A Lattice-based Accountable Subgroup Multi-signature Scheme with Verifiable Group Setup

Ahmet Ramazan Ağırtaş

Oğuz Yayla

Institute of Applied Mathematics,
Middle East Technical University,
06800, Çankaya, Ankara, Turkey
{*agirtas.ramazan, oguz*}*@metu.edu.tr*

December 12, 2023

Abstract

An accountable subgroup multi-signature (ASM) is a multi-signature that allows any subgroup of potential signers to jointly sign a message such that the subgroup of co-signers are accountable for the resulting signature and their identities are identifiable to any verifier. In this paper, we propose a novel lattice-based accountable subgroup multi-signature scheme, i.e., vMS_2, by combining the group setup method of recently proposed vASM scheme and Damgård et al.'s lattice-based MS_2 multi-signature scheme. Key generation, signature generation and verification phases of our proposed scheme are almost identical to the MS_2 scheme. In the group setup phase, we generate membership keys which is used for signing a message on behalf of a group $\mathcal{G}$ of users. These membership keys are generated via a joint verifiable secret sharing (VSS) scheme in a way that they include a piece of information from the secret keys of all users in $\mathcal{G}$ so that any subgroup of users in $\mathcal{G}$ having a valid membership key can sign in an accountable fashion. We also present a comparison of the underlying MS_2 scheme and our accountable subgroup multi-signature scheme vMS_2 to show the cost of accountability. We see that lattice-based accountable subgroup multi-signature scheme can be achieved by adding a one-time one-round group setup whose cost is slightly higher than signature generation and verification of the underlying MS_2 signature scheme.

Keywords: lattice-based multi-signatures, accountable subgroup multi-signatures, lattice-based accountable subgroup multi-signatures

1 Introduction

An accountable subgroup multi-signature (ASM) is a variant of multi-signatures that allows a subgroup of the signers to jointly sign a message with the property that the subgroup of co-signers are accountable for the resulting signature and their identities are identifiable to any verifier. The notion of accountable subgroup multi-signature is firstly introduced by Micali et al. [1] in 2001. In 2018, Boneh et al. proposed a pairing-based ASM scheme in [2]. It was based on BLS signature scheme [3]. Recently, another pairing-based ASM scheme with a verifiable group setup, i.e. vASM, and two more modified versions of the Boneh

et al.'s ASM scheme, i.e. ASMwSA and ASMwCA, were proposed by the authors in [4]. The authors also provided a detailed comparison between the existing pairing-based ASM schemes [2, 4]. The schemes that were proposed in [4] requires less pairing operations in the verification phase and shorter signature and membership key size than the ASM scheme proposed in [2].

The common structure of the existing pairing-based ASM schemes is as follows. Assume that $\mathcal{G}$ is a group of n users, and $\mathcal{S} \subseteq \mathcal{G}$ is a subgroup of τ signers among n. Users in $\mathcal{G}$ generate their secret and public key pairs individually. Then they participate in an interactive one-time group setup phase. After the group setup phase each user obtains her membership key. Then the users in $\mathcal{S}$ sign a common message m with her secret and/or membership keys. A combiner, who can be one of the signers or a designated third party, aggregates the individual signatures and outputs the accountable subgroup multi-signature of the subgroup $\mathcal{S}$. Given the message, the signature and the information $\mathcal{S}$, any verifier can verify whether the signature is valid or not.

The main difference among the existing pairing-based ASM schemes is their group setup phase. In [2], authors utilize another multi signature scheme to generate the users' membership keys, and authenticate each other to show that they are authorized to sign on behalf of the group $\mathcal{G}$. On the other hand, authors in [4] use verifiable secret sharing scheme for the same purposes. They share their secret keys via a joint verifiable secret sharing scheme to authorize other users in group $\mathcal{G}$.

We simply ask the following question:

> *If we apply the same group setup method to another multi-signature, can we have an accountable subgroup multi-signature with another assumption?*

We focus on the multi-signatures based on Dilithium [5] which is based on module-LWE and module-SIS problems and is one of the winner signature schemes of the NIST's (National Institute of Standards and Technology) post-quantum cryptography standardization process. We use Damgård et al.'s lattice-based MS_2 multi-signature scheme [6] which is based on the Dilithium scheme. In [6], authors propose one 3-round (DS_3) and one 2-round (DS_2) n-out-of-n multi-signature schemes, one 2-round multi-signature scheme (MS_2), and a trapdoor commitment scheme.

In this paper, we propose a novel lattice-based accountable subgroup multi-signature scheme by combining the group setup method of vASM scheme [4] and Damgård et al.'s lattice-based MS_2 multi-signature scheme [6]. We follow the same method in [4] for the group setup phase in our construction. Each user in the group $\mathcal{G}$ share her secret key via a VSS, and upon receiving other shares from other users, each user computes her membership keys by simply summing up the shares she received. However, summing the shares entails a problem in the size of the membership keys since the size (infinity norm, see Section 2.1) of the signing keys are crucial in the underlying MS_2 scheme. To tackle this issue, we allow users in $\mathcal{G}$ to sample a normalizer vector from a special distribution to ensure that the resulting membership keys are of the desired size. We should also note that adding accountability is not a cost-free operation, it comes with a cost of one-time one-round group setup phase.

In the following sections, first we give the assumptions, notation and a few definitions. Then we restate the vASM, Dilithium-G and MS_2 schemes. Finally we describe our proposed vMS_2 scheme, discuss its security and give a computational comparison with the underlying MS_2 scheme.

2 Preliminary

In this section we give the notation, assumptions, definitions of underlying hard problems, multi-signatures, accountable subgroup multi-signatures, and the protocols used in our construction.

2.1 Assumptions and the notation

Assume that $R = \mathbb{Z}[x]/(f(x))$ and $R_q = \mathbb{Z}_q[x]/(f(x))$ where N is a power of two and $f(x) = x^N + 1$ is the $2N$-th cyclotomic polynomial as in [6, 5]. As common in lattice-based cryptography, through out this report we use *centered reduction* $\bmod^{\pm} q$, i.e. for any $a \in \mathbb{Z}_q$, $\bar{a} = a \bmod^{\pm} q$ is defined to be a unique integer in the range $[-\frac{q-1}{2}, \frac{q-1}{2}]$. We write the ring elements with lower-case letters and column vector of elements with bold ones, i.e. $u \in R$ and $\mathbf{u} \in R^k$, respectively. For a set X we write $x \xleftarrow{\$} X$ to show that x is sampled from the uniform distribution defined over the set X.

