anonymous authentication system in wireless networks using verifier-local revocation group signature scheme, in: 2016 International Seminar on Intelligent Technology and Its Applications (ISITIA), IEEE, 2016, pp. 49–54.

[8] H. Zheng, Q. Wu, B. Qin, L. Zhong, S. He, J. Liu, Linkable group signature for auditing anonymous communication, in: Australasian Conference on Information Security and Privacy, Springer, 2018, pp. 304–321.

[9] J. Bringer, H. Chabanne, D. Pointcheval, S. Zimmer, An application of the boneh and shacham group signature scheme to biometric authentication, in: International Workshop on Security, Springer, 2008, pp. 219–230.

[10] X. Lin, R. Lu, Gsis: group signature and id-based signature-based secure and privacy-preserving protocol (2015).

[11] C.-K. Chu, J. K. Liu, X. Huang, J. Zhou, Verifier-local revocation group signatures with time-bound keys, in: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, 2012, pp. 26–27.

[12] M. Ajtai, Generating hard instances of lattice problems, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 1996, pp. 99–108.

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

[14] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, Journal of the ACM (JACM) 56 (6) (2009) 1–40.

[15] S. D. Gordon, J. Katz, V. Vaikuntanathan, A group signature scheme from lattice assumptions, in: International conference on the theory and application of cryptology and information security, Springer, 2010, pp. 395–412.

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

[17] F. Laguillaumie, A. Langlois, B. Libert, D. Stehlé, Lattice-based group signatures with logarithmic signature size, in: International conference on the theory and application of cryptology and information security, Springer, 2013, pp. 41–61.

[18] A. Langlois, S. Ling, K. Nguyen, H. Wang, Lattice-based group signature scheme with verifier-local revocation, in: International workshop on public key cryptography, Springer, 2014, pp. 345–361.

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

[20] Y. Zhang, Y. Hu, W. Gao, M. Jiang, Simpler efficient group signature scheme with verifier-local revocation from lattices, KSII Transactions on Internet and Information Systems (TIIS) 10 (1) (2016) 414–430.

[21] W. Gao, Y. Hu, Y. Zhang, B. Wang, Lattice-based group signature with verifier-local revocation, Journal of Shanghai Jiaotong University (Science) 22 (3) (2017) 313–321.

[22] M. N. S. Perera, T. Koshiba, Fully dynamic group signature scheme with member registration and verifier-local revocation, in: International conference on mathematics and computing, Springer, 2018, pp. 399–415.

[23] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, H. Wang, Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2016, pp. 373–403.

[24] K. Emura, T. Hayashi, A. Ishida, Group signatures with time-bound keys revisited: A new model and an efficient construction, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 777–788.

[25] C. Peikert, A. Rosen, Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices, in: Theory of Cryptography Conference, Springer, 2006, pp. 145–166.

[26] J. Alwen, C. Peikert, Generating shorter bases for hard random lattices, in: 26th International Symposium on Theoretical Aspects of Computer Science STACS 2009, IBFI Schloss Dagstuhl, 2009, pp. 75–86.

[27] D. Cash, D. Hofheinz, E. Kiltz, C. Peikert, Bonsai trees, or how to delegate a lattice basis, in: Annual international conference on the theory and applications of cryptographic techniques, Springer, 2010, pp. 523–552.

[28] C. Peikert, A decade of lattice cryptography, Foundations and Trends® in Theoretical Computer Science 10 (4) (2016) 283–424.

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

[30] H.-Y. Lin, W.-G. Tzeng, An efficient solution to the millionaires' problem based on homomorphic encryption, in: International Conference on Applied Cryptography and Network Security, Springer, 2005, pp. 456–466.

[31] A. Fiat, A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, in: Conference on the theory and application of cryptographic techniques, Springer, 1986, pp. 186–194.

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

[33] B. Libert, F. Mouhartem, K. Nguyen, A lattice-based group signature scheme with message-dependent opening, in: International Conference on Applied Cryptography and Network Security, Springer, 2016, pp. 137–155.

