DOI: 10.2478/ias-2025-0007 • JIAS • 20(3) • 2025 • 106-124

ML-GS: Module Lattice-Based Group Signature Scheme

Deng Pan [1*], Yatao Yang [1, 2*], Weitao Sun [1], Shuaibo Wang [2] and Ke Wang[1]

[1] Department of Electronics and Communication Engineering, Beijing Electronic Science and Technology Institute, Beijing 100070, China; 3096958887@qq.com (W.S.); 1206048996@qq.com (S.W.); wangke_unique@163.com (K.W.)

[2] School of Telecommunication Engineering, Xidian University, Xi'an 710071, China

*Correspondence author: pandeng@home.hpu.edu.cn(D.P.); yy2008@163.com (Y.Y.)

Abstract: With the rapid growth of online users, protecting user privacy in access control scenarios has become a critical challenge in the field of information security. Group signatures serve as a fundamental cryptographic primitive that enables users to sign on behalf of a group, providing anonymity while supporting traceability of signers. However, traditional group signature schemes, which rely on number-theoretic assumptions vulnerable to quantum algorithms, face significant security threats in the advent of quantum computing. In this paper, we propose a module lattice-based group signature scheme (ML-GS). ML-GS leverages FIPS 204 standard, a recently standardized signature scheme by the National Institute of Standards Technology (NIST), and integrates a dual-rejection-sampling signature mechanism with the K-PKE encryption scheme from FIPS 203 standard, forming a "sign-hybrid-encrypt" hybrid structure to ensure both efficiency and traceability. In the key generation phase, we introduce a module Gaussian preimage sampling algorithm that reduces public key size and supports dynamic user enrollment. The security of ML-GS is formally proven in the random oracle model under the Module Learning with Error (MLWE) and Module Short Integer Solution (MSIS) assumptions. Experimental results demonstrate that, compared to existing scheme with equivalent security level, the ML-GS scheme achieves significant improvements in both time and storage overhead.

Keywords: group signature, lattice, post-quantum cryptography, module learning with error, module short integer solution

1 Introduction

1.1 Background

Group signatures (GS) are a special class of digital signatures designed to provide privacy protection for groups. As a prominent research topic in public-key cryptography, GS has attracted increasing attention in recent years and has been applied in various multi-user scenarios. Any group member can generate a signature on behalf of the entire group. The validity of the signature can be publicly verified, but the identity of the actual signer remains hidden (anonymity). A group manager, equipped with a tracing key, can reveal the identity of the signer when necessary (traceability). Due to these two core properties, group signatures are well-suited for a range of network applications, such as electronic voting, vehicular networks, and anonymous cryptocurrencies [1–3]. Figure 1 illustrates the application of GS in electronic contract signing.

Most existing GS schemes are constructed based on traditional number-theoretic hardness assumptions, such as integer factorization and discrete logarithms. These problems are believed to be intractable for classical computers in polynomial time. However, the introduction of Shor algorithm [4] demonstrated that quantum computers can efficiently solve these problems, posing a significant threat to the cryptographic foundations of the current Internet infrastructure in the post-quantum era. Although the timeline for practical quantum computers remains uncertain, ongoing advances in quantum algorithms [5] suggest that their realization may not be far off. Considering this, the development of quantum-resistant cryptographic schemes has become an urgent and critical area of research.

In response to the disruptive threat that quantum computing poses to traditional cryptographic schemes, the National Institute of Standards and Technology (NIST) has launched a standardization process for post-quantum cryptography. Among the various candidates, lattice-based cryptography has emerged as one of the most promising approaches due to its strong security guarantees and practical efficiency. Since Ajtai's pioneering work in 1996 [6], which first introduced lattice problems into cryptography, subsequent research has shown that current quantum algorithms cannot efficiently solve the underlying hard problems on lattices [7]. In addition to quantum resistance, lattice-based schemes offer several desirable features, including worst-case to average-case reductions, parallelizability, and high computational efficiency. These advantages have made lattice-based group signature schemes a highly active area of research. Although existing lattice-based GS schemes have been proven secure under well-established lattice hardness assumptions, they still suffer from substantial storage overhead and relatively low signing efficiency, which remain key challenges for practical deployment.

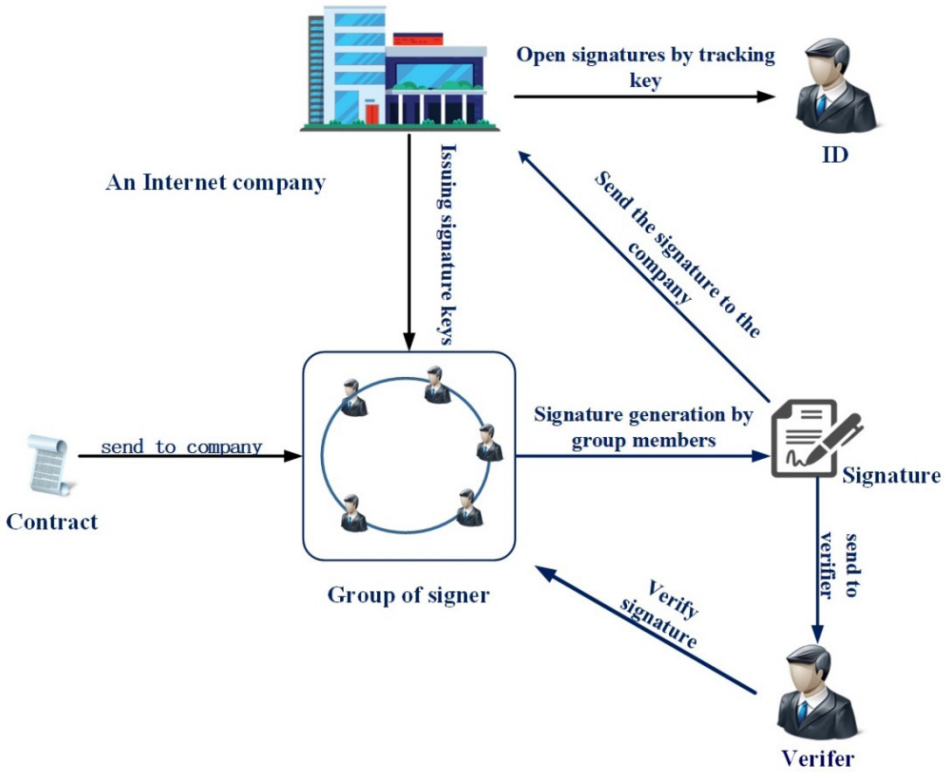

Figure 1. Group Signature in Electronic Contract Signing

In August 2024, NIST officially released the first set of post-quantum cryptographic standards, FIPS 203-205 [8–10], marking a critical milestone in the transition from classical to quantum-resistant cryptographic systems. Among them, FIPS 204 standardizes a lattice-based digital signature scheme known as Dilithium [11]. Dilithium is designed within the "Fiat-Shamir with Aborts" framework, which avoids the complexities of Gaussian sampling and enables a simple, efficient signing process. This construction has provided valuable insights for the design of GS scheme. In particular, by adopting Dilithium's signature structure for the group member signing phase, it is possible to significantly improve signing speed and reduce signature size.

Based on the above, the motivation for proposing our new group signature scheme is as follows:

1. **To address the urgent need for quantum-resistant group signatures:** The advancement of quantum computing threatens the security of traditional group signature schemes, making the development of post-quantum alternatives a critical research imperative.
2. **To overcome practical limitations in existing lattice-based group signatures:** While lattice-based cryptography offers a promising path to quantum resistance, many current lattice-based group signature schemes are hindered by significant storage overhead and suboptimal signing efficiency, limiting their practical applicability.
3. **To leverage recent advancements in post-quantum cryptography standards:** The Dilithium signature scheme, standardized by NIST in FIPS 204, provides robust and efficient building blocks. We aim to adapt its core principles to the group signature setting to enhance performance and security.

In this paper, we propose a module lattice-based group signature scheme (ML-GS). Our specific contributions are as follows:

To ensure the dynamic joining of group members, we adopt a Gaussian preimage sampling technique over module lattices. The module nature of our construction enables flexible parameter selection to accommodate varying security levels. We also utilize a public matrix to generate identity vectors for each group member. This approach reduces the number of public matrices required for identity verification to just two, thereby significantly minimizing both public and private key sizes. Furthermore, the ML-GS scheme follows a sign-hybrid-encrypt paradigm, with its design centered around the FIPS 204 signature scheme, Dilithium [9]. We employ a dual-Dilithium signature mechanism to enhance efficiency and reduce the overall signature size. This construction guarantees that, when a valid signature is opened, the algorithm cannot reveal the identity of any signer other than the actual one. To ensure correct identity opening, the identity vectors of group members are encrypted using the K-PKE encryption scheme from FIPS 203 [8]. The resulting ciphertext is partially reused to derive randomness in the local signature generation process, while certain components of the local signature are embedded as plaintext in the sign -hybrid-encrypt mechanism.

Additionally, the ML-GS scheme achieves both anonymity and traceability in the Random Oracle Model (ROM), with its security grounded on the Module Learning With Errors (MLWE) and Module Short Integer Solution (MSIS) assumptions. Finally, we compare ML-GS scheme with a GS scheme [12] proposed at Eurocrypt 2022. The results show that our scheme achieves a 6% reduction in public key size and an 80% reduction in signature size, at the cost of a 16% increase in each member's signing key size.