For any $u(x) = u_0 + u_1 x + \ldots + u_{N-1}x^{N-1} \in R_q$, ℓ^∞ norm is defined to be $||u||_\infty = \max_i |u_i|$ for $i = 0, \ldots, N-1$, and ℓ^2 norm of u is defined to be $||u||_2 = \sqrt{u_0^2 + \ldots + u_{N-1}^2}$. Similarly for $\mathbf{u} \in R_q^k$, $||\mathbf{u}||_\infty = \max_i ||u_i||_\infty$, and ℓ^2 norm of $\mathbf{u}$ is defined to be

$$||\mathbf{u}||_2 = \sqrt{(||u_1||_2)^2 + \ldots + (||u_k||_2)^2}.$$

Let $S_\eta \subseteq R$ be the set of small polynomials, i.e. for $\eta \in \mathbb{Z}$, $S_\eta = \{v \in R : ||v||_\infty \leq \eta\}$. Let $C \subseteq R$ be the challenge space consisting of small polynomials, which will be used as the image of random oracle $\mathrm{H}_0 : \{0,1\}^* \to C$, i.e. for $\kappa \in \mathbb{Z}$,

$$C = \{c \in R : ||c||_\infty = 1 \wedge ||c||_1 = \kappa\},$$

where $||c||_1$ is the ℓ^1 norm and it is equal to the sum of all the coefficients of the polynomial c. For constructing such a random oracle, a variant of Fisher-Yates shuffle [7] is used in Dilithium and Dilithium-G ([5, Algorithm 1]). The randomness, which is produced by a collision resistant hash function, is given to the random oracle as input (randomness seed).

We now give the definitions of Module-SIS and Module-LWE problems, which are assumed to be hard in our designs. We begin with the definition of discrete Gaussian distribution.

Definition 2.1 (Discrete Gaussian Distribution over R^m [6])**.** For $x \in R^m$, let

$$\rho_{v,s}(x) = exp(-\pi||x - v||_2^2/s^2)$$

be a Gaussian function of parameters $v \in R^m$ and $s \in R$. The discrete Gaussian distribution $D_{v,s}^m$ centered at v is

$$D_{v,s}^m(x) = \rho_{v,s}(x)/\rho_{v,s}(R^m),$$

where $\rho_{v,s}(R^m) = \sum_{x \in R^m} \rho_{v,s}(x)$.

Definition 2.2 (Module-SIS$_{q,k,\ell,\beta}$ Problem [6])**.** Given a random matrix $\mathbf{A} \in R_q^{k \times \ell}$, find a vector $\mathbf{x} \in R_q^{\ell+k}$, such that

$$[\mathbf{A}|\mathbf{I}] \cdot \mathbf{x} = \mathbf{0} \text{ and } ||\mathbf{x}||_2 \leq \beta.$$

Definition 2.3 (Module-LWE$_{q,k,\ell,\eta}$ Problem [6])**.** Given a pair $(\mathbf{A}, \mathbf{t}) \in R_q^{k \times \ell} \times R_q^k$, decide whether it is selected uniformly at random from $R_q^{k \times \ell} \times R_q^k$ or it is generated in a way that $t := [\mathbf{A}|\mathbf{I}] \cdot \mathbf{s}$ for some $\mathbf{s} \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$.

2.2 Multi-signatures and Accountable Subgroup Multi-signatures

Definition 2.4. A multi-signature scheme consists of four algorithms, i.e. *ParGen, KeyGen, Sign, and Verify.* Let $\mathcal{G} = \{P_1, \ldots, P_n\}$ be a set of n players.

- **ParGen(1^λ)** takes the security parameter λ as input, and outputs the public system parameters *par* including security parameter, hash functions, cyclic groups, generators, etc.

- **KeyGen**(par) takes the system parameters par as input, and outputs secret and public key pair, i.e. sk and pk.
- **Sign**(par, sk, m) is an interactive protocol which is run by $\mathcal{G}$, in two steps, as follows:
  - **Individual signature generation** takes the system parameters par, secret key sk_i and message m as inputs, and outputs the individual signature σ_i.
  - **Individual signature aggregation** takes a set of individual signatures $\{\sigma_i\}_{i\in\mathcal{G}}$ as inputs and outputs the multi-signature σ.
- **Verify**($par, \{pk_j\}_{j\in\mathcal{G}}, \sigma, m$) takes system parameter par, multi-signature σ, message m, and public keys of the players in $\mathcal{G}$ as inputs, and outputs 1 if it is valid or 0 otherwise.

For the definition of an accountable subgroup multi-signature scheme, we add an interactive group setup algorithm **GSetup**, which is a one-time protocol run by all the players in the group $\mathcal{G}$. Then we modify the **Sign** and **Verify** algorithms as follows.

Definition 2.5. An accountable subgroup multi-signature scheme is a tuple of five algorithms, that is *ParGen, KeyGen, GSetup, Sign, and Verify.* Let $\mathcal{G} = \{P_1, \ldots, P_n\}$ be a set of n players, and $\mathcal{PK} = \{pk_1, \ldots, pk_n\}$ is the set of public keys of all the users in group $\mathcal{G}$.

- **ParGen**(1^λ) takes the security parameter λ as input, and outputs the public system parameters par including security parameter, hash functions, cyclic groups, generators, etc.
- **KeyGen**(par) takes the system parameters par as input, and outputs secret and public key pair, i.e. sk and pk.
- **GSetup**($par, sk_i, \mathcal{PK}$) is an interactive protocol which is run by all the players in $\mathcal{G}$. It takes system parameters par, secret keys sk_i and set of all public keys $\mathcal{PK}$, and outputs a membership key mk_i, a set of membership public keys MPK, and a set of commitments COM.
- **Sign**(par, mk_i, m) is an interactive protocol which is run by any subset $\mathcal{S} \subseteq \mathcal{G}$, in two steps, as follows:
  - **Individual signature generation** takes the system parameters par, membership key mk_i and message m as inputs, and outputs the individual signature σ_i.
  - **Individual signature aggregation** takes a set of individual signatures $\{\sigma_i\}_{i\in\mathcal{S}}$ as inputs and outputs the accountable subgroup multi-signature σ.
- **Verify**($par, \mathrm{MPK}, \mathrm{COM}, \mathcal{S}, \sigma, m$) takes system parameter par, multi-signature σ, message m, definition of the subset $\mathcal{S}$, the set of membership public keys MPK, and the commitment set COM as inputs, and outputs 1 if it is valid or 0 otherwise.

Correctness and *unforgeability* are two properties that every accountable subgroup multi-signature scheme should meet. Correctness means that for any subgroup of signers $\mathcal{S} \subseteq \mathcal{G}$ and message m, if the signers $P_i \in \mathcal{S}$ run the **Sign**($\cdot$) protocol with their membership keys mk_i, and follow the protocol honestly, then all of the signers in $\mathcal{S}$ outputs exactly the same valid accountable subgroup multi-signature σ, such that **Verify**($par, \mathrm{MPK}, \mathrm{COM}, \mathcal{S}, \sigma, m$) $= 1$. Unforgeability means that it is infeasible for an adversary to forge a valid multi-signature where at least one honest user follows the protocol properly. Unforgeability can be described by the following game.