APPENDIX

A. Underlying Interactive Zero-Knowledge Argument System

Let D, L be positive integers. In 2016, Libert et al. [23] proposed an interactive zero-knowledge protocol for the relation R defined below.

$$R = \{(\boldsymbol{P}, \boldsymbol{y}; \boldsymbol{x}) \in \mathbb{Z}_q^{D\times L} \times \mathbb{Z}_q^{D} \times Valid : \boldsymbol{P}\boldsymbol{x} = \boldsymbol{y} \mod q\} \quad (8)$$

where, $Valid$ is the subset of $\{-1, 0, 1\}^L$ satisfying the following conditions:

$$\boldsymbol{x} \in Valid \iff T_\pi(\boldsymbol{x}) \in Valid \quad (9)$$

$$\text{If } \boldsymbol{x} \in Valid \text{ and } \pi \text{ is uniform in } S \text{ then } T_\pi(\boldsymbol{x}) \text{ is uniform in } Valid \quad (10)$$

where T_π is the permutation of L elements and set S is the permutation of m elements.

In this section, we construct an interactive zero-knowledge protocol that allows the signer to convince following conditions to the verifier.

1. Signer i is a certified group member i.e, he possess a valid secret key, $sec_i = \boldsymbol{z}_i$ and a membership certificate $cert_i = (i, \boldsymbol{d}_i, \boldsymbol{s}_i, \boldsymbol{f}_i, \tau_i, grt_i, \{\boldsymbol{x}_j, \boldsymbol{x}'_j\}_{j=1}^{r})$.
2. $\boldsymbol{x}_j$ and $\boldsymbol{x}'_j$ are the short vectors associated with $\boldsymbol{\tau}'_{ij}$.
3. The commitment $\boldsymbol{b}'_j$ obtained using the LWE function, is the correct commitment of $\boldsymbol{P}'\boldsymbol{x}_j$.
4. The commitment $\boldsymbol{b}''_j, \boldsymbol{b}'''_j$ is the correct commitment of $\boldsymbol{e}_j, \boldsymbol{y}_j$.

All the above conditions can be defined as a relation R'.

Definition 8. *: The relation $\boldsymbol{R}'$ is defined as follows:*

$$\begin{aligned}\boldsymbol{R}' =\{&\boldsymbol{A}, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{D}, \boldsymbol{D}_0, \boldsymbol{D}_1, \boldsymbol{P}, \boldsymbol{P}', \boldsymbol{F}, \{\boldsymbol{b}_j\}_{j=1}^{m'}, \boldsymbol{u};\\ &i, \boldsymbol{z}_i, \boldsymbol{v}_i, \boldsymbol{w}_i, \boldsymbol{d}_{i1}, \boldsymbol{d}_{i2}, \boldsymbol{s}_i, \boldsymbol{f}_i, \tau'_{ia}, \tau'_{ib}, \tau'_{ij}, \boldsymbol{b}, \boldsymbol{b}_j, \boldsymbol{b}'_j, \boldsymbol{b}''_j, \boldsymbol{b}'''_j, \boldsymbol{e}_j, \boldsymbol{y}_j, \boldsymbol{e}\}\end{aligned}$$

where

$\boldsymbol{A}, \boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{P}' \in \mathbb{Z}_q^{n\times m}, \boldsymbol{D} \in \mathbb{Z}_q^{n\times \acute{m}}$, $\boldsymbol{D}_0 \in \mathbb{Z}_q^{2n\times 2\acute{m}}, \boldsymbol{D}_1 \in \mathbb{Z}_q^{2n\times 2m}, \boldsymbol{F} \in \mathbb{Z}_q^{4n\times 4m}, \{\boldsymbol{b}_j\}_{j=1}^{m'}, \boldsymbol{u} \in \mathbb{Z}_q^{n}$, $i \in [N], \boldsymbol{z}_i \in [-\beta, \beta]^{4m}, \boldsymbol{v}_i \in \mathbb{Z}_q^{4n}, \boldsymbol{w}_i \in \mathbb{Z}_q^{2n}, \boldsymbol{d}_{i1}, \boldsymbol{d}_{i2} \in [-\beta, \beta]^{m}, \boldsymbol{s}_i \in [-\beta, \beta]^{2m}, \tau'_{ia}, \tau'_{ib} \in \{0, 1\}^m, \boldsymbol{b}, \boldsymbol{e}_j, \boldsymbol{y}_j, \boldsymbol{x}_j, \boldsymbol{x}'_j \in \mathbb{Z}^m, \boldsymbol{f}_i, \boldsymbol{b}_j, \boldsymbol{b}'_j, \boldsymbol{b}''_j, \tau'_{ij} \in \mathbb{Z}_q^{n}$.