1.2 Related Work

In 2010, Gordon et al. [13] were the first to introduce a lattice-based GS scheme, constructed using non-interactive zero-knowledge (NIZK) proofs. While theoretically significant, their scheme suffered from impractically large signature sizes. In 2013, Laguillaumie et al. [14] proposed an improved scheme that reduced the signature size to a logarithmic scale relative to the number of group members. In 2015, Ling et al. [15] presented a GS scheme based on ideal lattices, which significantly reduced key and signature sizes by transitioning to a ring-based setting. In 2018, Ling et al. [16] introduced a constant-size GS scheme using the "constrained guessing" technique [17] to address the issue of linearly growing signature sizes. However, this scheme required overly large parameters and suffered from completeness issues in its NIZK component. NIZK remains a foundational technique in many lattice-based GS schemes [18–21]. In 2019, Katsumata et al. [22] proposed a GS scheme in the standard model, eliminating the need for NIZK proofs of group membership during signature verification. In 2024, Tang et al. [23] proposed an event-oriented linkable GS scheme that uses a linking algorithm to detect multiple signatures from the same signer, thereby preventing malicious abuse. In 2025, Zhang et al. [24] creatively combined the Chinese Remainder Theorem and the trapdoor sampling inverse transformation algorithm based on the GKV scheme [13] to design a new lattice-based group signature scheme, which ensures privacy protection for medical data in the medical field. In recent years, a growing number of GS schemes have been proposed and applied across a wide range of scenarios [25–29].

From the development trajectory of lattice-based group signatures outlined above, we can summarize that group signature schemes are typically constructed from multiple cryptographic components: a signature scheme is used to issue private keys to group members, zero-knowledge proofs ensure the anonymity of identities, and a lattice-based encryption scheme guarantees the traceability of signatures. In recent years, the continuous proposal of lattice-based signature schemes [30, 31] has also led to the ongoing optimization of key issuance in group signatures. In ML-GS, we use the [32] scheme to issue private keys to users. Instead of using zero-knowledge proofs to ensure member anonymity, we employ a dual Dilithium signature architecture. Simultaneously, the continuous advancements in lattice-based encryption schemes [33] also ensure the traceability of the ML-GS scheme.

The remainder of this paper is organized as follows: In Section 2, we present the basic notation and relevant algorithms. In Section 3, we introduce the definition and security model of group signatures. In Section 4, we describe the construction of the ML-GS scheme. In Section 5, we provide a security analysis of the ML-GS scheme. In Section 6, we analyze the efficiency of ML-GS and present comparative experiments. Finally, Section 7 concludes the paper.

2 Preliminaries

2.1 Definition of Symbols

The notations used in this paper are described in Table 1.

Table 1. Notation Definition

Notations	Description
Z_q^n	n-dimensional vector over modulo q
$Z_q^{n\times m}$	$n\times m$ matrix over modulo q
R	polynomial $Z[[x]/(x^d+1)$
R_q	quotient ring $Z[_q[x]/(x^d+1)$
$x \leftarrow D$	x is sampled from the distribution D
$\|\|a\|\|$	$\sqrt{\sum a_i^2}$
$\|\| a \|\|_\infty$	$\max_j \| a_j \|$
$\|\| \boldsymbol{w} \|\|$	$\sqrt{\sum \|\| w_i \|\|^2}$, where $\boldsymbol{w}=(w_0,....,w_k)\in R^k$
$\|\| \boldsymbol{w} \|\|_\infty$	$\max_j \|\| w_j \|\|_\infty$, where $\boldsymbol{w}=(w_0,....,w_k)\in R^k$
$\perp$	the algorithm outputs fail
$negl(\lambda)$	a non-negligible function about λ
B	the set of integers representable by a single byte, i.e., {0,1,…,255}

2.2 Hardness Assumption

Definition 1 ($\mathrm{MLWE}_{m,n,x}$ problem[34]). Set distribution , $\chi=\{a\in R,\| a \|_\infty \le 1\}$, select a matrix $\boldsymbol{A}\in R_q^{m\times n}$ and a distributions $(\boldsymbol{s},\boldsymbol{e})\leftarrow \chi^n\times\chi^m$ chosen from χ , the $\mathrm{MLWE}_{q,m,n}$ problem is to distinguish between *m* samples drawn from $(\boldsymbol{A},\boldsymbol{As}+\boldsymbol{e})$ and $(\boldsymbol{A},\boldsymbol{b})\leftarrow R_q^{m\times n}\times R_q^m$.

Definition 2 ($\mathrm{MSIS}_{n,m,\beta}$ problem[34]). Select a matrix $\boldsymbol{A}\in R_q^{n\times m}$, find a vector $\boldsymbol{z}\in R^m$ such that $\boldsymbol{Az}=\boldsymbol{0}$ and $0\le\| z \|\le\beta$.

Lemma 1 [35]**.** For any standard deviation $\sigma>0$ and positive integer k , the following formula holds:

1) $\Pr[x\leftarrow D_\sigma :| x |> k\sigma]\le 2e^{-k^2/2}$.

2) $\Pr[\boldsymbol{x}\leftarrow D_\sigma^n :\| \boldsymbol{x} \|> \sqrt{2n}\cdot\sigma]<2^{-n/4}$.

The rejection sampling algorithm is illustrated in Figure 2. Next, we present a crucial lemma regarding rejection sampling, which ensures that the response values in our zero-knowledge protocol do not reveal any sensitive information.

Lemma 2 [36]. Given a probability distribution $h: V \to [0,1]$, where $V = \{\vec{\boldsymbol{v}} \in \mathrm{R}^n, \| \vec{\boldsymbol{v}} \| < T\}$. Let $\boldsymbol{v} \leftarrow h$, $\boldsymbol{y} \leftarrow D_\varsigma^n$ and $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{v}$. Then the probability that $\mathrm{Rej}(\boldsymbol{z}, \boldsymbol{v}, \xi)$ outputs $b = 1$ is within $1/3 + 2^{-100}$, and when $b = 1$, the statistical distance between the distribution of $\boldsymbol{z}$ and D_ς^n is within 2^{-100}.

```
Rej(z, v, ξ)
u ← [0,1)
If u > 1/M · exp(−2 < z, v > + || v ||² /2ς²)
    return 1
Else
    return 0
```

Figure 2. Algorithm for Rejection Sampling

2.3 Trapdoor Sampling Technique on Module Lattices

To issue signing keys to group members, a trapdoor sampling algorithm is used to sample short vectors. In 2021, Bert et al. [32] proposed an efficient preimage sampling technique on module lattices, which exhibits high modularity. Depending on the modularity and specific requirements of different schemes, the operations on R_q can be easily adjusted. Based on this, we introduce two algorithms.

Given a tool vector $\boldsymbol{g}^T = [1 \quad b \quad b^2 \quad \ldots \quad b^{k-1}]$, where $k = \log_b q$. In the modular setting, we can construct a matrix

$$\boldsymbol{G} = \boldsymbol{I}_d \otimes \boldsymbol{g}^T = \begin{bmatrix} \boldsymbol{g}^T & & & \\ & \boldsymbol{g}^T & & \\ & & \ddots & \\ & & & \boldsymbol{g}^T \end{bmatrix} \in \mathrm{R}^{d \times dk}$$

and G-lattice $\Lambda_q^{\perp}(\boldsymbol{g}^T) \in \mathrm{R}_q^{dk}$. After introducing the G-lattice, we have the following lemma.

Lemma 3. Let q, d be positive integers, $\boldsymbol{H} \in \mathrm{R}_q^{d \times d}$, and σ be a Gaussian parameter. There exists a polynomial - time algorithm (PPT) $\mathrm{TrapGen}(H, \eta)$ that outputs a random matrix $\boldsymbol{A} \in R^{d \times m}$ and a matrix $\boldsymbol{T} \in \mathrm{R}^{m \times m}$, where $\boldsymbol{T}$ is a trapdoor with label $\boldsymbol{H}$.

Lemma 4. Let $\boldsymbol{T} \in \mathrm{R}^{2d \times dk}$ be the trapdoor of matrix $\boldsymbol{A} \in R^{d \times m}$, where $\boldsymbol{H} \in \mathrm{R}_q^{d \times d}$ is the label of $\boldsymbol{T}$. Given a vector $\boldsymbol{u} \in R_q^d$ and a Gaussian parameter α. Then there exists a PPT algorithm $\mathrm{SamplePre}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{H}, \boldsymbol{u}, \eta)$ that outputs a vector $\boldsymbol{x}$, whose distribution is statistically close to $D_{\Lambda_q^{\boldsymbol{u}}(A), \eta}$.

Lemma 5. Let n, q, k, m, η are the parameters set in the scheme, and $m = d(k+2)$, then $\mathrm{SamplePre}(\boldsymbol{A}, \boldsymbol{T}_{\boldsymbol{A}}, \boldsymbol{u}, \eta)$ algorithm output $\boldsymbol{s}$ satisfies $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{u}$. Under the $\mathrm{MSIS}_{n,k,m,q,\beta}$ assumption (where $\beta = 2\eta\sqrt{mn}$), it holds that $\| \boldsymbol{s} \| \leq 2\eta\sqrt{mn}$.