Setup: The challenger randomly picks n values and computes membership public keys MPK and the commitment set COM, with respect to the indices $\{1, \ldots, n\}$. Finally it runs the adversary $\mathcal{A}(par, \mathrm{MPK}, \mathrm{COM})$, where par is the system parameters.

Signature queries: The adversary $\mathcal{A}$ makes queries on any message m, for any subset $\mathcal{S} \subseteq \{1, \ldots, n\}$ of users, and challenger responds with valid signatures.

Output: The adversary $\mathcal{A}$ eventually outputs a subset of indices $\mathcal{S}$, a message m, and an accountable subgroup multi-signature σ. The adversary $\mathcal{A}$ wins the game if $\textbf{Verify}(par, \text{MPK}, \text{COM}, \mathcal{S}, \sigma, m) = 1$, where the message m has been never queried as part of a signing query before.

2.3 Feldman's VSS Protocol

Feldman's verifiable secret sharing (VSS) scheme [8] is a protocol which is used for sharing a secret among predetermined users in a verifiable fashion, where Shamir's secret sharing scheme [9] was directly used to share and reconstruct the secret. In addition to Shamir's scheme, the shares can be checked for consistency in Feldman's scheme, for which the dealer computes commitments to the coefficients of the secret polynomial. By this way, users can verify that they receive the consistent shares from the dealer.

Assume that we have n players. Let $\mathbb{F}_q$ be the finite field with prime order q and g be a primitive element in $\mathbb{F}_q$. The dealer shares a secret as follows:

- Choose a polynomial of degree $t - 1 < q$,

$$f(x) = \alpha_{t-1}x^{t-1} + \ldots + \alpha_1 x + \alpha_0$$

  with random $\alpha_k \in \mathbb{F}_q^*$ for $k = 1, \ldots, t-1$, and α_0 is the secret to be shared.

- Compute a set of commitments $\text{COM} = \{C_k : C_k = g^{\alpha_k}, k = 0, 1, \ldots, t-1\}$.

- Send $f(i)$ and COM to the i-th player for $i = 1, 2, \ldots, n$.

After receiving a share and the set of commitments, the i-th player checks

$$g^{f(i)} \stackrel{?}{=} \prod_{k=0}^{t-1} C_k^{i^k}. \tag{2.1}$$

The received share is consistent with the shared secret only if (2.1) is satisfied. If at least any t or more players perform Lagrange interpolation with their shares, they can uniquely determine the secret polynomial and $f(0)$ will yield the secret.

Note that Shamir's secret sharing scheme is an information theoretically secure protocol. Although VSS protocol uses Shamir's secret sharing scheme, it does not have the same level of security. Since the users commit to the secret polynomials, the security is defined by the underlying assumption (e.g. discrete logarithm assumption) of the commitment method. See [10] and [11] for the general security discussion of Shamir's SSS.

In Section 2.4 we use this protocol as an implicit authentication and a proof of possession method. We use only sharing, committing and verifying phases of this protocol.

2.4 vASM: An ASM scheme with VSS based group setup

The vASM scheme [4] is a pairing-based accountable subgroup multi-signature scheme which is based on the BLS signature scheme [3]. In vASM scheme, each user generates his secret and public key pair independently. Then all users jointly perform a group setup in which they participate in a VSS protocol. At the end of this procedure, each user obtains his membership key, which satisfies a common public commitment set, and his membership public key. Any number of users can sign the message with their membership keys and aggregate the individual signatures. Given the message, the signature, the commitment set, and the information of the subgroup of signers, any verifier can verify the signature.

Consider a set $\mathcal{G}$ of n signers. Assume that the users of a subgroup $\mathcal{S} \subseteq \mathcal{G}$ want to sign a common message m. Assume that $\mathrm{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing function which takes inputs from the cyclic additive groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and maps to the cyclic multiplicative group $\mathbb{G}_T$. $\mathrm{H} : \{0,1\}^* \to \mathbb{G}_1$ is a hash function which maps any binary inputs with arbitrary length onto group $\mathbb{G}_1$. We give the steps of the vASM scheme [4] below.

1. Key Generation: Each user $i \in \mathcal{G}$ picks a secret key $sk_i \xleftarrow{\$} \mathbb{Z}_q$, and computes the public key $pk_i \leftarrow g_2^{sk_i}$, where g_2 is a generator of $\mathbb{G}_2$.

2. Group Setup: Each user $i \in \mathcal{G}$ proceeds as follows:

   - Chooses a polynomial $f_i(x) = \alpha_{n-1}^{(i)} x^{n-1} + \ldots + \alpha_1^{(i)} x + \alpha_0^{(i)} \in \mathbb{Z}_q[x]$, where $\alpha_0^{(i)} = sk_i$ and $\alpha_k^{(i)}$'s are all nonzero and distinct, for $k = 1, \ldots, n-1$.
   - Computes the set of commitments $\mathrm{COM}_i := \{C_k^{(i)} = g_2^{\alpha_k^{(i)}} | k = 0, \ldots, n-1\}$.
   - Sends $(f_i(j), \mathrm{COM}_i)$ to j-th user in $\mathcal{G}$, for $j = 1, \ldots, n$.
   - After receiving $(f_j(i), \mathrm{COM}_j)$ from j-th user,
     - computes the membership key $mk_i = \sum_{j \in \mathcal{G}} f_j(i)$.
     - computes $\mathrm{COM} := \{C_k = \prod_{j \in \mathcal{G}} C_k^{(j)} | k = 0, \ldots, n-1\}$.
   - Checks:

     (a) $C_0 \stackrel{?}{=} \prod_{i \in \mathcal{G}} pk_i$

     (b) $g_2^{mk_i} \stackrel{?}{=} \prod_{k=0}^{n-1} C_k^{i^k}$
   - If either (a) or (b) fails, then she aborts. Else, she makes COM and set of membership public keys mpk_i's public. Define $\mathrm{MPK} = \{mpk_i\}_{i \in \mathcal{G}}$. Note that these public keys can also be computed by the verifier, i.e. $mpk_i = g_2^{mk_i} = \prod_{k=0}^{n-1} C_k^{i^k}$.

3. Signature Generation: A signer $i \in \mathcal{G}$ computes his/her individual signature $s_i = \mathrm{H}(m)^{mk_i}$ on the message m and sends s_i to the combiner.

4. Signature Aggregation: After receiving the individual signatures of the signers, the combiner first forms the set of signers $\mathcal{S} \subseteq \mathcal{G}$. Then, she computes the aggregated subgroup multi-signature $\sigma = \prod_{i \in \mathcal{S}} s_i$.

