A Dilithium-like Multisignature in Fully Split Ring and Quantum Random Oracle Model

Shimin Pan[1], Tsz Hon Yuen[2], and Siu-Ming Yiu[1]

[1] Department of Computer Science, University of Hong Kong, Hong Kong, China
smpan@connect.hku.hk
smyiu@cs.hku.hk
[2] Department of Software Systems and Cybersecurity, Monash University, Clayton, VIC 3168, Australia john.tszhonyuen@monash.edu

Abstract. Multisignature schemes are crucial for secure operations in digital wallets and escrow services within smart contract platforms, particularly in the emerging post-quantum era. Existing post-quantum multisignature constructions either do not address the stringent requirements of the Quantum Random Oracle Model (QROM) or fail to achieve practical efficiency due to suboptimal parameter choices.

In this paper, we present a novel Dilithium-based multisignature scheme designed to be secure in the QROM and optimized for practical use. Our scheme operates over the polynomial ring $\mathbb{Z}_q[X]/(x^n + 1)$ with $q \equiv 1 \pmod{2n}$, enabling full splitting of the ring and allowing for efficient polynomial arithmetic via the Number Theoretic Transform (NTT). This structure not only ensures post-quantum security but also bridges the gap between theoretical constructs and real-world implementation needs.

We further propose a new hardness assumption, termed ν-SelfTargetMSIS, extending SelfTargetMSIS (Eurocrypt 2018) to accommodate multiple challenge targets. We prove its security in the QROM and leverage it to construct a secure and efficient multisignature scheme. Our approach avoids the limitations of previous techniques, reduces security loss in the reduction, and results in a more compact and practical scheme suitable for deployment in post-quantum cryptographic systems.

Keywords: Multisignature, Dilithium, NTT, Quantum Random Oracle Model

1 Introduction

With the development of science and technology, it becomes more and more realistic that quantum computers will come to our life sooner or later. Since the traditional cryptographic system will be broken when quantum computers come, the US National Institute of Standards and Technology (NIST) announced three selected digital signatures in the post-quantum signature standardization on July 5, 2022: CRYSTALS-Dilithium, FALCON, and SPHINCS+. On August 13, 2024, NIST published the Federal Information Processing Standard (FIPS) 204 for

the CRYSTALS-Dilithium algorithm [20], which has been renamed as Module-Lattice-Based Digital Signature Algorithm (ML-DSA). CRYSTALS-Dilithium is the one with more studies in academia because of its good structure and the worst-case to average-case reduction.

Quantum computers have not only the ability to attack traditional cryptosystems, but also the power to query a function with multiple inputs at *a single query*. Currently, most signature schemes rely on a hash function to turn message(s) into challenge(s), and thus sign them. However, many of them use the random oracle model (ROM) to simulate the output of a hash function in their security proofs (e.g., [14]). When we consider the case of a quantum adversary, it always makes a query in superposition, which makes it hard for the simulator to find out what preimage the adversary is interested in. More threateningly, the adversary could make an attack on superposition that comprises different messages in one move. This uncertainty of such powerful adversaries calls eagerly for the security reductions over the quantum random oracle model (QROM) [4].

QROM-based Multisignatures. In this paper, we consider the case of *multisignature* - an aggregation of signatures to show that multiple users agree on the same message. It is widely used in cryptocurrency and blockchain technologies. For the case of a digital wallet for cryptocurrency, a wallet could require multisignatures (from any two public keys out to three) to authorize spending. So even if one secret key is stolen, the attacker cannot access the funds. It is also useful for escrow services in a smart contract. With a 2-out-of-3 multisignature with a buyer, seller, and a neutral arbitrator, funds are only released when two of the three agree, avoiding the need to trust a single party.

Currently, there already exist some multisignature solutions based on lattice-based hard problems [5, 6, 8, 11, 15], but few have the support for the quantum random oracle model [11, 15]. Such a need brings us to QROM-based multisignatures.

Towards practical Multisignatures. Dilithium [10] is a post-quantum secure signature scheme constructed via the "Fiat-Shamir with aborts" framework [17]. It is computed in a polynomial ring $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^n + 1)$, where q is a prime modulus. A QROM-based Dilithium was proposed in [14].