2.4 K-PKE Encryption Scheme

The FIPS 203 standard's key encapsulation protocol [8] is centered around an encryption scheme based on the MLWE problem. This standard provides a public-key encryption scheme, K-PKE, that is secure under CPA (Chosen Plaintext Attack). In our design of ML-GS, we will use K-PKE to encrypt group members' identity information, and the encrypted result will serve as a commitment value as part of the signature generation process.

K-PKE consists of three components: K-PKE.KeyGen, K-PKE.Encrypt and K-PKE.Decrypt. Below, we briefly describe these three algorithms according to the requirements of our scheme.

$\cdot \mathrm{K-PKE.KeyGen}()$: The algorithm samples two vectors $\boldsymbol{s}$ and $\boldsymbol{e}$ from a binomial distribution and generates a matrix $\boldsymbol{a}$ from a uniform distribution. It computes the public key $\mathrm{key} ek_{PKE} = (\mathbf{a},\mathbf{b})$, where $\boldsymbol{b} = \mathrm{NTT}^{-1}(\boldsymbol{a} \circ \hat{\boldsymbol{s}}) + \boldsymbol{e}$, the private key $dk_{PKE} = \mathbf{s}$, and outputs (ek_{PKE}, dk_{PKE}).

$\mathrm{K-PKE.Encrypt}(ek_{PKE}, m, r)$: The algorithm takes a public key ek_{PKE}, a message m, and a random bit seed r, and randomly selects a vector $\boldsymbol{r}, \boldsymbol{e}_1, \boldsymbol{e}_2$ from a binomial distribution. It then computes , $u = \mathrm{NTT}^{-1}(\boldsymbol{a} \circ \hat{\boldsymbol{r}}) + \boldsymbol{e}_1$ $v = \mathrm{NTT}^{-1}(\hat{\boldsymbol{b}} \circ \hat{\boldsymbol{r}}) + \boldsymbol{e}_2 + m$, and outputs the ciphertext $ct = (u, v)$.

$\mathrm{K-PKE.Decrypt}(dk_{PKE}, ct)$: The algorithm takes a private key dk_{PKE}, a ciphertext ct, computes $m = (v - \mathrm{NTT}^{-1}(\hat{s} \circ \hat{u}))$ and outputs the plaintext m.

3 Definition of Group Signature and Security Model

3.1 Definition of Group Signature

In this section, the proposed ML-GS scheme follows the fundamental definition of group signatures introduced by Bellare et al. [37].

Definition 3 (ML-GS). An ML-GS scheme consists of the following five PPT algorithms.

1. $\mathrm{ML\text{-}GS.Setup}(1^{\lambda})$: Given a security parameter λ, the algorithm outputs a group public key gpk, a group trapdoor key gmk, a tracing key gtk for the group manager, and an empty registration list reg.
2. $\mathrm{ML-GS.KeyGen}(gpk, gmk, i, reg)$: Given the gpk, gmk, a member's identity i, and the registration list reg, the algorithm outputs the member's signing key sk_i.
3. $\mathrm{ML-GS.Sign}(gpk, sk_i, M)$: Given the gpk, sk_i, and a message M, the algorithm outputs a signature σ.
4. $\mathrm{ML-GS.Verify}(gpk, M, \sigma)$: Given the gpk, M and a signature σ, the algorithm outputs **"Valid"** if the signature is valid; otherwise, it outputs **"Invalid"**.
5. $\mathrm{ML-GS.Open}(gpk, gtk, M, \sigma, reg)$: Given the gpk, gtk, M, σ and the registration list reg, the algorithm outputs a member index $i \in [N]$ or $\perp$.

3.2 Correctness

Verification correctness:

$$\Pr\left[0 \leftarrow \mathrm{ML\text{-}GS.Verify}(gpk, M, \sigma) \middle| \begin{array}{l} gpk, gmk, gtk \leftarrow \mathrm{ML\text{-}GS.Setup}(1^{\lambda}) \\ sk_i \leftarrow \mathrm{ML\text{-}GS.KeyGen}(gpk, gmk, i) \\ \sigma \leftarrow \mathrm{ML\text{-}GS.Sign}(gpk, sk_i, M) \end{array} \right] \leq negl(n).$$

Opening correctness:

$$\Pr\left[\perp \leftarrow \mathrm{ML\text{-}GS.Open}(gpk, gtk, M, \sigma, reg) \middle| \begin{array}{l} gpk, gmk, gtk, reg \leftarrow \mathrm{ML\text{-}GS.Setup}(1^{\lambda}) \\ sk_i \leftarrow \mathrm{ML\text{-}GS.KeyGen}(gpk, gmk, i, reg) \\ \sigma \leftarrow \mathrm{ML\text{-}GS.Sign}(gpk, sk_i, M) \\ 1 \leftarrow \mathrm{ML\text{-}GS.Verify}(gpk, M, \sigma) \end{array} \right] \leq negl(n).$$

The group signature scheme should guarantee the message reliability. The ML-GS scheme satisfies the message authentication requirements by ensuring that only legitimate group members

can generate valid signatures through the modular lattice assumptions and the FIPS 204 standard signature structure, and that any attempts to forge signatures are equivalent to solving the MLWE/ MSIS problem. The validity of a signature can be publicly verified, but only the signer can generate a legitimate signature for the corresponding message (integrity and authenticity are guaranteed).

3.3 Security Model

The security requirements of GS scheme vary depending on different application scenarios. According to the summary by Bellare et al. [37], the two most fundamental security properties of group signatures are **anonymity** and **traceability**. Anonymity ensures that the identity of the actual signer is indistinguishable to external observers. ML-GS guarantees that the verification of a valid signature does not reveal the signer's identity. In our scheme, the identity vector of the group member is concealed during the signing process using K-PKE encryption standardized in FIPS 203. Combined with a dual-Dilithium mechanism and random masking, we construct a "sign–hybrid–encrypt" structure. Traceability allows the group manager to reveal the identity of the signer when necessary, ensuring that the signer cannot deny their signing behavior. Our design guarantees traceability: any attack aiming to break traceability is shown to be as hard as solving the underlying MSIS problem. In our security proof, we demonstrate that, assuming the hardness of MSIS, no adversary can forge a valid signature and falsely attribute it to another group member.

We first introduce three random oracles.

1. $\mathrm{O}_{corrupt}(i)$ Given a member's identity i, returns a signing key sk_i.
2. $\mathrm{O}_{sign}(i, M)$: Given a member's identity i and a message M, returns a signature $\sigma \leftarrow \mathrm{ML-GS}.Sign(gpk, sk_i, M)$.
3. $\mathrm{O}P(M, \sigma)$: Given a message M and a signature σ, if the signature is the correct signature of the member for the message M, returns the identity i; otherwise, it returns $\perp$.

$Game^{anony}_{ML-GS,\mathcal{A}}(\lambda)$(Anonymity Game):

- Setup. Given λ, the challenger $\mathcal{B}$ runs the ML-GS.Setup(1^λ) and returns gpk to $\mathcal{A}$.
- Query. $\mathcal{A}$ adaptively conducts all oracle.
- Challenge. $\mathcal{A}$ submits two identities $i_0, i_1 \in [N]$ and a message M. $\mathcal{B}$ selects random $b \leftarrow \{0,1\}$ and returns ML-GS.Sign$(gpk, sk_{i_b}, M) \rightarrow \sigma$ *to* $\mathcal{A}$.
- Guess. $\mathcal{A}$ give a guess $b' \leftarrow \{0,1\}$. $\mathcal{A}$ wins if $b' = b$ and never conduct open oracle $\mathcal{O}_{open}(M, \sigma)$.

Figure 3. The Definitions for Anonymity Game

Definition 4 (Anonymity). The ML-GS scheme achieves anonymity if no adversary can identify the actual signer of a valid group signature. Specifically, the scheme is said to satisfy anonymity if every adversary's advantage in the corresponding security game (see Figure 3) remains negligible.

Definition 5 (Traceability). Traceability guarantees that, even in the presence of collusion among group members and with access to the group manager's tracing key, the adversary cannot forge a signature that resists correct attribution. The ML-GS scheme satisfies this property if the adversary's advantage in the traceability game (see Figure 4) is negligible.

$Game^{trace}_{ML-GS,\mathcal{A}}(\lambda)$(Traceability Game):

- Setup. Given λ, the challenger $\mathcal{B}$ runs the ML-GS.Setup(1^λ), return gpk and gtk to $\mathcal{A}$.
- Query. $\mathcal{A}$ adaptively conducts $\mathcal{O}_{corrupt}(\cdot)$ and $\mathcal{O}_{sign}(\cdot)$.
- Forge. $\mathcal{A}$ outputs (M^*, σ^*). It wins if:
  1) ML-GS.Verify$(gpk, event^*, M^*, \sigma^*) = 1$.
  2) One of following conditions is satisfied:
     - ML-GS.Open$(gpk, gtk, M^*, \sigma^*) = \perp$
     - ML-GS.Open $= i^*$, i^* not queried by $\mathcal{A}$

Figure 4. The Definitions for Traceability Game

4 ML-GS Scheme

4.1 ML-GS Setup

The setup algorithm $\mathrm{ML-GS.Setup}$ takes the security parameter λ as input. First, it invokes the trapdoor generation algorithm TrapGen to generate a matrix $\boldsymbol{A}$ and its corresponding trapdoor $\boldsymbol{T}_A$, preparing for the issuance of signing keys to the group members.