5. Verification: Anyone, who is given $\{par, \mathrm{MPK}, \mathcal{S}, m, \sigma\}$, can verify the signature σ by checking

$$\mathrm{e}\big(\mathrm{H}(m), \prod_{i \in \mathcal{S}} mpk_i\big) \stackrel{?}{=} \mathrm{e}(\sigma, g_2).$$

We will use the same method of generating membership keys of the users of the group $\mathcal{G}$ for constructing a novel lattice based accountable subgroup multi-signature scheme. To this end, in the rest of this section we will give the definitions of the Dilithium-G [5] and the MS_2 [6] schemes.

2.5 Dilithium-G

In 2022, the National Institute of Standards and Technology (NIST) announced the winner signature schemes for post-quantum cryptography standardization process. The CRYSTALS-Dilithium signature scheme [5] is one of the winner signature schemes which is based on module-LWE and module-SIS

problems. In the original paper, authors proposed two variants, Dilithium and Dilithium-G. The main difference between these variants is the distribution that the variants sample random elements from and the way of rejection sampling procedure. In the Dilithium scheme, the sampling procedure is done from a uniform distribution whereas the Dilithium-G uses a Gaussian one. The Dilithium-G scheme has better parameters in terms of size and number of repetitions. However, it is noted that it has weaknesses against side channel attacks [5].

Below we state a simplified version of Dilithium-G, and then we restate a few lemmas for the signature parameters (see Table 1) which were defined in [6].

Algorithm 1 Key Generation

Input: $par := \{R_q, k, \ell, \eta, B, s, M\}$

Output: (sk,pk)

1: $\mathbf{A} \xleftarrow{\$} R_q^{k \times \ell}$

2: $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (\ell+k)}$, where $\mathbf{I}$ is the $k \times k$ identity matrix

3: $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$, $\mathbf{s} := \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$

4: $\mathbf{t} := \bar{\mathbf{A}} \cdot \mathbf{s}$

5: $pk = (\mathbf{A}, \mathbf{t})$ and $sk = \mathbf{s}$

6: **return** (sk, pk)

Algorithm 2 Signature Generation

Input: $sk = \mathbf{s}, pk = (\mathbf{A}, \mathbf{t}), \mu, par := \{R_q, k, \ell, \eta, B, s, M\}$

Output: A signature pair $(\mathbf{z}, c)$

1: $(\mathbf{y}_1, \mathbf{y}_2) \xleftarrow{\$} D_s^\ell \times D_s^k$, $\mathbf{y} := \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$

2: $\mathbf{w} := \bar{\mathbf{A}} \cdot \mathbf{y}$

3: $c \leftarrow \mathrm{H}_0(\mathbf{w}, \mu, pk)$

4: $\mathbf{z} := c\mathbf{s} + \mathbf{y}$

5: With probability $\min(1, D_s^{\ell+k}(\mathbf{z})/(M \cdot D_{c\mathbf{s},s}^{\ell+k}(\mathbf{z})))$:

6: **return** $(\mathbf{z}, c)$

7: Restart otherwise

Algorithm 3 Verification

Input: $(\mathbf{z}, c), pk = (\mathbf{A}, \mathbf{t}), \mu, par := \{R_q, k, \ell, \eta, B, s, M\}$

Output: Accept or Reject

1: **if** $||\mathbf{z}||_2 \leq B$ and $c = \mathrm{H}_0(\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}, \mu, pk)$ **then**

2: **return** Accept

3: **end if**

4: Otherwise, **return** Reject

Aggregating individual signatures inevitably results in a larger size of multi-signature. According to Lemma 1 of [6], the upper bound for the size of the sum of n Gaussian samples is $\sqrt{n} \cdot B$, where B is the upper bound for only one sample and we restate it below.

Lemma 1. *(Sum of Discrete Gaussian Samples [6, Lemma 1]) Suppose s exceeds the smoothing parameter by a factor $\geq \sqrt{2}$. Let x_i for $i \in [n]$ be independent samples from the distribution D_s^m. Then the distribution of $x = \sum_i x_i$ is statistically close to $D_{s\sqrt{n}}^m$.*

According to [12] the upper bound for the size of a single signature is computed by $B = \gamma\sigma\sqrt{(\ell + k)N}$. For ensuring the size of a single signature stays below this bound with high probability, one should set $\gamma > 1$.

Lemma 2 ([6, Lemma 2] and [12, Lemma 4.4])**.** *For any $\gamma > 1$,*

$$Pr[||z||_2 > B = \gamma\sigma\sqrt{mN} : \boldsymbol{z} \xleftarrow{\$} D_s^m] < \gamma^{mN} e^{mN(1-\gamma^2)/2}.$$

The following lemma given in [12] determines both the standard deviation and the value M, i.e. the number of repetition of signing attempts until producing a valid signature. Note that setting $\alpha = 11$ and $t = 12$ results in $M = 3$ which provides $\epsilon < e^{-100}$, i.e. at least 100-bit security.

Lemma 3 ([6, Lemma 3], and [12, Lemma 4.5])**.** *For $\boldsymbol{V} \subseteq R^m$, let $T = max_{v \in \boldsymbol{V}}||v||_2$. Fix some t such that $t = \omega\sqrt{\log(mN)}$ and $t = o(\log(mN))$. If $\sigma = \alpha T$ for any positive α, then*

$$Pr[M > D_s^m(\boldsymbol{z})/D_{v,s}^m(\boldsymbol{z}) : \boldsymbol{z} \xleftarrow{\$} D_s^m(\boldsymbol{z})] \geq 1 - \epsilon$$

where $M = e^{t/\alpha + 1/(2\alpha^2)}$ and $\epsilon = 2e^{-t^2/2}$.

The proofs of the above lemmas can be found in [12].

2.6 MS_2 Scheme

In [6], authors proposed 3 novel multi-signature schemes and a trapdoor commitment scheme. One of those multi-signature schemes is 3-round n-out-of-n multi-signature scheme (DS_3) that utilizes Baum et al.'s additively homomorphic commitment scheme [13]. The second one is a 2-round version of the first one (DS_2). The only difference is that they use a novel additively homomorphic trapdoor commitment scheme which is indeed a modified version of Baum et al.'s commitment scheme. Moreover, a 2-round multi-signature scheme called MS_2 was proposed in the same paper which is also constructed by the help of the above mentioned trapdoor commitment scheme (TCOM). Below we restate the 2-round MS_2 scheme proposed by Damgård et al. in [6].