satisfying

$$\begin{aligned}&\boldsymbol{A}\boldsymbol{d}_{i1} + \boldsymbol{A}_1\boldsymbol{d}_{i2} + i\boldsymbol{A}_2\boldsymbol{d}_{i2} = \boldsymbol{u} + \boldsymbol{D}bin(\boldsymbol{w}_i) + \boldsymbol{f}_i\\ &\boldsymbol{w}_i = \boldsymbol{D}_0 bin(\boldsymbol{v}_i) + \boldsymbol{D}_1\boldsymbol{s}_i \mod q \;\; and \;\; \boldsymbol{v}_i = \boldsymbol{F}\boldsymbol{z}_i \mod q\\ &\boldsymbol{f}_i = \boldsymbol{C}\tau'_{ia} + \boldsymbol{C}_1\tau'_{ib} + i\boldsymbol{C}_2\tau'_{ib} \;\; and \;\; \tau'_i = H_1(\tau_i)\end{aligned} \quad (11)$$

$$\begin{aligned}&\boldsymbol{b}_j = \boldsymbol{C}\boldsymbol{x}_j + \boldsymbol{P}\boldsymbol{\tau}'_{ij}\\ &\boldsymbol{\tau}'_{ij} = \boldsymbol{P}'\boldsymbol{x}'_j \;\; and \;\; \boldsymbol{\tau}'_{ij} = H_2(\tau_{ij})\\ &\boldsymbol{b} = \boldsymbol{A}_3 grt_i + \boldsymbol{e}' \mod q \;\; and \;\; grt_i = \boldsymbol{A}\boldsymbol{d}_{i1}\end{aligned} \quad (12)$$

$$\begin{aligned}&\boldsymbol{b}'_j = \boldsymbol{B}^T\boldsymbol{P}'\boldsymbol{x}_j + \boldsymbol{y}_j \mod q \;\; and \;\; \boldsymbol{y}_j = \boldsymbol{x}_j + \boldsymbol{e}_j\\ &\boldsymbol{b}''_j = \boldsymbol{C}\boldsymbol{e}_j \mod q\\ &\boldsymbol{b}'''_j = \boldsymbol{C}_3\boldsymbol{y}_j \mod q\end{aligned} \quad (13)$$

An interactive zero-knowledge for the relation R(defined earlier) is already constructed by [23]. An interactive zero-knowledge protocol for the relation R' can be generated by transforming the relation R' to the relation R. The following section describes the steps to transform R' to R.

B. Transformation of R' to R

To transform the relation R' to R, we transform eq. (11-13) to the form $\boldsymbol{P}\boldsymbol{x} = \boldsymbol{y} \mod q$, and define a set VALID and permutation T such that eq. (8-10) is satisfied. Some sets and matrices which are used in the transformation are defined below:

1. B_{3m} is the set of all vectors in $\{-1, 0, 1\}^{3m}$ having equal number of $-1, 0, 1$. B_{2l} is the set of all vectors in $\{0, 1\}^{2l}$ having hamming weight l.
2. For any $\alpha > 0$, one can define the sequence $(\alpha_1, \alpha_2, \alpha_3, ..., \alpha_p)$ such that $\sum_{i=1}^{p}\alpha_i = \alpha$, where $p = \lfloor \log \beta \rfloor + 1$ [18]. We define a matrix $\boldsymbol{H}_{m,\alpha} = [\alpha_1, \alpha_2, \alpha_3, ..., \alpha_p] \otimes \boldsymbol{I}_m \in \mathbb{Z}^{m\times mp}$. Define another matrix $\boldsymbol{H}^*_{m,\alpha} \in \mathbb{Z}^{m\times 3mp}$ which is obtained by adding $2mp$ columns to $\boldsymbol{H}_{m,\alpha}$.
3. We define the matrix $\boldsymbol{R}_1$ as $\boldsymbol{R}_1 = \boldsymbol{I}_{4n} \otimes [1|2|4|....|2^{\lceil \log q\rceil - 1}]$ and $\boldsymbol{R}_2$ as $\boldsymbol{R}_2 = \boldsymbol{I}_{2n} \otimes [1|2|4|....|2^{\lceil \log q\rceil - 1}]$.