For the current QROM-based multisignature relying on Dilithium [11], the chosen parameters are good for invertibility and make the security proof easy. More specifically, it sets the prime number by $q \equiv 5 \pmod 8$. The choice of this parameter started from [19], with the underlying polynomial ring being $\mathbb{Z}_q[X]/(X^n + 1)$ and n being a power of two. Moreover, the invertibility comes from the isomorphism $\mathbb{Z}_q[X]/(X^n + 1) \cong \mathbb{Z}_q[X]/(X^{n/2} - r) \times \mathbb{Z}_q[X]/(X^{n/2} + r)$ where the $\mathbb{Z}_q[X]/(X^{n/2} \pm r)$ are fields because both of them are unsplittable. However, this structure is not good for multiplication implementation. A naive polynomial multiplication is $O(n^2)$. It is hard to take Number-Theoretic Transform (NTT) acceleration similar to the underlying Dilithium, which is only $O(n \log n)$.

Practical Dilithium would always choose a faster ring with $q \equiv 1 \pmod{2n}$ as in [3]. The ring is fully split as $\mathbb{Z}_q[X]/(X^n + 1) \cong \mathbb{Z}_q^{\times n}$, and NTT could

make good use of this isomorphism [3]. The hardness of this scheme is based on the hardness of three computational problems: Module Learning with Errors (MLWE), Module Short Integer Solution (MSIS), and SelfTargetMSIS. MLWE and MSIS are widely believed to be secure. In 2024, Jackson et al. [13] presented a reduction of the hardness of SelfTargetMSIS to MLWE in the QROM model. In this paper, we extend the assumption of SelfTargetMSIS, and give the security proof of a multisignature with compact parameters.

1.1 Related works

Researches on QROM. There have already been some researches on QROM in the past decade. Following up with [22], lots of researchers agree that it is almost impossible to rewind the quantum machine, compared to the classical random oracle model, and it is inefficient to prove 2-soundness and similar properties.

After Zhandry [23] remarked that QROM could be easily simulated by $2q$-wise independent functions, there are a lot of trials presenting more reliable properties of QROM and building schemes based on those properties. For example, it is shown in [21] how to adaptively reprogram a hash function. But still, the simulation, extraction, and rewinding are considered to be hard because of the superposition of quantum queries [16].

In 2019, Zhandry [24] proposed the compressed oracle model. This model uses lazy sampling and supposes that the hash outcome $\mathsf{H}(x)$ of any input x is a uniform superposition like $\sum |y\rangle$ before taking measurements. If anybody wants to know the result, it must measure this register and further impact the register of a simulator. It gives the simulator more power to control and guess the inputs and outputs of hash queries. In [7], the authors provided an efficient simulation of the compressed oracle.

QROM-Dilithium and NTT. The QROM-Dilithium was raised in [14], in which the security could rely on either lossy key generation or SelfTargetMSIS. The lossy key generation relies firmly on $q \equiv 5 \pmod 8$, while the SelfTargetMSIS reduction could cater to different needs. However, it only showed that SelfTargetMSIS is secure in the random oracle model in [14]. In 2024, [13] showed a hardness proof on SelfTargetMSIS in the QROM model. Using [9], the authors showed SelfTargetMSIS is at least as hard as MLWE in the given parameter. Furthermore, it demonstrated that using $q \equiv 1 \pmod{2n}$ with NTT acceleration would accelerate the original QROM-Dilithium several times.

Other QROM Multisignature schemes. To the best of the authors' knowledge, there are two researches on QROM multisignatures. The first one [11] provides a simple extension of the Dilithium to multisignature. It uses [21] as the main reprogramming theory. After reprogramming, the authors proved that the schemes's unforgeability under chosen message attack (MS-UF-CMA) is almost equivalent to the unforgeability under no message attack (MS-UF-NMA). And to prove the MS-UF-NMA of the signature scheme, the authors used the MLWE assumption to make the keys lossy. Then the authors presented a gen-

eral search problem with bounded model such that no quantum adversary could easily attack the lossy MS-UF-NMA of the scheme.

The second QROM multisignatures [15] provides a two-round solution on multisignature, which relies on [21] for online-extractability. But it still uses a ring that does not provide fast NTT acceleration.