1. **Key Generation:** Key generation phase is identical to the **Algorithm 1**. For choosing the matrix $\mathbf{A} \in R_q^{k \times \ell}$, a trusted third party can do it or users can jointly compute it interactively.
2. **Signature Generation:** Let $\mathrm{H}_0 : \{0,1\}^* \to C$ and $\mathrm{H}_3 : \{0,1\}^* \to S_{ck}$ be two random oracles, where S_{ck} is the commitment key space as defined in [6]. Let the functions Commit and Open belong to the additively homomorphic trapdoor commitment scheme TCOM which was proposed by Damgård et al. in [6].
   (a) Compute the message-specific commitment key $ck \leftarrow \mathrm{H}_3(\mu, L)$, where μ is the message and $L = \{pk_1, \ldots, pk_n\}$.
   (b) Pick a random $\mathbf{y}_i \xleftarrow{\$} D_s^{\ell+k}$ and compute $\mathbf{w}_i = \bar{\mathbf{A}}\mathbf{y}_i$, where $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (\ell+k)}$.
   (c) Compute $com_i \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_i, r_i)$, where r_i is a discrete Gaussian vector sampled from the Gaussian distribution $D(S_r)$ defined in the trapdoor commitment scheme in [6].
   (d) Send com_i to the other users.
   (e) Upon receiving com_j for $i \neq j$, compute $com := \sum_{j \in \mathcal{G}} com_j$
   (f) Compute $c_i \leftarrow \mathrm{H}_0(\mathbf{t}_i, com, \mu, L)$
   (g) Compute his individual signature $\mathbf{z}_i = c_i\mathbf{s}_i + \mathbf{y}_i$

(h) Run the rejection sampling on input $(c_i\mathbf{s}_i, \mathbf{z}_i)$, i.e. with probability

$$min(1, D_s^{\ell+k}(\mathbf{z}_i)/(M \cdot D_{c_i s_i, s}^{\ell+k}(\mathbf{z}_i)))$$

send individual signature $(\mathbf{z}_i, r_i)$ to all other users, otherwise send out RESTART and go to (b).

(i) After receiving RESTART from some user go to (b). Otherwise, aggregate the individual signatures $(\mathbf{z}_j, r_j)$ for $j \neq i$ as follows:

i. Compute $c_j \leftarrow \mathrm{H}_0(\mathbf{t}_j, com, \mu, L)$ for all $j \neq i$, and compute $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c_j\mathbf{t}_j$.

ii. Check $||\mathbf{z}_j||_2 \leq B$ and $\mathsf{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1$.

iii. If the check fails for some j send out ABORT.

iv. Compute $\mathbf{z} = \sum_{j \in \mathcal{G}} \mathbf{z}_j$ and $r := \sum_{j \in \mathcal{G}} r_j$

(j) If the protocol does not ABORT the multi-signature is $(com, \mathbf{z}, r)$.

3. **Signature Verification:** Given $\{(com, \mathbf{z}, r), \mu, L = \{pk_1, \cdots, pk_n\}\}$ one can verify the multi-signature as follows:

(a) Compute $c_j := \mathrm{H}_0(\mathbf{t}_j, com, \mu, L)$ for $j \in \mathcal{G}$, and reconstruct $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_{j \in \mathcal{G}} c_j\mathbf{t}_j$.

(b) Accept if $||\mathbf{z}||_2 \leq B_n$ and $\mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1$, where $B_n = \sqrt{n}B$.

Throughout the next section, in order to avoid confusion we follow the same notation of [6] and the parameters in Table 1.

3 vMS$_2$: A lattice-based ASM scheme with verifiable group setup

In the vASM scheme (see section 2.4), it is considered to sign a message with a special signing key, i.e. membership key (mk). Each user in the group $\mathcal{G}$ participate in a 1-round interactive group setup protocol in which each user shares his own secret key (sk) via a verifiable secret sharing scheme. Upon receiving the shares from other users, each user sums those shares up and computes his membership key (mk). Therefore, each membership key is composed of the secret keys of all users in the group $\mathcal{G}$. So, this construction provides accountability to any signer who signs with a membership key. Below we apply the above mentioned group setup method to the MS_2 scheme [6].

1. **Key Generation Phase:** Let $\mathcal{G}$ be a group of n potential signers and let $\mathcal{S} \subseteq \mathcal{G}$ be the subgroup of τ signers among n. Let $\mathrm{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$ be another random oracle in addition to H_0 and H_3. (Note that the length l_1 should be long enough for the random oracle to be secure [6].) Although the matrix $\mathbf{A}$ can be determined by a trusted party, here we assume that it is computed interactively. Given public parameters $par := \{R_q, k, \ell, \eta, B, s, M\}$ i-th user proceeds as follows:

(a) $\mathbf{A}_i \xleftarrow{\$} R_q^{k \times \ell}$

(b) Computes $g_i := \mathrm{H}_1(\mathbf{A}_i, i)$ and sends g_i to the j-th user.

(c) Upon receiving g_j for $j \neq i$ sends $\mathbf{A}_i$ to j-th user.

(d) Upon receiving $\mathbf{A}_j$ for $j \neq i$:

i. If $g_j \neq \mathrm{H}_1(\mathbf{A}_j, j)$ for some j then send out ABORT.

ii. Otherwise set $\mathbf{A} := \sum_{j=1}^{n} \mathbf{A}_j$ and $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (\ell+k)}$, where $\mathbf{I}$ is $k \times k$ identity matrix.

(e) Picks random secrets $(\mathbf{s}_{i_1}, \mathbf{s}_{i_2}) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$, $\mathbf{s}_i := \begin{bmatrix} s_{i_1} \\ s_{i_2} \end{bmatrix}$

(f) Compute $\mathbf{t}_i := \bar{\mathbf{A}} \cdot \mathbf{s}_i$

(g) $\mathbf{s}_i$ is the secret key and $(\bar{\mathbf{A}}, \mathbf{t}_i)$ is the public key pair of i-th user.

2. **Group Setup Phase:** Each user $i \in \mathcal{G}$ proceeds as follows:

(a) Choose a polynomial $f_i(x) = \alpha_{n-1}^{(i)} x^{n-1} + \ldots + \alpha_1^{(i)} x + \alpha_0^{(i)}$ where $\alpha_0^{(i)} = \mathbf{s}_i$ and $\alpha_1^{(i)}, \ldots, \alpha_{n-1}^{(i)}$ are all nonzero and chosen randomly from $S_\eta^{\ell+k}$.

(b) Compute the set of commitments $\mathrm{CS}_i := \{\mathbf{C}_u^{(i)} = \bar{\mathbf{A}} \cdot \alpha_u^{(i)} : u = 0, \ldots, n-1\}$.

(c) Sends $(f_i(j), \mathrm{CS}_i)$ to the j-th user in $\mathcal{G}$, for $j = 1, \ldots, n$, along with a signature ϖ_i on $\mathbf{t}_i$ using the secret key $\mathbf{s}_i$. Note that signature ϖ_i is used for proof of possession of the secret key $\mathbf{s}_i$ that is being shared in the group setup phase. (See the Remark 3.2)

(d) After receiving $(f_j(i), \mathrm{CS}_j)$ and the proof of possession signature ϖ_j:

- Compute the raw membership key $\mathbf{rmk}_i = \sum_{j \in \mathcal{G}} f_j(i)$. Notice that $||\mathbf{rmk}_i||_\infty$ would be larger than η. Therefore it has to be normalized. But before that the following consistency check is done.
- Compute $\mathrm{CS} := \{\mathbf{C}_u = \sum_{j \in \mathcal{G}} \mathbf{C}_u^{(j)} : u = 0, \ldots, n-1\}$, and check:
  - $\bar{\mathbf{A}} \cdot \mathbf{rmk}_i \stackrel{?}{=} \sum_{u=0}^{n-1} \mathbf{C}_u i^u$,
  - $\mathbf{C}_0 \stackrel{?}{=} \sum_{i \in \mathcal{G}} \mathbf{t}_i$,
  - Verify the proof of possession signature ϖ_j using $\mathbf{t}_j$.