The algorithm $\mathrm{K-PKE.KeyGen}$ generate an encrypted public-private key pair (ek_{PKE}, dk_{PKE}).. This step ensures that during the signature generation phase, the signer's identity is correctly encrypted, and during the opening phase, the signer's identity is correctly decrypted using the private key.

A random seed ζ is selected and expanded using an XOF function (such as SHAKE-256, denoted as H in this paper) to generate additional random values. Specifically:

1) A 64-byte public key random seed ρ. It can be used to randomly sample a polynomial vector $\boldsymbol{u}$ and a polynomial matrix $\boldsymbol{B}$ from ExpandPK algorithm.

2) A 32-byte private key random seed ρ'.. It can be used to sample and generate the signing key for the member during the key generation phase.

Finally, the registration list reg is initialized as empty. The setup algorithm is shown in Algorithm 1.

Algorithm 1 $\mathrm{ML-GS.Setup}(1^{\lambda})$

1. $(\boldsymbol{A}, \boldsymbol{T}_A) \in \mathrm{R}_q^{k\times l} \times \mathrm{R}_q^{l\times l} \leftarrow \mathrm{TrapGen}(k, l, q)$
2. $(ek_{PKE} = (\boldsymbol{a}, \boldsymbol{b}), dk_{PKE} = \boldsymbol{s}) \leftarrow \mathrm{K\text{-}PKE.KeyGen}()$
3. $\zeta \leftarrow \mathrm{B}^{32}$
4. $\rho, \rho' \in \mathbf{B}^{64} \times \mathbf{B}^{32} \leftarrow H(\zeta)$
5. $(\boldsymbol{u}, \boldsymbol{B}) \in \mathrm{R}_q^{k} \times \mathrm{R}_q^{k\times k} \leftarrow \mathrm{ExpandPK}(\rho)$
6. $reg = \{\}$
7. **return** $gpk = (\rho, \boldsymbol{A}, \boldsymbol{a}, \boldsymbol{b}), gmk = (\rho', \boldsymbol{T}_A), gtk = \boldsymbol{s}, reg = \{\}$

4.2 ML-GS Key Generation

The key generation algorithm ML-GS.KeyGen takes *gpk*, *gmk*, the identity *i* of the member and the registration list *reg* as inputs. First, the private key random seed ρ'. is expanded into a polynomial vector $x_i \in S_\eta^k$ through the ExpandS algorithm, which is taken as part of the member's signing key. Then, calculate the public identity vector $\boldsymbol{g}_i = \boldsymbol{B}\boldsymbol{x}_i$, and another part of the signing key $\boldsymbol{s}_i$ is output through the trapdoor sampling algorithm $Sample(\boldsymbol{A}, \boldsymbol{T}_A, \boldsymbol{u} - \boldsymbol{g}_i, \eta)$. Finally, the identity vector $\boldsymbol{g}_i$ is added to the registration list and output member *i*'s signing key $gsk_i = (\boldsymbol{x}_i, \boldsymbol{s}_i)$ The key generation algorithm is shown in Algorithm 2.

Algorithm 2 ML-GS.KeyGen(*gpk*,*gmk*,*i*,*reg*)

1. $\boldsymbol{x}_i \in S_\eta^k \leftarrow \mathrm{ExpandS}(\rho')$
2. $(\boldsymbol{u}, \boldsymbol{B}) \in \mathrm{R}_q^{k} \times \mathrm{R}_q^{k\times k} \leftarrow \mathrm{ExpandPK}(\rho)$
3. $\boldsymbol{g}_i = \boldsymbol{B}\boldsymbol{x}_i$
4. $\boldsymbol{s}_i \in \mathrm{R}_q^{l\times 1} \leftarrow \mathrm{SamplePre}(\boldsymbol{A}, \boldsymbol{T}_A, \boldsymbol{u} - \boldsymbol{g}_i, \eta)$
5. $reg \leftarrow reg \cup \{g_i\}$
6. **return** $gsk_i = (\boldsymbol{x}_i, \boldsymbol{s}_i)$

4.3 ML-GS Signing

The signing algorithm ML-GS.Sign takes *gpk*, sk_i and *M* as input. First, the public key random seed ρ is used to expand the public parameters $\boldsymbol{u}, \boldsymbol{B}$ through the ExpandPK algorithm, and then the identity vector $\boldsymbol{g}_i$ is computed. Before signing the message, it is concatenated with seed ρ to produce a 64-byte message representation μ.

Signing algorithm consists of rejection sampling loop and an encryption. The specific steps are as follows:

To obtain the signature, the signing algorithm first encrypts the member's identity vector $\boldsymbol{g}_i$ using the encryption algorithm of K-PKE. It then selects two vectors $\boldsymbol{y}_1, \boldsymbol{y}_2$ from a Gaussian distribution $D_{\gamma_1}^k$ computes the commitment $(\boldsymbol{w}_1, \boldsymbol{w}_2)$, and uses the message representation μ and the ciphertext of $\boldsymbol{g}_i$ to generate challenge values $\tilde{c}_1, \tilde{c}_2$. Finally, the signature algorithm generates the final challenge value $\tilde{c}$ by combining the ciphertext value of $\tilde{c}_1$, and $\tilde{c}_2$ using the hash function H, and produces the final signature value through the rejection sampling algorithm. The final signature σ contains the challenge value $\tilde{c}_2$, all ciphertext information, and the response value $\boldsymbol{z}_1, \boldsymbol{z}_2$. The signing algorithm is shown in Algorithm 3.

Algorithm 3 ML-GS.Sign(gpk,sk_i,M)

1. $(\boldsymbol{u}, \boldsymbol{B}) \in \mathrm{R}_q^k \times \mathrm{R}_q^{k \times k} \leftarrow \mathrm{ExpandPK}(\rho)$
2. $\boldsymbol{g}_i = \boldsymbol{B}\boldsymbol{x}_i$
3. $\mu \leftarrow \mathrm{H}(\rho \,\|\, M)$
4. $(\boldsymbol{z}_1, \boldsymbol{z}_2) = \perp$
5. **while** $(\boldsymbol{z}_1, \boldsymbol{z}_2) = \perp$ **do**
6. $r, r' \leftarrow \mathrm{B}^{64}$
7. $ct_1 \leftarrow \text{K-PKE.Encrypt}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}_i, r)$
8. $\boldsymbol{y}_1, \boldsymbol{y}_2 \leftarrow D_{\gamma_1}^k \times D_{\gamma_1}^l$
9. $(\boldsymbol{w}_1, \boldsymbol{w}_2) = (\boldsymbol{B}\boldsymbol{y}_1, \boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2)$
10. $\tilde{c}_1 \leftarrow H(\mu \,\|\, \boldsymbol{w}_1 \,\|\, ct_1)$
11. $\tilde{c}_2 \leftarrow H(\mu \,\|\, \boldsymbol{w}_2 \,\|\, ct_1)$
12. $ct_2 \leftarrow \text{K-PKE.Encrypt}(\boldsymbol{a}, \boldsymbol{b}, \tilde{c}_1, r')$
13. $\tilde{c} \leftarrow H(\tilde{c}_2 \,\|\, ct_2)$
14. $c \leftarrow \mathrm{SampleInBall}(\tilde{c})$
15. $(\boldsymbol{z}_1, \boldsymbol{z}_2) = (\boldsymbol{y}_1 + c\boldsymbol{x}_i, \boldsymbol{y}_2 + c\boldsymbol{s}$
16. **if** $\mathrm{Rej}(\boldsymbol{z}_1, c\boldsymbol{x}_i, \gamma_1) = 0$ or $\mathrm{Rej}(\boldsymbol{z}_2, c\boldsymbol{s}_i, \gamma_1) = 0$, then $(\boldsymbol{z}_1, \boldsymbol{z}_2) = \perp$
17. **return** $\sigma = (\tilde{c}_2, \boldsymbol{z}_1, \boldsymbol{z}_2, ct_1, ct_2)$

4.4 ML-GS Verifying

The verification algorithm ML-GS.Verify takes the group public key *gpk,* a message *M*, and a signature σ as input. First, the public key random seed ρ is used to expand the public parameters $\boldsymbol{u}, \boldsymbol{B}$ through the

ExpandPK algorithm, and then the hash ρ and message M into a 64-byte message representation μ. Next, the challenge value $\tilde{c}_2$ from the signature part and the ciphertext ct_2 are used to obtain the challenge value $\tilde{c}_2$. Finally, the verification algorithm checks the validity of $\boldsymbol{z}_1, \boldsymbol{z}_2$ and verifies whether the reconstructed value $\boldsymbol{w}_2'$ matches the signer's commitment hash $\tilde{c}_2$. The verification algorithm is shown in Algorithm 4.