1.2 Our Contributions

There are two main contributions in our paper. To illustrate our contribution, we define some notations first. Apart from the ring $\mathcal{R}_q$ with $q \equiv 1 \pmod{2n}$ and n is a power of two, we also define $\mathcal{R} := \mathbb{Z}[X]/(X^n + 1)$. For any $a \in \mathbb{Z}_q$, a' mod $^{\pm}q$ is defined by the unique $-(q-1)/2 \leq a' \leq (q-1)/2$ such that $a' \equiv a \pmod{q}$. For any $r = r_0 + r_1 X + \cdots + r_{n-1}X^{n-1} \in \mathcal{R}_q$, we use the notation $\|r\|_i := |r_i \mod^{\pm} q|$ and $\|r\|_\infty := \max_i \|r\|_i$. For $\mathbf{r} = (r_1, \cdots, r_m) \in \mathcal{R}_q^m$, we have $\|\mathbf{r}\|_\infty := \max_i \|r_i\|_\infty$. For $\tau \in \mathbb{N}$, B_τ denotes the set of all elements in $r \in \mathcal{R}_q$ with $\|r\|_\infty = 1$ and the number of ± 1 coefficients is indeed τ. $\mathbf{I}_m$ is an identity matrix of size m.

Contribution 1. We propose a variant to the SelfTargetMSIS, named ν-SelfTargetMSIS, and present a hardness proof for this variant. In the original SelfTargetMSIS, the adversary is given a matrix $\mathbf{A} \in \mathcal{R}_q^{m \times k}$ with a hash function $\mathsf{H} : \{0,1\}^* \to B_\tau$, it needs to find out a message M and corresponding $\mathbf{y} \in \mathcal{R}^{m+k}$ such that $\|\mathbf{y}\|_\infty$ is small, $\mathsf{H}(\begin{bmatrix}\mathbf{I}_m & \mathbf{A}\end{bmatrix}\mathbf{y}, M) = \mathbf{y}_{m+k}$, of which $\mathbf{y}_{m+k}$ is the $(m+k)$-th element of $\mathbf{y}$.

However, single hash target $\mathbf{y}_{m+k}$ is not suitable for multisignature. In our variant ν-SelfTargetMSIS, the adversary needs to find out a solution for ν different targets at the same time. Precisely, it is given $\mathbf{A} \in \mathcal{R}_q^{m \times k}$ and hash function $\mathsf{H} : \{0,1\}^* \to B_\tau$. The adversary needs to find out a solution $\mathbf{A}' \in \mathcal{R}_q^{m \times (\nu-1)}$, M, and $\mathbf{y} \in \mathcal{R}^{m+k+\nu-1}$ such that $\|\mathbf{y}\|_\infty$ is small, and $\forall i \in [\nu], \mathsf{H}(i, \mathbf{A}', \begin{bmatrix}\mathbf{I}_m & \mathbf{A} & \mathbf{A}'\end{bmatrix}\mathbf{y}, M) = \mathbf{y}_{(m+k-1)+i}$.

More importantly, we prove ν-SelfTargetMSIS could achieve almost the same security level as SelfTargetMSIS in the QROM. Thus, it could further use the technique from choosing parameters in SelfTargetMSIS to the one in ν-SelfTargetMSIS.

Contribution 2. We propose a multisignature scheme based on [11]. Their scheme relies on the security of $q \equiv 5 \pmod 8$ at Lemma 4.6 of [14], and further on Lemma 2.2 of [18]. Although the lemma provides invertibility, it is insufficient for NTT acceleration. We prove that our multisignature scheme is secure at $q \equiv 1 \pmod{2n}$, by using ν-SelfTargetMSIS.

Moreover, [11] uses a lemma for getting reprogrammable from [21]. The lemma in [21] is stateless and simulated by a polynomial. It further implies the underlying hash function H_2 is length-preserving. And it also needs an additional almost compressed oracle to help [16], which brings more losses. To get rid of these limitations, we replace the [21] reprogramming with [12]. This change also reduces MS-UF-NMA to MS-UF-CMA of the multisignature and gets rid of the almost compressed oracle, which gives a better loss.

2 Preliminaries

Notations. For $\eta \in \mathbb{N}$, the notation $S_\eta \in \mathcal{R}_q$ is the set of all polynomials such that the coefficients all belong to $\{-\eta, \ldots, \eta\}$. Besides, we use $\mathbf{Adv}^{\mathsf{SCHEME}}_{\mathsf{params}}(\mathcal{A}) := \mathsf{Pr}[x \text{ passes the verification} | x \leftarrow \mathcal{A}]$ to denote the advantage that an adversary $\mathcal{A}$ could break a search version hard problem of the scheme SCHEME on forging. We use $\mathsf{negl}(\lambda)$ to represent a negligible function in λ.