  If one of the above checks fail, then ABORT.
- Otherwise, in order to normalize the raw membership key, i.e.

$$\mathbf{rmk}_i = \begin{pmatrix} b_{N-1}^{(1)} x^{N-1} + \ldots + b_1^{(1)} x + b_0^{(1)} \\ b_{N-1}^{(2)} x^{N-1} + \ldots + b_1^{(2)} x + b_0^{(2)} \\ \vdots \\ b_{N-1}^{(\ell+k)} x^{N-1} + \ldots + b_1^{(\ell+k)} x + b_0^{(\ell+k)} \end{pmatrix},$$

construct a vector $\mathbf{\Delta}_i \in R_q^{\ell+k}$, that is

$$\mathbf{\Delta}_i = \begin{pmatrix} \boldsymbol{\delta}^{(1)} \\ \boldsymbol{\delta}^{(2)} \\ \vdots \\ \boldsymbol{\delta}^{(\ell+k)} \end{pmatrix} = \begin{pmatrix} \delta_{N-1}^{(1)} x^{N-1} + \ldots + \delta_1^{(1)} x + \delta_0^{(1)} \\ \delta_{N-1}^{(2)} x^{N-1} + \ldots + \delta_1^{(2)} x + \delta_0^{(2)} \\ \vdots \\ \delta_{N-1}^{(\ell+k)} x^{N-1} + \ldots + \delta_1^{(\ell+k)} x + \delta_0^{(\ell+k)} \end{pmatrix},$$

where $\delta_u^{(v)}$ is sampled uniformly random from the interval $[q - b_u^{(v)} - \eta, q - b_u^{(v)} + \eta]$, for $u = 0, \ldots, N-1$ and $v = 1, \ldots, \ell + k$.

- Then compute the membership key $\mathbf{mk}_i = \mathbf{rmk}_i + \mathbf{\Delta}_i$ and make the set of membership public keys, i.e., $\mathrm{MPK} = \{\mathbf{T}_i = \bar{\mathbf{A}} \cdot \mathbf{mk}_i\}_{i \in \mathcal{G}}$, public.

Notice that if the normalization is not performed then the coefficients of the membership keys will be in the interval $[-\frac{q-1}{2}, \frac{q-1}{2}]$, but after the normalization process they will be small enough, i.e. in the interval $[-\eta, \eta]$ as desired for a signing key.

3. **Signature Generation:** Let H_0 and H_3 be the random oracles as in Section 2.6. Given a message $\mu \in \{0,1\}^*$, i-th user signs as follows:

(a) Computes the commitment key $ck \leftarrow \mathrm{H}_3(\mu, L)$, where $L = \{pk_1, \ldots, pk_n\}$.

(b) Picks a random $\mathbf{y}_i \xleftarrow{\$} D_s^{\ell+k}$ and computes $\mathbf{w}_i = \bar{\mathbf{A}}\mathbf{y}_i$

(c) Computes $com_i \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_i, r_i)$, where r_i is a discrete Gaussian vector sampled from the Gaussian distribution $D(S_\tau)$ defined in the trapdoor commitment scheme in [6].

(d) Sends com_i to the the j-th user in $\mathcal{S}$, for $j \neq i$.

(e) Upon receiving com_j from other users in $\mathcal{S}$, computes $com := \sum_{j \in \mathcal{S}} com_j$

(f) Computes $c_i \leftarrow \mathrm{H}_0(\mathbf{t}_i, com, \mu, L)$

(g) Computes his individual signature $\mathbf{z}_i = c_i \mathbf{mk}_i + \mathbf{y}_i$

(h) Runs the rejection sampling on input $(c_i\mathbf{mk}_i, \mathbf{z}_i)$, i.e. with probability

$$min(1, D_s^{\ell+k}(\mathbf{z}_i)/(M \cdot D_{c_i mk_i, s}^{\ell+k}(\mathbf{z}_i)))$$

sends out $(\mathbf{z}_i, r_i)$, otherwise sends out RESTART and goes to (b).

(i) If he receives RESTART from some user, then goes to (b). Otherwise, upon receiving $(\mathbf{z}_j, r_j)$ for $j \in \mathcal{S}$ computes the aggregated signature as follows:

i. Computes $c_j \leftarrow \mathrm{H}_0(\mathbf{t}_j, com, \mu, L)$ for all $j \in \mathcal{S}$, and computes

$$\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c_j \mathbf{T}_j$$

ii. Checks $||\mathbf{z}_j||_2 \leq B$ and $\mathsf{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1$.

iii. If the check fails for some $j \in \mathcal{S}$, he sends out ABORT.

iv. Otherwise, he computes $\mathbf{z} = \sum_{j \in \mathcal{S}} \mathbf{z}_j$ and $r := \sum_{j \in \mathcal{S}} r_j$

(j) If the protocol does not ABORT, then the accountable subgroup multi-signature is $(com, \mathbf{z}, r)$.

4. **Verification:** Given $\{(com, \mathbf{z}, r), \mu, \mathcal{S}, L = \{pk_1 \ldots, pk_n\}, \mathrm{MPK} = \{\mathbf{T}_1, \ldots, \mathbf{T}_n\}\}$ one can verify the vMS_2 signature as follows:

(a) Compute $c_j := \mathrm{H}_0(\mathbf{t}_j, com, \mu, L)$ for $j \in \mathcal{S}$, and reconstruct $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_{j \in \mathcal{S}} c_j \mathbf{T}_j$.

(b) Accept if $||\mathbf{z}||_2 \leq B_\tau$ and $\mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1$, where $B_\tau = \sqrt{\tau} \cdot B$ (see Lemma 1).

3.1 Remarks

Remark 3.1. Since the underlying multi-signature scheme [6] has interactive signing phase, in our construction we assume that the subgroup $\mathcal{S}$ is known to all signers before the protocol starts. In other words, each signer in $\mathcal{S}$ knows his co-signers.

Remark 3.2. The checks in the group setup phase is performed to verify whether the users shared their secret keys honestly. The first one checks whether the shares are consistent with the shared secrets. The second and the third ones ensures that the users shared their secret keys honestly. Notice that if we do not enforce the users to send a proof of possession signature on their public keys, a malicious user may share an arbitrary value γ_j with $||\gamma_j||_\infty > \eta$ such that $\mathbf{t}_j = \bar{\mathbf{A}} \cdot \gamma_j$, which satisfies the second check without knowing the $\mathbf{s}_j$.