Algorithm 4 ML-GS.Verify(gpk, M, σ)

1. $(\boldsymbol{u}, \boldsymbol{B}) \in \mathrm{R}_q^k \times \mathrm{R}_q^{k \times k} \leftarrow \mathrm{ExpandPK}(\rho)$
2. $\mu \leftarrow \mathrm{H}(\rho \,||\, M)$
3. $\tilde{c} = \mathrm{H}(\tilde{c}_2 \,||\, ct_2)$
4. $c = \mathrm{SampleInBall}(\tilde{c})$
5. $\boldsymbol{w}_2' = \boldsymbol{B}\boldsymbol{z}_1 + \boldsymbol{A}\boldsymbol{z}_2 - \boldsymbol{u}c$
6. **if** $\| \boldsymbol{z}_1 \|_\infty \le B$ and $\| \boldsymbol{z}_2 \|_\infty \le B$ and $\tilde{c}_2 = H(\mu \,||\, \boldsymbol{w}_2' \,||\, ct_1)$ **then**
7. **return** "Valid
8. **else**
9. **return** "Invalid'

4.5 ML-GS Opening

The opening algorithm ML-GS.Open takes *gpk*, the tracing key *gtk*, a message *M*, a signature σ, and the registration list *reg* as input. First, as in the verification algorithm, it reconstructs $\boldsymbol{u}, \boldsymbol{B}, \mu, \tilde{c}$. Then, using the decryption algorithm of the K-PKE scheme and the tracing key $\boldsymbol{s}$, it recovers the identity vector $\boldsymbol{g}_i$ and $\tilde{c}_1$. If $\boldsymbol{g}_i \in reg$ and $\tilde{c}_1 = H(\mu \,||\, \boldsymbol{B}\boldsymbol{z}_1 - \boldsymbol{g}_i c \,||\, ct_1)$, the algorithm returns the identity *i*; otherwise, it returns $\perp$. The open algorithm is shown in Algorithm 5.

Algorithm 5 ML-GS.Open(gpk, gtk, M, σ, reg)

1. $(\boldsymbol{u}, \boldsymbol{B}) \in \mathrm{R}_q^k \times \mathrm{R}_q^{k \times k} \leftarrow \mathrm{ExpandPK}(\rho)$
2. $\mu \leftarrow \mathrm{H}(\rho \,||\, M)$
3. $\tilde{c} \leftarrow H(\tilde{c}_2 || \tilde{c}\mathrm{t}_2)$
4. c=SampleInBall($\tilde{\mathrm{c}}$)
5. $\boldsymbol{g}_i = \mathrm{K\text{-}PKE.Decrypt}(\boldsymbol{s}, ct_1)$
6. $\tilde{c}_1 = \mathrm{K\text{-}PKE.Decrypt}(\boldsymbol{s}, ct_2)$
7. **if** $\tilde{c}_1 = H(\mu \,||\, \boldsymbol{B}\boldsymbol{z}_1 - \boldsymbol{g}_i c \,||\, ct_1)$ and $\boldsymbol{g}_i \in reg$ **then**
8. **return** *i*
9. **else**
10. **return** $\perp$

4.6 Correctness Analysis

In this section, we prove the correctness of the ML - GS scheme, mainly including the verification correctness and the opening correctness. Suppose the signature $\sigma = (\tilde{c}_2, \boldsymbol{z}_1, \boldsymbol{z}_2, ct_1, ct_2)$ is a valid signature generated by a group member for the message *M* under the group public key $gpk = (\rho, \boldsymbol{A}, \boldsymbol{a}, \boldsymbol{b})$.

1)Verification correctness

The verifier first recovers the public key $(\boldsymbol{u}, \boldsymbol{B})$, the hash value μ of the message M, and the challenge $\tilde{c}$, and calculates the commitment $\boldsymbol{w}_2'$. If all coefficients of $\backslash \boldsymbol{z}_1, \boldsymbol{z}_2$ are less than $\gamma_1 - \beta$, and the commitment $\boldsymbol{w}_2'$, and the ciphertext ct_1 is equal to $\tilde{c}$, then the signature is accepted. Next, we will provide the proofs for these two parts of verification passing.

a)$\| \boldsymbol{z}_1 \|_\infty \leq \gamma_1 - \beta \ \wedge \ \| \boldsymbol{z}_2 \|_\infty \leq \gamma_1 - \beta$

During the signature stage, according to step 16 of the signature algorithm procedure, if any coefficient of $\boldsymbol{z}_1, \boldsymbol{z}_2$ are greater than $\gamma_1 - \beta$, it will lead to the failure of the rejection sampling output, and recalculation will be done with the new masking vector $\boldsymbol{y}_i$. Therefore, the correct signatures output by the signature algorithm procedure will all satisfy $\| \boldsymbol{z} \|_\infty \leq \gamma_1 - \beta$.

b) $\tilde{c}_2 = H(\mu \| \boldsymbol{w}_2' \| ct_1)$

First, according to step 5 of the verification algorithm, we have $\boldsymbol{w}_2' = \boldsymbol{B}\boldsymbol{z}_1 + \boldsymbol{A}\boldsymbol{z}_2 - \boldsymbol{u}c$. Since $\boldsymbol{z}_1 = \boldsymbol{y}_1 + c\boldsymbol{x}_i, \boldsymbol{z}_2 = \boldsymbol{y}_2 + c\boldsymbol{s}_{i,1}$, we can obtain:

$$\begin{aligned} \boldsymbol{w}_2' &= \boldsymbol{B}\boldsymbol{z}_1 + \boldsymbol{A}\boldsymbol{z}_2 - \boldsymbol{u}c \\ &= \boldsymbol{B}(\boldsymbol{y}_1 + c\boldsymbol{x}_i) + \boldsymbol{A}(\boldsymbol{y}_2 + c\boldsymbol{s}_i) - \boldsymbol{u}c \\ &= \boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2 + \boldsymbol{B}c\boldsymbol{x}_i + \boldsymbol{A}c\boldsymbol{s}_i - \boldsymbol{u}c \end{aligned} \tag{1}$$

Then, according to steps 3 and 7 of the key generation algorithm, we know that: $\boldsymbol{g}_i = \boldsymbol{B}\boldsymbol{x}_i$, $\boldsymbol{A}\boldsymbol{s}_i = \boldsymbol{u} - \boldsymbol{g}_i$. Combining with formula (1), we can get:

$$\begin{aligned} \boldsymbol{w}_2' &= \boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2 + \boldsymbol{B}c\boldsymbol{x}_i + \boldsymbol{A}c\boldsymbol{s}_i - \boldsymbol{u}c \\ &= \boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2 + \boldsymbol{g}_i c + \boldsymbol{A}\boldsymbol{s}_i c - (\boldsymbol{A}\boldsymbol{s}_i + \boldsymbol{g}_i)c \\ &= \boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2 + \boldsymbol{g}_i c - \boldsymbol{g}_i c \end{aligned} \tag{2}$$

we can get: $\boldsymbol{w}_2' = \boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2 = \boldsymbol{w}_2$.

2) Therefore, we can obtain $\tilde{c}_2 = H(\mu \| \boldsymbol{w}_2 \| ct_1) = H(\mu \| \boldsymbol{w}_2' \| ct_1)$.

Opening correctness

Based on the security guarantees of the Kyber encryption algorithm specified in FIPS 203 [8], the K-PKE.Decrypt algorithm can recover the identity matrix $\boldsymbol{g}_i$ and the challenge value $\tilde{c}_1$ with overwhelming probability. If the signature was generated by a legitimate group member and is valid, then the identity vector satisfies $\boldsymbol{g}_i \in reg$ according to the registration table. Moreover, from Step 10 of the signing algorithm, we obtain $\boldsymbol{w}_1 = \boldsymbol{B}\boldsymbol{y}_1 = \boldsymbol{B}\boldsymbol{z}_1 - \boldsymbol{g}_i c$ and $\tilde{c}_1 = H(\mu \| \boldsymbol{B}\boldsymbol{z}_1 - \boldsymbol{g}_i c \| ct_1)$. Therefore, the **opening correctness** of the ML-GS scheme is ensured.

5 Security Analysis

To prove that the ML-GS scheme satisfies **anonymity**, we first construct a **simulator algorithm**. If an adversary A is able to break the anonymity of a signature generated by the simulator, then there exists a **challenger** B that can extract a solution to the **M-LWE** problem. The simulator algorithm is shown in Algorithm 6.