2.1 Security Model of Multisignature

In multisignature with the maximum signer size τ, every individual participant could generate their own key pair, participate in *interactively* signing, and make verifications. We use the notation similar to [5], without using key aggregation and online/offline signing.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ takes a security parameter 1^λ as input, and outputs public parameters. Throughout, we assume that pp is given as implicit input to all other algorithms.
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}()$ generates a secret key sk and a public key pk.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathbf{PK} := \{\mathsf{pk}\}, m)$ signs interactively on message m with signers who own the secret keys of $\mathbf{PK}$ where $|\mathbf{PK}| \leq \tau$, and outputs a signature σ.
- $\{0, 1\} \leftarrow \mathsf{Verify}(\mathbf{PK}, \sigma, m)$ verifies whether the signature σ is sign by the secret owners of all $\mathbf{PK}$ on the message m and $|\mathbf{PK}| \leq \tau$.

Definition 1. *A multisignature scheme Scheme is MS-UF-CMA safe if the advantage any adversary $\mathcal{A}$ wins the given experiment with negligible probability.*

$$\mathsf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{Scheme}} := \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{Scheme}}(\mathcal{A}, \lambda)\right] \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Exp}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{Scheme}}(\mathcal{A}, \lambda)$ is defined as

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KeyGen}()$.
2. $(\mathbf{PK}^*, \sigma^*, m^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{pk}^*)$.
3. *Return* $\mathsf{Verify}(\mathbf{PK}^*, \sigma^*, m^*) = 1 \wedge \mathsf{pk}^* \in \mathbf{PK} \wedge (\mathbf{PK}^*, m^*) \notin \mathcal{Q}$.

The oracle $\mathcal{O}_{\mathsf{Sign}}(\mathbf{PK}, m)$ aborts if $\mathsf{pk}^* \notin \mathbf{PK}$. Otherwise, it sets the queried set $\mathcal{Q} \leftarrow \mathcal{Q} \cup (\mathbf{PK}, m)$ and answers all interactive queries by $\mathsf{Sign}(\mathsf{sk}^*, \mathbf{PK}, m)$.

Definition 2. *A multisignature scheme's MS-UF-NMA is the same as MS-UF-CMA except that no $\mathcal{O}_{\mathsf{Sign}}$ query could be made.*

2.2 Quantum random oracle model (QROM)

A state of a qubit $|\phi\rangle$ could be expressed as a two-dimensional complex vector $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $\{|0\rangle, |1\rangle\}$ is an orthogonal basis of $\mathbb{C}^2$ and $|\alpha|^2 + |\beta|^2 = 1$. The qubit is in superposition when $0 < |\alpha| < 1$. Moreover, the state of n qubits could be written as $|\phi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$ in $\mathbb{C}^{2^n}$ and $\sum_x |\alpha_x|^2 = 1$. When

measuring the state in the computational basis, it outputs x with probability $|\alpha_x|^2$.

One of the most important observations is that it is able to replace any hash function by a fixed polynomial following the Lemma 1.

Lemma 1 (Lemma in [23]). *Let $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$. There is no quantum polynomial time (QPT) adversary $\mathcal{A}^{\mathsf{H}}$ that could distinguish a real hash $|\mathsf{H}\rangle$ and $2Q$-wise independent function f_{2Q} when making at most Q quantum queries to the random oracle. And the f_{2Q} could be treated as a $2Q$ degree polynomial in $\mathbb{F}_{\mathcal{Y}}$.*

2.3 Lattice

In the original QROM-Dilithium [14], the security level of the Dilithium protocol is given by the advantages $\mathbf{Adv}^{\mathsf{MLWE}}_{k,l,\eta}(\mathcal{B})$, $\mathbf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathsf{H},\tau,k,\ell+1,\zeta}(\mathcal{C})$, and $\mathbf{Adv}^{\mathsf{MSIS}}_{k,l,\zeta'}(\mathcal{D})$. We rephrase the definition of these hardness assumptions in the style of [13].