Remark 3.3. Notice that each $||\alpha_i^{(j)}||_\infty \leq \eta$ for $i = 0, \ldots, n-1$ and $j = 1, \ldots, n$. Therefore neither $||f_j(i)||_\infty$ nor $||\mathbf{rmk}_i||_\infty = ||\sum_{j \in \mathcal{G}} f_j(i)||_\infty$ is under control (i.e. their infinity norms may be larger than η). In order to have a membership key with ℓ^∞ norm at most η, we normalize the $\mathbf{rmk}_i$ with a user-specific normalizer vector, i.e. Δ_i which has entries whose coefficients are chosen from a coefficient-specific interval $[q - b_u^{(v)} - \eta, q - b_u^{(v)} + \eta]$, for $u = 0, \ldots, N-1$ and $v = 1, \ldots, \ell + k$. Then the membership key is $\mathbf{mk}_i = \mathbf{rmk}_i + \Delta_i$ and $||\mathbf{mk}_i||_\infty \leq \eta$ as desired.

Remark 3.4. Note that one may publish the set of public normalizer vectors, i.e., NS = $\{\mathbf{N}_i = \bar{\mathbf{A}}\boldsymbol{\Delta}_i\}$ and the commitment set CS := $\{\mathbf{C}_u = \sum_{j\in\mathcal{G}} \mathbf{C}_u^{(j)} : u = 0, \ldots, n-1\}$ to make the membership public keys publicly verifiable. In this case, the verifier can also compute $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_{j\in\mathcal{S}} c_j . \left(\sum_{u=0}^{n-1} \mathbf{C}_u j^u + \mathbf{N}_j\right)$. However, in this case the we incur additional operations in the verification equation. Moreover this modification requires $2nk$ ring elements, i.e., for CS and NS, to be public.

Remark 3.5. Assume that B is the upper bound for the size of the individual signatures according to Lemma 2-3. From Lemma 1, the upper bound of our vMS_2 signature is $B_\tau \leq \sqrt{\tau} \cdot B$, where $\tau = |\mathcal{S}|$.

Remark 3.6. In our proposed vMS_2 scheme, we have a 2-round interaction in signing phase. In addition to that we have also a one-time 1-round interaction in the group setup phase. In the key generation phase we assume that the matrix $\mathbf{A}$ is computed interactively. Therefore we also assume 2 more rounds of interactions for once. Note that, in case of a trusted third party, our key generation phase becomes non-interactive.

Remark 3.7. Before ending technical remarks, we would like to point out that our group setup method which we use in vMS_2 could be applied to any future multi-signature scheme based on Fiat Shamir with Aborts paradigm.

3.2 Security Analysis

We simply change the signing key of MS_2 scheme [6] by adding a group setup phase which includes a secure joint verifiable secret sharing scheme. Namely a vMS_2 signature is nothing but a MS_2 signature with τ signers, which is signed by the membership keys (mk_i) instead of secret keys (s_i). Therefore security of our vMS_2 scheme simply follows from the security of the MS_2 scheme.

Lemma 4. *The group setup phase is secure.*

Proof. The security of the group setup phase follows from the security of VSS scheme whose security depends on Module LWE and Module SIS problems. □

Lemma 5. *If the MS_2 scheme is secure, then the vMS_2 scheme is secure.*

Proof. Notice that key generation phase is the same as MS_2 scheme. In the group setup phase, each user shares his secret key via a VSS protocol. The output of the group setup phase is the membership keys which are of the same size and from the same distribution as of secret keys. Signature generation, signature aggregation and verification phases are the same as in MS_2. □

Next, we modify the security theorem of the MS_2 scheme according to vMS_2 scheme, and state it below.

Theorem 3.8 (Theorem 3 [6])**.** *Suppose the trapdoor commitment scheme TCOM is secure, additively homomorphic and has uniform keys. For any probabilistic polynomial-time adversary $\mathcal{A}$ that initiates $\mathcal{Q}_s$ signature generation protocols by querying $\mathcal{O}_\tau^{\mathsf{vMS}_2}$, and makes $\mathcal{Q}_h$ queries to the random oracle H_0, H_3, the protocol vMS_2 is MS-UF-CMA secure under Module-SIS$_{q,k,\ell+1,\beta}$ and Module-LWE$_{q,k,\ell,\eta}$ assumptions, where $\beta = 2\sqrt{B_\tau^2 + \kappa}$. Concretely, using other parameters specified in Table 1, the advantage of $\mathcal{A}$ is bounded as follows.*

$$Adv^{MS\text{-}UF\text{-}CMA}_{\mathsf{vMS}_2}(\mathcal{A}) \leq e \cdot (\mathcal{Q}_h + \mathcal{Q}_s + 1) \cdot \Bigg((\mathcal{Q}_h + \mathcal{Q}_s)\epsilon_{td} + \mathcal{Q}_s \cdot \frac{2e^{-t^2/2}}{M} + Adv_{Module-LWE_{q,k,\ell,\eta}} + \frac{(\mathcal{Q}_h + \mathcal{Q}_s + 1)}{|C|} + \sqrt{(\mathcal{Q}_h + \mathcal{Q}_s + 1) \cdot \left(\epsilon_{bind} + Adv_{Module-SIS_{q,k,\ell+1,\beta}}\right)} \Bigg)$$

Parameter	Description
n	Number of users in group $\mathcal{G}$
τ	Number of signers in subgroup $\mathcal{S} \subseteq \mathcal{G}$
N	the degree of f(X) which is a power of 2
$f(X) = X^N + 1$	The $2N$-th cyclotomic polynomial
q	Prime modulus
$R = \mathbb{Z}[x]/(f(X))$	Cyclotomic ring
$R_q = \mathbb{Z}_q[x]/(f(X))$	Ring
k	The height of the public matrix A
ℓ	The width of the public matrix A
γ	Parameter defining the tail-bound of Lemma 2
$B = \gamma\sigma\sqrt{N(\ell+k)}$	The maximum ℓ^2 norm of the individual signatures $z_j \in R^{\ell+k}$ for $j = 0, \ldots, n$
$B_n = \sqrt{n}B$	The maximum ℓ^2 norm of combined signature $z \in R^{\ell+k}$
κ	The maximum ℓ^1 norm of challenge vector c
$C = \{c \in R : \|\|c\|\|_\infty = 1 \wedge \|\|c\|\|_1 = \kappa\}$	Challenge space where $\|C\| = \binom{N}{\kappa}2^\kappa$
$S_\eta = \{x \in R : \|\|x\|\|_\infty \leq \eta\}$	Set of small secrets
$T = \kappa\eta\sqrt{N(\ell+k)}$	Chosen such that Lemma 4 of [6] holds
α	Parameter defining σ and M
$\sigma = s/\sqrt{2\pi} = \alpha T$	Standard deviation of the Gaussian distribution
$t = \omega(\sqrt{\log(mN)}) \wedge t = o(\log(mN))$	Parameter defining M such that Lemma 3 holds
$M = e^{t/\alpha + 1/(2\alpha^2)}$	The expected number of restarts until a single party can proceed
$M_n = M^n$	The expected number of restarts until all n parties proceed simultaneously
l_1	Output bit length of random oracles H_1
TCOM	Additively homomorphic trapdoor commitment scheme proposed in [6].
(Commit,Open)	"Commiting" and "Opening" algorithms of TCOM