Algorithm 6 $\mathrm{Sign}_{\mathrm{Simulator}}(gpk, i, M)$

1. $(\boldsymbol{u}, \boldsymbol{B}) \in \mathrm{R}_q^k \times \mathrm{R}_q^{k\times k} \leftarrow \mathrm{ExpandPK}(\rho)$
2. $\boldsymbol{g}_i = \boldsymbol{B}\boldsymbol{x}_i$
3. $\mu \leftarrow H(\rho \,\|\, M)$
4. $r, r' \leftarrow \mathrm{B}^{64}$
5. $ct_1 \leftarrow \text{K-PKE.Encrypt}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}_i, r)$
6. $\boldsymbol{y}_1, \boldsymbol{y}_2 \leftarrow D_{\gamma_1}^k \times D_{\gamma_1}^l$
7. $\tilde{c}_1 \leftarrow \mathrm{B}^{32}$
8. $\tilde{c}_2 \leftarrow \mathrm{B}^{32}$
9. $ct_2 \leftarrow \text{K-PKE.Encrypt}(\boldsymbol{a}, \boldsymbol{b}, \tilde{c}_1, r')$
10. $\tilde{c} \leftarrow H(\tilde{c}_2 \,\|\, ct_2)$
11. $c \leftarrow \mathrm{SampleInBall}(\tilde{c})$
12. $\boldsymbol{z}_1 \leftarrow D_{\gamma_1}^k$
13. $\boldsymbol{z}_2 \leftarrow D_{\gamma_1}^l$
14. $(\boldsymbol{w}_1, \boldsymbol{w}_2) = (\boldsymbol{B}\boldsymbol{z}_1 - \boldsymbol{g}_i c, \boldsymbol{B}\boldsymbol{z}_1 + \boldsymbol{A}\boldsymbol{z}_2 - \boldsymbol{u}c)$
15. **if** H oracle has queried about $\boldsymbol{w}_1$ or $\boldsymbol{w}_2$ **then**
16. Abort
17. **else**
18. Program $\tilde{c}_1 \leftarrow H(\mu \,\|\, \boldsymbol{w}_1 \,\|\, ct_1)$ and $\tilde{c}_2 \leftarrow H(\mu \,\|\, \boldsymbol{w}_2 \,\|\, ct_1)$
19. **return** $\sigma = (\tilde{c}_2, \boldsymbol{z}_1, \boldsymbol{z}_2, ct_1, ct_2)$

Theorem 1. The statistical distance between the output of the ML-GS signing algorithm and that of a simulated algorithm which does not take the member's secret key as input is **negligible.**

Proof. The only difference between the real signing algorithm and the simulated one lies in the generation of the response value $\boldsymbol{z}_1, \boldsymbol{z}_2$ and the challenge value $\tilde{c}_1, \tilde{c}_2$. In the simulation, the challenge $\tilde{c}_2$ is chosen uniformly at random, whereas in the real signing algorithm is $\tilde{c}_2 = H(\mu \,\|\, \boldsymbol{w}_2 \,\|\, ct_1) = H(\mu \,\|\, \boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2 \,\|\, ct_1)$, the random oracles H are programmed to respond consistently with previous queries. When the value $\boldsymbol{B}\boldsymbol{y}_1 + \boldsymbol{A}\boldsymbol{y}_2$ matches a previously queried value, the simulation algorithm and the real signing algorithm may behave differently. However, the probability of such a collision, denoted as $1/2^{256\tau}$, is negligible. The same reasoning applies to the proof for $\tilde{c}_1$.

In the signing algorithm, the output $\boldsymbol{z}_1, \boldsymbol{z}_2$ obtained through the rejection sampling procedure is statistically indistinguishable from a uniform sample over distribution $S_{\gamma_1-\beta}^k$. Therefore, the statistical

distance between the outputs of the signing algorithm and the simulation algorithm is negligible under the parameters given in this paper.

Theorem 2. The proposed ML-GS scheme satisfies anonymity based on the hardness of the MLWE problem in ROM.

Proof. We prove that the ML-GS scheme satisfies anonymity through a sequence of games.

$Game0$: The challenger B runs the setup and key generation algorithms normally, and gives the group public key gpk and the member's signing key $gsk_i = (\boldsymbol{x}_i, \boldsymbol{s}_i)$. When A makes an opening query $\mathrm{O}_{open}(gpk, gmsk, \cdot)$, returns either $i \in [N]$ or $\perp$. Therefore, A selects two identities i_0, i_1 and message M to send to B. Then, B chooses a bit $\mathrm{b} \in \{0,1\}$ and sends the signature $\sigma \leftarrow \text{ML-GS.Sign}(gpk, gsk_{i_b}, M)$ to A. Finally, A outputs a guess b' for the signer's identity i_b.

$Game1$: This game is identical to $Game0$, except that the signing algorithm is replaced with the simulated algorithm $\mathrm{Sign}_{\mathrm{Simulator}}$.

In **Game1**, the simulated algorithm is used to generate signatures, so the ciphertexts ct_1 and ct_2 are the **only** means by which the attacker can obtain information about the signer's identity, i.e.,

$$ct_1 \leftarrow \text{K-PKE.Encrypt}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}_i, r)$$
$$ct_2 \leftarrow \text{K-PKE.Encrypt}(\boldsymbol{a}, \boldsymbol{b}, \tilde{c}_1, r')$$

$Game2$: The challenger B first runs the key generation algorithm to generate an encryption key pair and sends the public key $(\boldsymbol{a}, \boldsymbol{b})$ to A. Then, A selects two random messages M_0 and M_1 and sends them to B, B randomly selects a bit $\mathrm{b} \in \{0,1\}$ and returns the ciphertext $ct \leftarrow \text{K-PKE.Encrypt}(M_0)$ to A. Finally, A outputs a guess b'. According to FIPS 203 [8], the K-PKE scheme satisfies IND-CPA (indistinguishability under chosen-plaintext attacks) security. Therefore, $\Pr[Game2] \le 1/2 + negl$, and we conclude that $Game1$ does not leak any information about the signer's identity.

Combining the above games, we conclude that $\Pr[Game0] \le 1/2 + negl$. Therefore, the ML-GS scheme satisfies **anonymity**.

Theorem 3. The proposed ML-GS scheme satisfies traceability based on the hardness of the MSIS problem in ROM.

Proof. Assume that the adversary A can break the traceability of the scheme in the game defined by Definition 5 with non-negligible probability. Then we can construct a PPT algorithm C that breaks the MSIS problem with non-negligible probability. When A queries the signing oracle O_{sign}:

If $i \neq j$, then run the honest signing algorithm and return $\text{ML-GS.Sign}(gpk, sk_i, \cdot)$;

If $i = j$, then C runs the simulator algorithm and returns the output of the simulator algorithm $\sigma \leftarrow \mathrm{Sign}_{\mathrm{Simulator}}$.

When the corruption oracle $\mathrm{O}_{corrupt}$ is queried about $i \in [N]$:

If $i \neq j$, then C returns gsk_i;

If $i = j$, then C returns $\perp$.

After the query phase ends, A outputs a signature $\sigma = (\tilde{c}_2, \boldsymbol{z}_1, \boldsymbol{z}_2, ct_1, ct_2)$.

The first case is σ is a valid signature for j. In the ROM, $\mathrm{H}(\mu \| \boldsymbol{w}_1 \| ct_1)$ and $\mathrm{H}(\mu \| \boldsymbol{w}_2 \| ct_1)$ have already been queried, and O_{sign} *returned* $\tilde{c}_1$ and $\tilde{c}_2$ while processing messages from A. Similarly, A can obtain another signature $\sigma' = (\tilde{c}_2, \boldsymbol{z}_1', \boldsymbol{z}_2', ct_1', ct_2')$. We define $\boldsymbol{w}_1' = \boldsymbol{B}\boldsymbol{z}_1' - \boldsymbol{g}_i' c$ *and* $\boldsymbol{w}_2' = \boldsymbol{B}\boldsymbol{z}_1' + \boldsymbol{A}\boldsymbol{z}_2' - \boldsymbol{u}c$, then we have:

$\mathrm{H}(\mu \| \boldsymbol{w}_1 \| ct_1) = \mathrm{H}(\mu' \| \boldsymbol{w}_1' \| ct_1')$; $\mathrm{H}(\mu \| \boldsymbol{w}_2 \| ct_1) = \mathrm{H}(\mu' \| \boldsymbol{w}_2' \| ct_1')$.

In the ROM, we have $(\mu, \tilde{c}_2, \boldsymbol{z}_1, \boldsymbol{z}_2, ct_1, ct_2) \neq (\tilde{c}_2, \boldsymbol{z}_1', \boldsymbol{z}_2', ct_1', ct_2')$,and the following conditions hold:

$\boldsymbol{B}(\boldsymbol{z}_1 - \boldsymbol{z}_1') = 0$; $\boldsymbol{B}(\boldsymbol{z}_1 - \boldsymbol{z}_1') + \boldsymbol{A}(\boldsymbol{z}_2' - \boldsymbol{z}_2) = 0$.

Since $\| \boldsymbol{z}_i' - \boldsymbol{z}_i \|_\infty \leq 2B$ $(i \in 1,2)$, *it follows that* $\boldsymbol{z}_1 - \boldsymbol{z}_1'$ *is a solution to* $\mathrm{MSIS}_{n,m,\beta}$. Therefore, A cannot output a valid signature for j in the game $Game_{ELGS,\mathrm{A}}^{trace}(\lambda)$.

Another case is when the output of the opening algorithm is $\perp$. If $g_i \neq reg$, then we have $\boldsymbol{B}(\boldsymbol{z}_1' - \boldsymbol{z}_1') + \boldsymbol{A}(\boldsymbol{z}_2' - \boldsymbol{z}_2') = \boldsymbol{0} \bmod q$.

Alternatively, when $\tilde{c}_1 \neq H(\mu \| \boldsymbol{B}\boldsymbol{z}_1 - \boldsymbol{g}_i c \| ct_1) \wedge g_i \neq reg$, we have $\boldsymbol{B}(\boldsymbol{z}_1' - \boldsymbol{z}_1') + \boldsymbol{A}(\boldsymbol{z}_2' - \boldsymbol{z}_2') - \boldsymbol{u}(c' - c) = \boldsymbol{0} \bmod q$. Like the proof in the above cases, this will break the security of the MSIS problem.

6 Auxiliary Functions

Next, we provides some subroutines used in the scheme construction presented in Section 4 (e.g., ExpandPK). These Algorithms 7 and 8 are inspired by examples from the FIPS 204 standard [9], with certain modifications made to align with the scheme proposed in this paper. For more fundamental subroutines, such as SampleInBall, please refer to the FIPS 204 standard.