Definition 3 (Module Learning with Errors (MLWE)). *For $m, k, \eta \in \mathbb{N}$, the advantage that any adversary $\mathcal{A}$ breaking the hard assumption is*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{MLWE}}_{m,k,\eta}(\mathcal{A}) := \|&\mathsf{Pr}\left[b = 0 \middle| \mathbf{A} \leftarrow \mathcal{R}_q^{m\times k}, t \leftarrow \mathcal{R}_q^m, b \leftarrow \mathcal{A}(\mathbf{A}, t)\right] - \\ &\mathsf{Pr}\left[b = 0 \middle| \mathbf{A} \leftarrow \mathcal{R}_q^{m\times k}, (\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^k \times S_\eta^m, t \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2, b \leftarrow \mathcal{A}(\mathbf{A}, t)\right]\|.\end{aligned}$$

Definition 4 (Module Short Integer Solutions (MSIS)). *For $m, k, \gamma \in \mathbb{N}$, the advantage that any adversary $\mathcal{A}$ breaking the MSIS is defined as*

$$\mathbf{Adv}^{\mathsf{MSIS}}_{m,k,\gamma}(\mathcal{A}) := \mathsf{Pr}\left[0 < \|\mathbf{y}\|_\infty \le \gamma \wedge [\mathbf{I}_m|\mathbf{A}]\mathbf{y} = \mathbf{0} \middle| \mathbf{A} \leftarrow \mathcal{R}_q^{m\times k}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A})\right].$$

Definition 5 (SelfTargetMSIS). *For $\tau, m, k, \gamma \in \mathbb{N}$ and $\mathsf{H} : \{0,1\}^* \to B_\tau \subset \mathcal{R}_q$, the advantage that any adversary $\mathcal{A}$ breaking the SelfTargetMSIS is defined as*

$$\begin{aligned}&\mathbf{Adv}^{\mathsf{SelfTargetMSIS}}_{\mathsf{H},\tau,m,k,\gamma}(\mathcal{A}) := \\ &\qquad\mathsf{Pr}\left[\begin{matrix}\|\mathbf{y}\|_\infty \le \gamma \\ \wedge\, \mathsf{H}([\mathbf{I}_m|\mathbf{A}]\mathbf{y}, M) = c\end{matrix} \middle| \mathbf{A} \leftarrow \mathcal{R}_q^{m\times k}; \left(\mathbf{y} := \begin{bmatrix}\mathbf{r}\\ c\end{bmatrix}, M\right) \leftarrow \mathcal{A}^{|\mathsf{H}\rangle}(\mathbf{A})\right].\end{aligned}$$

The adversary has quantum access to the H hash function.

Definition 6 (rejected-MLWE (rMLWE) [11]). *For $k, l, \gamma, \beta, \eta, \tau \in \mathbb{N}$ and $\mathsf{H} : \{0,1\}^* \to B_\tau$, the advantage that any adversary breaking the rMLWE is*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{rMLWE}}_{k,l,\gamma,\beta,\eta,\tau}(\mathcal{A}) := \Bigg\| &\mathsf{Pr}\begin{bmatrix}\mathcal{A}(\mathbf{A}, \mathbf{w}, c) = 0 \\ or \\ \mathbf{y} + c\mathsf{s} < \gamma - \beta\end{bmatrix}\begin{matrix}\mathbf{A}, \mathbf{s} \leftarrow \mathcal{R}_q^{k\times l} \times S_\eta^l \\ \mathbf{y}, c \leftarrow S_{\gamma-1}^l \times B_\tau, \\ \mathbf{w} \leftarrow \mathbf{A}\mathbf{y}\end{matrix} \\ &- \mathsf{Pr}\begin{bmatrix}\mathcal{A}(\mathbf{A}, \mathbf{w}, c) = 0 \\ or \\ \mathbf{y} + c\mathsf{s} < \gamma - \beta\end{bmatrix}\begin{matrix}\mathbf{A}, \mathbf{s} \leftarrow \mathcal{R}_q^{k\times l} \times S_\eta^l \\ \mathbf{y}, c \leftarrow S_{\gamma-1}^l \times B_\tau, \\ \mathbf{w} \leftarrow \mathcal{R}_q^k\end{matrix}\Bigg\|.\end{aligned}$$

2.4 QROM-Dilithium

Here we give a brief revisit of QROM-Dilithium in [14].