Table 1: Parameters for MS_2 [6] and vMS_2 schemes

3.3 Computational Analysis

Since our underlying hard problems are Module-SIS and Module-LWE, our computations are done in the base ring $R_q = \mathbb{Z}_q[x]/(x^N + 1)$, i.e. polynomial addition and multiplication modulo $x^N + 1$. In Table 2 we give the number of computations required in each phase of our proposed vMS_2 scheme in comparison with the MS_2. In the last column of the comparison table we state the cost of accountability under the assumption of $\tau = n$, i.e. all of the users in the group $\mathcal{G}$ participate in the vMS_2 signature. In other words, if the same number of signers participate in vMS_2 and MS_2 schemes at the same time, the vMS_2 scheme requires more operations than MS_2 as a cost of accountability.

More precisely, in comparison with MS_2 scheme,

- Matrix generation and key generation phases of our vMS_2 scheme are identical to the MS_2 scheme,
- vMS_2 scheme has an additional one time group setup phase.
- In vMS_2 scheme, the broadcasted data size also grows with extra nk ring elements.

- Since each user has an additional membership key (with same size) in vMS_2 scheme, the need of storage doubles.

Phases	MS_2 Scheme [6]	vMS_2 Scheme	# extra operations in vMS_2 (for $\tau = n$)
Matrix Generation	Uniform Sampling: $k\ell$ R_q-Elt. Computation: n Hashes (H_0) $k\ell(n-1)$ Add.	Uniform Sampling: $k\ell$ R_q-Elt. Computation: n Hashes (H_0) $k\ell(n-1)$ Add.	none
Key Generation	Uniform Sampling: $k+\ell$ R_q-Elt. Computation: $k(k+\ell)$ Mult. $k(k+\ell-1)$ Add.	Uniform Sampling: $k+\ell$ R_q-Elt. Computation: $k(k+\ell)$ Mult. $k(k+\ell-1)$ Add.	none
Group Setup	None	Multiplication: $(n+1)(k^2+k\ell)+n^2$ Addition: $(n+1)(k^2+k\ell-k)+n^2(2k+\ell)-1$ Uniform Sampling ($f(x)$): $(n-1)(k+\ell)$ R_q-Elt. Uniform Sampling (Δ_i): $N(k+\ell)$ integers	Multiplication: $(n+1)(k^2+k\ell)+n^2$ Addition: $(n+1)(k^2+k\ell-k)+n^2(2k+\ell)-1$ Uniform Sampling ($f(x)$): $(n-1)(k+\ell)$ R_q-Elt. Uniform Sampling (Δ_i): $N(k+\ell)$ integers
Signature Generation	1 Hash (H_3) 1 Commit $(n-1)$ Open n Hashes (H_0) Sampling $(k+\ell)$ R_q-Elt. Sampling $(\ell+2w)$ R_q-Elt$^{(*)}$. $n(k^2+k\ell+k)+\ell$ Mult. $n(k^2+k\ell+k+\ell)$ Add.	1 Hash (H_3) 1 Commit $(\tau-1)$ Open τ Hashes (H_0) Sampling $(k+\ell)$ R_q-Elt. Sampling $(\ell+2w)$ R_q-Elt$^{(*)}$. $\tau(k^2+k\ell+2k)+\ell$ Mult. $\tau(k^2+k\ell+k+\ell)$ Add.	none
Verification	n Hashes H_0 $k(k+\ell+n)$ Mult. $k(k+\ell+n-2)$ Add. 1 Open	τ Hashes H_0 $k(k+\ell+\tau)$ Mult. $k(k+\ell+\tau-2)$ Add. 1 Open	none
Transmission (R_q Elements)	$k\ell+k+\ell$ 1 (integer) $\ell+2w+2$ for $\mathsf{TCOM}^{(*)}$	$k(n+\ell)+2(k+\ell)$ 1 (integer) $\ell+2w+2$ for $\mathsf{TCOM}^{(*)}$	$k(n+1)+\ell$
Broadcasting (R_q Elements)	$k(\ell+k+1)$	$k(n+k+\ell+1)$	nk
Storage (R_q Elements)	$(k+\ell)$	$2(k+\ell)$	$(k+\ell)$

(*) Notice that $r \in R_q^{\ell+2w}$ and $com \in R_q^2$. (see TCOM definition in [6])
Add. - Addition
Mult. - Multiplication
Elt. - Element

Table 2: Comparison of the schemes

4 Conclusion

In this paper we propose a novel lattice-based ASM scheme, i.e., vMS_2 which is based on the two-round multi-signature scheme MS_2 proposed in [6]. By adding a group setup phase to MS_2 scheme, we ensure all the users give authorization to others in $\mathcal{G}$ to sign on behalf of the group $\mathcal{G}$. We achieve the accountability

with the cost of one-time one-round interactive group setup and increased broadcast data size. Design of a more efficient vASM scheme based on other post-quantum problems would be a good future work.

References

[1] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, page 245–254, New York, NY, USA, 2001. Association for Computing Machinery.

[2] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018*, pages 435–464, Cham, 2018. Springer International Publishing.

[3] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, September 2004.

[4] Ahmet Ramazan Ağırtaş and Oğuz Yayla. Pairing-based accountable subgroup multi-signatures with verifiable group setup. Cryptology ePrint Archive, Report 2022/018, 2022. `https://ia.cr/2022/018`.

[5] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium algorithm specifications and supporting documentation. 2017.

[6] Ivan Bjerre Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2), April 2022.

[7] Ronald A Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research.* Oliver and Boyd Ltd, London, 1943.

[8] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438, Oct 1987.

[9] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[10] Martin Tompa and Heather Woll. How to share a secret with cheaters. *Journal of Cryptology*, 1(3):133–138, Oct 1989.

[11] Ahmet Ramazan Ağırtaş and Oğuz Yayla. Compartment-based and hierarchical threshold delegated verifiable accountable subgroup multi-signatures. Cryptology ePrint Archive, Paper 2023/548, 2023. `https://eprint.iacr.org/2023/548`.

[12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 738–755, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks*, Lecture Notes in Computer Science, pages 368–385. Springer International Publishing, August 2018. 11th Conference on Security and Cryptography for Networks, SCN 2018 ; Conference date: 05-09-2018 Through 07-09-2018.