Algorithm 7 ExpandPK(ρ)

Samples a k *vector* $\boldsymbol{u}$ *and a* $k \times k$ *matrix* $\boldsymbol{B}$:

1. **for** i **from** 0 **to** k-1 **do**
2. $\rho' \leftarrow \rho \,\|\, \mathrm{IntegerToBytes}(i, 2)$
3. $\boldsymbol{u}[i] \leftarrow \mathrm{RejNTTPoly}(\rho')$
4. **for** i **from** 0 **to** k-1 **do**
5. **for** j **from** 0 **to** k-1 **do**
6. $\rho' \leftarrow \rho \,\|\, \mathrm{IntegerToBytes}(i,1) \,\|\, \mathrm{IntegerToBytes}(j,1)$
7. $\boldsymbol{B}[i, j] \leftarrow \mathrm{RejNTTPoly}(\rho')$
6. **return** $(\boldsymbol{u}, \boldsymbol{B})$

Algorithm 8 ExpandS(ρ)

Samples a vector $\boldsymbol{x}$ *, each with polynomial coordinates whose coefficients are in* $[-\eta, \eta]$

1. **for** i **from** 0 **to** k-1 **do**
2. $\rho' \leftarrow \rho \,\|\, \mathrm{IntegerToBytes}(i,1)$
3. $\boldsymbol{x}[i] \leftarrow \mathrm{RejBoundedPoly}(\rho')$
4. **return** $\boldsymbol{x}$

7 Efficiency Analysis

7.1 Parameter Setting

Our proposed ML-GS scheme is constructed based on hard problems over lattices. We evaluate its security using two sets of parameters and corresponding estimations, as shown in Table 2. First, under the parameter set **PARM I**, and based on the estimation by enumeration [38], the ML-GS scheme achieves 9**6 bits** post-quantum security level, which meets the **Category I** standard defined by the **NIST**. Similarly, **PARM II** enables our scheme to achieve **154 bit** post-quantum security.

Table 2. The Parameters for ML-GS Scheme

Parameter	Recommended Parameters	
	PARM I	**PARM II**
q	1073738753	1073738753
Q	3329	3329
n	256	256
(k,l)	(4,4)	(6,5)
η	83832	83290
η_1	3	3
τ	39	49
$\gamma_1 \geq 11\tau\sqrt{kn\eta}$	$\approx 2^{31}$	$\approx 2^{31}$
BKZ blocksize b to break SIS	364	583
Classical security	106	170
Quantum security	96	154

7.2 Time efficiency analyzes

The main time-consuming operations in the ML-GS scheme stem from trapdoor generation and sampling algorithms over module lattices. In evaluating the computational efficiency of the ML-GS scheme, we primarily focus on operations that consume significant time, while omitting those with relatively low overhead, such as polynomial addition and hash operations. We employed the Sage Math library and conducted performance tests under a Windows 11 operating system with an AMD Ryzen 5 5600G 3.9 GHz processor. The average runtime of these time-consuming operations was measured over 1000 executions. The computation times for trapdoor generation and sampling algorithms are presented in Table 3.

Table 3. Algorithm Notations and Execution Time (ms)

Notation	Description	Execution time
$T_{M-TrapGen}$	Trapdoor generation algorithm on module lattices	248.09
$T_{M-Sample}$	Trapdoor sampling algorithm on module lattices	31.10

We tested the setup, key generation, signing, and verification algorithms of the ML-GS scheme under two sets of security parameters, and the results are shown in Table 4.

Table 4. The Time Cost for ML-GS Scheme

	Setup	KeyGen	Sign	Verify
PARM I	248.09	56.54	89.04	57.24
PARM II	426.40	96.12	179.67	114.48

Table 5. The comparison of time cost

	KeyGen	Signature	Verify	Security
[12]	T_{Sample}	$3dT_{Mul}+T_S$	$dT_{Mul}+T_S$	CPA-Anonymity
[23]	T_{Sample}	$7dT_{Mul}+T_S$	$2dT_{Mul}+T_S$	CPA-Anonymity
[ours]	$T_{M-Sample}$	$5dT_{Mul}$	$3dT_{Mul}$	CCA-Anonymity

Next, we compare our scheme with two lattice-based GS schemes: Lyubashevsky et al. introduced their scheme in 2022 [12], and Tang et al. proposed theirs in 2024 [23]. The comparison results regarding time computation overhead are shown in Table 5. Here, d denotes the number of polynomial matrix multiplications, T_M represents the total time cost of these multiplications, and T_S and T_V refer to the time required for non-interactive zero-knowledge proof generation and verification, respectively.

To make the comparison results more intuitive, we conducted simulated implementations of the main time-consuming operations for both schemes and the ML-GS scheme. Figure 5 presents the specific comparison of time overhead.

As shown in Figure 5, the ML-GS scheme significantly outperforms existing solutions in all core operations. Key generation is over 75% faster, signing time is reduced by up to 69%, and verification is about 30% faster. These improvements highlight the efficiency and practicality of our scheme for real-time applications.

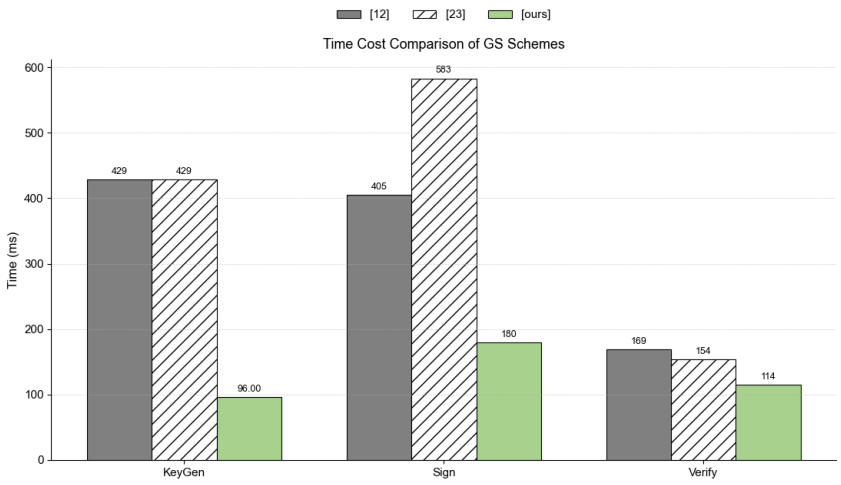

Figure 5. Time Cost Comparison of GS Schemes (ms).

7.3 Storage efficiency analyzes

The sizes of the keys and signatures are calculated as follows:

Group public key. The group public key consists of a public random seed ρ, a public matrix $\boldsymbol{A}$, and a K-PKE key pair $(\boldsymbol{a},\boldsymbol{b})$. Therefore, its size is $32+32kl\lceil\log_2 q\rceil+32k(k+1)\lceil\log_2 Q\rceil$ bits.

Group trapdoor key. The group trapdoor key consists of a secret random seed ρ and a trapdoor matrix $\boldsymbol{T}_A$. Thus, its size is $32+32l^2\lceil\log_2 q\rceil$ bits.

Group tracing key. The group tracing key includes only the K-PKE decryption key $\boldsymbol{s}$. Hence, its size is $32k\lceil\log_2\eta_1\rceil$ bits.

Signing key. The group member's signing secret key consists of a random short vector $\boldsymbol{x}$ and a sampled short vector $\boldsymbol{s}$. To compute the size of $\boldsymbol{s}$, we refer to Lemma 5. Accordingly, the total size is $32k\lceil \log_2 \eta \rceil + 32l\left\lceil \log_2(2\eta\sqrt{d(\log_2 q+2)n}) \right\rceil$ bits.

Signature. A signature σ consists of $(\tilde{c}_2, z_1, z_2, ct_1, ct_2)$. The total size of the signature is $32+32+32(k+l)\log_2(12\gamma_1)+1024k$ bits.

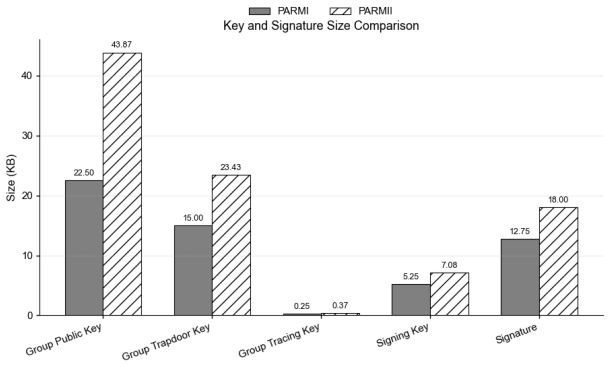

Figure 6. The Store Cost of ML-GS Scheme.

Based on the above calculations of the storage overhead for each key and signature, Figure 6 presents the communication cost of the ML-GS scheme under two parameter sets. As shown in Figure 6, the group public key incurs the largest overhead in the ML-GS scheme, reaching 22.5 KB under PARM I, followed by the group trapdoor key, which costs 15 KB. This substantial cost mainly results from the trapdoor setup and preimage sampling over lattices, which require relatively large parameter settings.

Regarding the signature size, thanks to the signature architecture adopted from FIPS 204, our scheme exhibits a unique advantage among lattice-based group signature schemes. Under the PARM I parameter set that achieves a 96-bit post-quantum security level, the signature size is only 12.75 KB.