We extensively use the following lemma defined in [33] for transformation of R to R' .

Lemma 6. *Let m, O be positive integers and $\delta_O = \lfloor \log O \rfloor + 1$. On input a vector $\boldsymbol{v} \in [-O, O]^m$, extension and decomposition technique outputs a vector $\boldsymbol{v}^* \in B_{3m\delta_O}$ such that $\boldsymbol{H}^*_{m,O}\boldsymbol{v}^* = \boldsymbol{v}$.*

Conversion of all the equations in Definition 8 into $\boldsymbol{P}\boldsymbol{x} = \boldsymbol{y} \mod q$ proceeds as follows:

Transformation of eq. (11) to the appropriate form: Let $id \in \{0, 1\}^l$ be the binary representation of i and id_j represents the j-th bit of id. Let $\boldsymbol{y}_1 = bin(\boldsymbol{v}_i) \in \{0, 1\}^{2m}$ and $\boldsymbol{y}_2 = bin(\boldsymbol{w}_i) \in \{0, 1\}^m$. Equation (11) can be written as

$$\begin{aligned}&\boldsymbol{A}\boldsymbol{d}_{i1} + \boldsymbol{A}_1\boldsymbol{d}_{i2} + \sum_{i=1}^{l}(2^{l-i}\boldsymbol{A}_2)\; id_i\boldsymbol{d}_{i2} - \boldsymbol{D}\boldsymbol{y}_2 -\\ &(\boldsymbol{C}\tau'_{ia} + \boldsymbol{C}_1\tau'_{ib} + \sum_{i=1}^{l}(2^{l-i}\boldsymbol{C}_2)\;\tau'_{ib}) = \boldsymbol{u}\end{aligned} \quad (14)$$

$$\begin{aligned}&\boldsymbol{D}_0\boldsymbol{y}_1 + \boldsymbol{D}_1\boldsymbol{s}_i - \boldsymbol{R}_2\boldsymbol{y}_2 \mod q = 0 \;\; \text{and}\\ &\boldsymbol{R}_1\boldsymbol{y}_1 - \boldsymbol{F}\boldsymbol{z}_i = 0 \mod q\end{aligned} \quad (15)$$

Apply Lemma 6 to the vectors $\boldsymbol{d}_{i1}$, $\boldsymbol{d}_{i2} \in \mathbb{Z}^m$ to generate the vectors $\boldsymbol{d}^*_1, \boldsymbol{d}^*_2 \in B_{3m\delta\beta}$ respectively. Extend

$\tau'_{ia}, \tau'_{ib} \in \{0,1\}^m$ to $\hat{\tau_1}, \hat{\tau_2} \in \mathrm{B}_{2m}$ respectively. Extend the vector $\boldsymbol{y}_2 \in \{0,1\}^m$ to obtain $\hat{\boldsymbol{y}}_2 \in \mathrm{B}_{2m}$. Extend the identity $id \in \{0,1\}^l$ to $\hat{id} \in \mathrm{B}_{2l}$. Now, eq. (14) is reduced to

$$\boldsymbol{A}^*\boldsymbol{x}_{11} = \boldsymbol{u} \mod q \tag{16}$$

where,
$\boldsymbol{A}^* = [\boldsymbol{AH}^*_{m,\beta}|\boldsymbol{A}_1\boldsymbol{H}^*_{m,\beta}|2^{l-1}\boldsymbol{A}_2\boldsymbol{H}^*_{m,\beta}|....|2^0\boldsymbol{A}_2\boldsymbol{H}^*_{m,\beta}|(-\boldsymbol{D})|$ $0^{n\times m}|(-\boldsymbol{C})|0^{n\times m}|\boldsymbol{C}_1|0^{n\times m}|2^{l-1}\boldsymbol{C}_2|0^{n\times m}|....|2^0\boldsymbol{C}_2|0^{n\times m}]$
and,
$\boldsymbol{x}_{11} = [\boldsymbol{d}^*_1||\boldsymbol{d}^*_2||\hat{id}[1]\boldsymbol{d}^*_2||....||\hat{id}[2l]\boldsymbol{d}^*_2||\hat{\boldsymbol{y}}_2||\hat{\tau_1}||\hat{\tau_2}||$ $\hat{id}[1]\hat{\tau_2}||....||\hat{id}[2l]\hat{\tau_2}]$.

Similarly, eq. (15) is reduced to

$$\boldsymbol{K}\boldsymbol{x}_{12} = 0 \mod q \tag{17}$$

where

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}_1 & 0 \\ 0 & \boldsymbol{K}_2 \end{pmatrix} \text{ and } \boldsymbol{x}_{12} = \begin{pmatrix} \boldsymbol{t}_1 \\ \boldsymbol{t}_2 \end{pmatrix}$$

and
$\boldsymbol{K}_1 = [\boldsymbol{D}_0|0^{2n\times 2m}|\boldsymbol{D}_1\boldsymbol{H}^*_{2m,\beta}|(-\boldsymbol{R}_2)|0^{2n\times m}]$,
$\boldsymbol{K}_2 = [\boldsymbol{R}_1|0^{4n\times 2m}|\boldsymbol{FH}^*_{4m,\beta}]$,
$\boldsymbol{t}_1 = [\hat{\boldsymbol{y}}_1||\boldsymbol{s}^*_i||\hat{\boldsymbol{y}}_2]$ and $\boldsymbol{t}_2 = [\hat{\boldsymbol{y}}_1||\boldsymbol{z}^*_i]$.

The vectors $\boldsymbol{s}^*_i \in B_{3(2m)\delta\beta}$ and $\boldsymbol{z}^*_i \in B_{3(4m)\delta\beta}$ are obtained by applying Lemma (6) to $\boldsymbol{s}_i \in \mathbb{Z}^{2m}$ and $\boldsymbol{z}_i \in \mathbb{Z}^{4m}$ respectively and $\hat{\boldsymbol{y}}_1$ is obtained by extending $\boldsymbol{y}_1$ such that $\hat{\boldsymbol{y}}_1 \in \mathrm{B}_{4m}$.
We combine eq. (16) and eq. (17) and obtain to obtain $\boldsymbol{P}^*_1\boldsymbol{x}^*_1 = \boldsymbol{z}_1 \mod q$ where

$$\boldsymbol{P}^*_1 = \begin{pmatrix} \boldsymbol{A}^* & 0 \\ 0 & \boldsymbol{K} \end{pmatrix} \boldsymbol{x}^*_1 = \begin{pmatrix} \boldsymbol{x}_{11} \\ \boldsymbol{x}_{12} \end{pmatrix} \text{ and } \boldsymbol{z}_1 = \begin{pmatrix} \boldsymbol{u} \\ 0 \end{pmatrix} \tag{18}$$

Transformation of eq. (12) to the required form: Equation eq. (13) can be written as,

$$\boldsymbol{b}_j = \boldsymbol{C}\boldsymbol{x}_j + \boldsymbol{P}\boldsymbol{P}'\boldsymbol{x}'_j \tag{19}$$

$$\boldsymbol{b} = \boldsymbol{A}_3\boldsymbol{A}\boldsymbol{d}_{i1} + \boldsymbol{e}' \mod q \tag{20}$$

Apply Lemma 6 to the vectors $\boldsymbol{x}_j, \boldsymbol{x}'_j \in \mathbb{Z}^m$ to generate vectors $\boldsymbol{x}^*_j, \boldsymbol{x}'^*_j \in B_{3m\delta\beta}$. Now equation 19 reduces to,

$$\boldsymbol{K}_3\boldsymbol{t}_3 = \boldsymbol{b}_j \tag{21}$$

where, $\boldsymbol{K}_3 = [\boldsymbol{CH}^*_{m,\beta}|\boldsymbol{PP}'\boldsymbol{H}^*_{m,\beta}]$ and $\boldsymbol{t}_3 = [\boldsymbol{x}^*_j||\boldsymbol{x}'^*_j]$
Similarly apply Lemma 6 to the vectors $\boldsymbol{e}' \in \mathbb{Z}^m$ and to generate vector $\boldsymbol{e}'^*_j \in B_{3m\delta\beta}$. Now equation 20 reduces to,