Table 6. Comparison of Storage Cost

	Public Key Size	Secret Key Size	Signature Size
[12]	48KB	6KB	92KB
[23]	268.5KB	68.7KB	386.1KB
Ours	44.9KB	7KB	18.0KB

To demonstrate the efficiency of the ML-GS scheme in terms of storage cost, we compare our scheme with scheme [12] and [23], and the comparison results are shown in Table 6. For a fair comparison, we evaluate ML-GS under PARM II, which provides the same post-quantum security level as scheme [12] and [23].

From Table 6, it can be seen that the public key size of our scheme is reduced by approximately 6.46% compared to scheme [12]'s 48KB, and by about 83.32% compared to scheme [23]'s 268.5KB, effectively lowering storage overhead. The secret key size is 7KB, which is slightly increased by 16.67% compared to [12], but reduced by 89.75% compared to [23]'s 68.7KB, maintaining a relatively low key burden. More notably, the signature size is only 18.0KB, representing a reduction of approximately 80.43% and 95.34% compared to the 92KB of [12] and 386.1KB of [23], respectively.

However, compared with some existing group signature schemes based on zero - knowledge proofs, our ML - GS scheme also has some drawbacks when it comes to tracing the identity of group members. That is, the ML - GS.Open algorithm in the opening phase requires a registration list reg as an input parameter. The registration list contains the identity vectors of all registered group members. Therefore, as the number of group members increases, the overhead cost of the opening algorithm for the group administrator will increase.

8 Conclusion

This paper presents a module lattice-based group signature scheme, ML-GS, designed to achieve efficient signing performance and privacy protection while remaining resilient against quantum attacks. The scheme builds upon the standardized post-quantum signature algorithm FIPS 204 and incorporates a dual-Dilithium signing algorithm along with the K-PKE encryption scheme from FIPS 203. By employing a "sign-hybrid-encrypt"structure, ML-GS achieves both member anonymity and traceability. Under the Random Oracle Model, we prove that ML-GS satisfies anonymity and traceability based on the hardness of the Module Learning With Errors and Module Short Integer Solution assumptions. Compared to existing group signature schemes at the same security level, ML-GS achieves a 6% reduction in public key size and an 80% reduction in signature size, with only a slight increase in private key size, demonstrating its practicality and promise in constructing efficient and scalable group signature systems.

As future work, we aim to address the communication overhead introduced using the registration list in the opening algorithm. We also plan to further optimize the communication and computational efficiency of the scheme and explore its deployment in real-world privacy-preserving applications.

Funding

This work was supported by the Beijing Natural Science Foundation (Grant No. 4232034) and the Fundamental Research Funds for the Central Universities (Grant Nos. 3282024058 and 3282024052).

References

[1] Dong, Shi, et al. "Improved PBFT Consensus Mechanism Based on Voting Sort Clustering Partition With Group Signature for IoT." *IEEE Transactions on Intelligent Transportation Systems* (2024).

[2] Jayashree, S., and SVN Santhosh Kumar. "An efficient group signature based certificate less verification scheme for vehicular ad-hoc network." *Wireless Networks* 30.5 (2024): 3269-3298.

[3] Wang, Dong, et al. "A Novel Group Signature Scheme with Time-Bound Keys for Blockchain." International Conference on Web Information Systems and Applications. Singapore: Springer Nature Singapore, 2023.

[4] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer." *SIAM review* 41.2 (1999): 303-332.

[5] Montanaro, Ashley. "Quantum algorithms: an overview." *npj Quantum Information* 2.1 (2016): 1-8.

[6] Ajtai, Miklós. "Generating hard instances of lattice problems." *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996.

[7] Gentry, Craig. "Fully homomorphic encryption using ideal lattices." *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009.

[8] FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. https://doi.org/10.6028/NIST.FIPS.203

[9] FIPS 204: Module-Lattice-Based Digital Signature Standard. https://doi.org/10.6028/NIST.FIPS.204

[10] FIPS 205: Stateless Hash-Based Digital Signature Standard. https://doi.org/10.6028/NIST.FIPS.205

[11] Lyubashevsky, Vadim, et al. "Crystals-dilithium." *Algorithm Specifications and Supporting Documentation* (2020).

[12] Lyubashevsky, Vadim, Ngoc Khanh Nguyen, and Maxime Plançon. "*Lattice-based zero-knowledge proofs and applications: shorter, simpler, and more general*." Annual International Cryptology Conference. Cham: Springer Nature Switzerland, 2022.

[13] Gordon, S. Dov, Jonathan Katz, and Vinod Vaikuntanathan. "A group signature scheme from lattice assumptions." *International conference on the theory and application of cryptology and information security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[14] Laguillaumie, Fabien, et al. "Lattice-based group signatures with logarithmic signature size." *International conference on the theory and application of cryptology and information security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[15] Ling, San, Khoa Nguyen, and Huaxiong Wang. "Group signatures from lattices: simpler, tighter, shorter, ring-based." *IACR International Workshop on Public Key Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[16] Ling, San, et al. "*A lattice-based group signature scheme with verifier-local revocation*." Theoretical Computer Science 730 (2018): 1-20.

[17] Ducas, Léo, and Daniele Micciancio. “Improved short lattice signatures in the standard model.” *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference*, *Santa Barbara, CA*, *USA*, August 17-21, 2014, Proceedings, Part I 34. Springer Berlin Heidelberg, 2014.

[18] Ling, San, et al. “*A lattice-based group signature scheme with verifier-local revocation*.” Theoretical Computer Science 730 (2018): 1-20.

[19] Preethi, Thakkalapally, and B. B. Amberker. “Lattice-based group signature scheme without random oracle.” *Information Security Journal: A Global Perspective* 29.6 (2020): 366-381.

[20] Tang, Yongli, et al. “Lattice-Based Group Signature with Message Recovery for Federal Learning.” *Applied Sciences* 13.15 (2023): 9007.

[21] Ağırtaş, Ahmet Ramazan, and Oğuz YAYLA. “A Lattice-based Accountable Subgroup Multi-signature Scheme with Verifiable Group Setup.” *Cryptology ePrint Archive* (2024).

[22] Katsumata, Shuichi, and Shota Yamada. “Group signatures without NIZK: from lattices in the standard model.” *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38. Springer International Publishing, 2019.

[23] Tang, Yongli, et al. “Event-oriented linkable group signature from lattice.” *IEEE Transactions on Consumer Electronics* (2024).

[24] Zhang, Ping, et al. “Lattice-based group signature scheme and its application in IoMT.” *Journal of Communications and Networks* 27.2 (2025): 59-69.

[25] Şahin, Meryem Soysaldı, and Sedat Akleylek. “A survey of quantum secure group signature schemes: Lattice-based approach.” *Journal of Information Security and Applications* 73 (2023): 103432.

[26] Chen, Simin, et al. “Constant-size group signatures with message-dependent opening from lattices.” *International Conference on Provable Security*. Cham: Springer Nature Switzerland, 2023.

[27] Amir, N. A. S., Othman, W. A. M., & Wong, K. B. (2023). Securing an authenticated privacy preserving protocol in a group signature scheme based on a group ring. *Mathematics*, 11(18), 3918.

[28] Chen, X., Huang, J., Xiao, K., Li, H., & Huang, Q. (2025). A Non-Interactive Identity-Based Multi-Signature Scheme on Lattices with Public Key Aggregation. *IEEE Transactions on Dependable and Secure Computing*.

[29] Jayashree, S., & Kumar, S. S. (2024). An efficient group signature-based certificate less verification scheme for vehicular ad-hoc network. *Wireless Networks*, 30(5), 3269-3298.

[30] Yu, Yang, Huiwen Jia, and Xiaoyun Wang. “Compact lattice gadget and its applications to hash-and-sign signatures.” *Annual International Cryptology Conference*. Cham: Springer Nature Switzerland, 2023.

[31] Jeudy, Corentin, Adeline Roux-Langlois, and Olivier Sanders. “Lattice signature with efficient protocols, application to anonymous credentials.” *Annual International Cryptology Conference*. Cham: Springer Nature Switzerland, 2023.

[32] Bert, Pauline, et al. “Implementation of lattice trapdoors on modules and applications.” *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021*, Daejeon, South Korea, July 20–22, 2021, Proceedings 12. Springer International Publishing, 2021.

[33] Bos, Joppe, et al. “CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM.” 2018 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2018.

[34] Langlois, Adeline, and Damien Stehlé. “Worst-case to average-case reductions for module lattices.” Designs, Codes and Cryptography 75.3 (2015): 565-599.

[35] Lyubashevsky, Vadim, et al. “Shorter lattice-based group signatures via “almost free” encryption and other optimizations.” *International Conference on the Theory and Application of Cryptology and Information Security*. Cham: Springer International Publishing, 2021.

[36] Lyubashevsky, Vadim. “Lattice signatures without trapdoors.” *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[37] Bellare, Mihir, Haixia Shi, and Chong Zhang. “*Foundations of group signatures: The case of dynamic groups.” Cryptographers’ track at the RSA conference*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[38] Albrecht, Martin R., et al. “Estimate all the {LWE, NTRU} schemes!.” *Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy*, September 5–7, 2018, Proceedings 11. Springer International Publishing, 2018.