$$\boldsymbol{K}_4\boldsymbol{t}_4 = \boldsymbol{b} \tag{22}$$

where, $\boldsymbol{K}_4 = [\boldsymbol{A}_3\boldsymbol{AH}^*_{m,\beta}|\boldsymbol{I}_m\boldsymbol{H}^*_{m,\beta}]$ and $\boldsymbol{t}_4 = [\boldsymbol{d}^*_1||\boldsymbol{e}^*]$
We combine eq. (21) and eq. (22) and obtain to obtain $\boldsymbol{P}^*_2\boldsymbol{x}^*_2 = \boldsymbol{z}_2 \mod q$ where

$$\boldsymbol{P}^*_2 = \begin{pmatrix} \boldsymbol{K}_3 & 0 \\ 0 & \boldsymbol{K}_4 \end{pmatrix} \boldsymbol{x}^*_2 = \begin{pmatrix} \boldsymbol{t}_3 \\ \boldsymbol{t}_4 \end{pmatrix} \text{ and } \boldsymbol{z}_2 = \begin{pmatrix} \boldsymbol{b}_j \\ \boldsymbol{b} \end{pmatrix} \tag{23}$$

Transformation of eq. (13) to the required form:

Apply Lemma 6 to the vectors $\boldsymbol{y}_j, \boldsymbol{e}_j \in \mathbb{Z}^m$ to generate the vectors $\boldsymbol{y}^*_j, \boldsymbol{e}^*_j \in B_{3m\delta\beta}$. Now equation 13 reduces to,

$$\boldsymbol{K}_5\boldsymbol{t}_5 = \boldsymbol{b}'_j \tag{24}$$

$$\boldsymbol{K}_6\boldsymbol{t}_6 = \boldsymbol{b}''_j \tag{25}$$

$$\boldsymbol{K}_7\boldsymbol{t}_7 = \boldsymbol{b}'''_j \tag{26}$$

where, $\boldsymbol{K}_5 = [\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}'\boldsymbol{H}^*_{m,\beta}|\boldsymbol{I}_m\boldsymbol{H}^*_{m,\beta}]$ and $\boldsymbol{t}_5 = [\boldsymbol{x}^*_j||\boldsymbol{y}^*_j]$
$\boldsymbol{K}_6 = \boldsymbol{CH}^*_{m,\beta}$ and $\boldsymbol{t}_6 = \boldsymbol{x}^*_j$
$\boldsymbol{K}_7 = \boldsymbol{DH}^*_{m,\beta}$ and $\boldsymbol{t}_7 = \boldsymbol{y}^*_j$

We combine eq. (24), (25) and eq. (26) and obtain to obtain $\boldsymbol{P}^*_3\boldsymbol{x}^*_3 = \boldsymbol{z}_3 \mod q$ where

$$\boldsymbol{P}^*_3 = \begin{pmatrix} \boldsymbol{K}_5 & 0 & 0 \\ 0 & \boldsymbol{K}_6 & 0 \\ 0 & 0 & \boldsymbol{K}_7 \end{pmatrix} \boldsymbol{x}^*_3 = \begin{pmatrix} \boldsymbol{t}_5 \\ \boldsymbol{t}_6 \\ \boldsymbol{t}_7 \end{pmatrix} \text{ and } \boldsymbol{z}_3 = \begin{pmatrix} \boldsymbol{b}'_j \\ \boldsymbol{b}''_j \\ \boldsymbol{b}'''_j \end{pmatrix} \tag{27}$$

Finally we combine equations (18), (23) and (27) to obtain $\boldsymbol{Px} = \boldsymbol{y} \mod q$

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}^*_1 & 0 & 0 \\ 0 & \boldsymbol{P}^*_2 & 0 \\ 0 & 0 & \boldsymbol{P}^*_3 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}^*_1 \\ \boldsymbol{x}^*_2 \\ \boldsymbol{x}^*_3 \end{pmatrix} \text{ and } \boldsymbol{y} = \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \\ \boldsymbol{z}_3 \end{pmatrix}$$

Thus, all the equations in the relation R' are transformed to the form $\boldsymbol{Px} = \boldsymbol{y} \mod q$.

Let $L = 8m + (4l+12)3m\delta_\beta$. We define a set VALID as follows:
VALID: Set of all vectors $\{-1,0,1\}^L$ of the form

$$\boldsymbol{h} = [\boldsymbol{h}_1||\boldsymbol{h}_2||\boldsymbol{w}_1\boldsymbol{h}_2||\ldots||\boldsymbol{w}_{2l}\boldsymbol{h}_2||\boldsymbol{h}_3||\boldsymbol{h}_4||\boldsymbol{h}_5||\boldsymbol{w}_1\boldsymbol{h}_5||\ldots||\boldsymbol{w}_{2l}\boldsymbol{h}_5|| \\ \boldsymbol{h}_6||\boldsymbol{h}_7||\boldsymbol{h}_3||\boldsymbol{h}_6||\boldsymbol{h}_8||\boldsymbol{h}_9||\boldsymbol{h}_{10}||\boldsymbol{h}_1||\boldsymbol{h}_{11}||\boldsymbol{h}_9||\boldsymbol{h}_{12}||\boldsymbol{h}_9||\boldsymbol{h}_{12}]$$

where $\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_9, \boldsymbol{h}_{10}, \boldsymbol{h}_{11}, \boldsymbol{h}_{12} \in \mathrm{B}_{3m\delta_\beta}, \boldsymbol{h}_7 \in \mathrm{B}_{3(2m)\delta_\beta}, \boldsymbol{h}_8 \in \mathrm{B}_{3(4m)\delta_\beta}, \boldsymbol{h}_3, \boldsymbol{h}_4, \boldsymbol{h}_5, \boldsymbol{h}_6 \in \mathrm{B}_{2m}$, $\boldsymbol{w}=[\boldsymbol{w}_1\boldsymbol{w}_2...\boldsymbol{w}_{2l}]\in\mathrm{B}_{2l}$.

Let $\mathrm{S} = \mathrm{S}_{3m\delta_\beta} \times \mathrm{S}_{3m\delta_\beta} \times \mathrm{S}_{3m\delta_\beta} \times \mathrm{S}_{3m\delta_\beta} \times \mathrm{S}_{3m\delta_\beta} \times \mathrm{S}_{3m\delta_\beta} \times \mathrm{S}_{3(2m)\delta_\beta} \times \mathrm{S}_{3(4m)\delta_\beta} \times \mathrm{S}_{2l} \times \mathrm{S}_{2m} \times \mathrm{S}_{2m} \times \mathrm{S}_{2m} \times \mathrm{S}_{2m}$

Let $\pi = (\rho, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6, \pi_7, \pi_8, \pi_9, \pi_{10}, \pi_{11}, \pi_{12}) \in \mathrm{S}$.
Define the permutation T as

$$\mathrm{T}(\mathbf{h}) = [\pi_1(\boldsymbol{h}_1)||\pi_2(\boldsymbol{h}_2)||w_{\rho(1)}(\pi_2(\boldsymbol{h}_2))||....||w_{\rho(2l)}(\pi_2(\boldsymbol{h}_2))||\pi_3(\boldsymbol{h}_3)|| \\ \pi_4(\boldsymbol{h}_4)||\pi_5(\boldsymbol{g}_5)||w_{\rho(1)}(\pi_5(\boldsymbol{h}_5))||....||w_{\rho(2l)}(\pi_5(\boldsymbol{h}_5))|| \\ \pi_6(\boldsymbol{h}_6)||\pi_7(\boldsymbol{h}_7)||\pi_3(\boldsymbol{h}_3)||\pi_6(\boldsymbol{h}_6)||\pi_8(\boldsymbol{h}_8)||\pi_9(\boldsymbol{h}_9)||\pi_{10}(\boldsymbol{h}_{10})|| \\ \pi_1(\boldsymbol{h}_1)||\pi_{11}(\boldsymbol{h}_{11})||\pi_9(\boldsymbol{h}_9)||\pi_{12}(\boldsymbol{h}_{12})||\pi_9(\boldsymbol{h}_9)||\pi_{12}(\boldsymbol{h}_{12})]$$

We observe that $\boldsymbol{x}$ belongs to the set VALID and eq. (8-10) is satisfied. Therefore, the relation R' (given in Definition 8) is transformed to the relation R. Thus, an interactive zero-knowledge protocol for R' is obtained directly from the protocol for the relation R, described in [23].