- KeyGen. This algorithm first picks a random $\rho \leftarrow \{0,1\}^{256}$, and extends it to $\mathbf{A} \leftarrow \mathcal{R}_q^{k\times l} := \mathsf{Sam}(\rho)$. It picks $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} S_\eta^l \times S_\eta^k$, and sets the public number $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. By using the key compression $\mathbf{t}_H \leftarrow \mathsf{Power2Round}_q(\mathbf{t}, d)$, $\mathbf{t}_L := \mathbf{t} - \mathbf{t}_H 2^d$, the $\mathbf{t}_L$ is used in singing only and treated as secret key. It outputs $\mathsf{pk} := (\rho, \mathbf{t}_H)$, $\mathsf{sk} := (\rho, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_L)$.
- Sign. On input a message m and sk, this algorithm reconstructs $\mathbf{A} \leftarrow \mathsf{Sam}(\rho)$. Then it computes $\mathbf{w} := \mathbf{A}\mathbf{y}$ where the random number $\mathbf{y}$ is picked by $\mathbf{y} \leftarrow S_{\gamma_1 - 1}^{\ell}$. It sets $\mathbf{w}_H := \mathsf{HighBits}_q(\mathbf{w}, 2\gamma)$ and picks $c \leftarrow \mathsf{H}(\mathsf{pk}, \mathbf{w}_H, m)$. It computes $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$. If $\|\mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)\|_\infty > \gamma_2 - \beta$ or $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$, then it restarts the algorithm. It finally gets the hint by $\mathbf{h} := \mathsf{MakeHint}_q(-c\mathbf{t}_L, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_L, 2\gamma_2)$. It outputs the signature $\sigma = (\mathbf{z}, \mathbf{w}_H, \mathbf{h})$.
- Verify. On input pk, a message m and a signature σ, this algorithm reconstructs $\mathbf{A} \leftarrow \mathsf{Sam}(\rho)$. It gets the challenge $c \leftarrow \mathsf{H}(\mathsf{pk}, \mathbf{w}_H, m)$. And it later outputs success 1 if $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ and $\mathbf{w}_H = \mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - 2^d c\mathbf{t}_H, 2\nu\gamma_2)$; it outputs fail 0 otherwise.

Here the supporting algorithms are defined in Figure 1, which extract "higher-order" and "lower-order" bits of elements in $\mathbb{Z}_q$ from [10].

Lemma 2 (Lemma 4.1 in [14]). *When setting the parameters as $q > 2\alpha$, $q \equiv 1 \pmod{\alpha}$, $\mathbf{r}, \mathbf{z} \in \mathcal{R}_q$, $\|\mathbf{z}\|_\infty \leq \alpha/2$, and α is even.*

- $\mathsf{UseHint}_q(\mathsf{MakeHint}_q(\mathbf{z}, \mathbf{r}, \alpha), \mathbf{r}, \alpha) = \mathsf{HighBits}_q(\mathbf{r} + \mathbf{z}, \alpha)$.
- $\|\mathbf{r} - \mathbf{v}_H \alpha\|_\infty \leq \alpha + 1$, *where* $\mathbf{v}_H = \mathsf{UseHint}_q(\mathbf{h}, \mathbf{r}, \alpha)$.
- $\mathsf{UseHint}_q(\mathbf{h}, \mathbf{r}, \alpha) = \mathsf{UseHint}_q(\mathbf{h}', \mathbf{r}, \alpha)$ *leads to* $\mathbf{h} = \mathbf{h}'$.

Lemma 3 (Lemma 4.2 in [14]). $\mathsf{HighBits}_q(\mathbf{r}, \alpha) = \mathsf{HighBits}_q(\mathbf{r} + \mathbf{s}, \alpha)$ *when* $\|\mathbf{s}\|_\infty \leq \beta$ *and* $\|\mathsf{LowBits}_q(\mathbf{r}, \alpha)\|_\infty \leq \frac{\alpha}{2} - \beta$.

2.5 Compressed oracles and reprogramming

The compressed oracle is built to record all the information made by the adversary. Unlike the classical method, the update and response procedures are all considered in the superposition. More precisely, if the adversary makes $\sum |x\rangle$ to a random oracle, the simulator could make it entangled with its own database D by answering $\sum_{x,u,D} |x, u\rangle \times |D\rangle$ with $\sum_{x,u,D} |x, u \oplus D(x)\rangle \times |D\rangle$. When the adversary tries to obtain the hash of a single value, it needs to measure $|x\rangle$ register and consequently fix $D(x)$ in the simulator.

The reprogrammability inside quantum random oracles is studied in [2]. Although this paper gets a better bound on the loss, we require an adaptive variant such that the reprogrammed domain could have been impacted by the adversary. The variant we use in our paper is as follows [12], and it is built on top of a compressed oracle. Here, the probability difference is the advantage that any distinguisher knows the oracle has been reprogrammed.

Input: $r \in \mathbb{Z}_q, d \in \mathbb{N}$
$r = r \mod^+ q$
$r_L = r \mod^\pm 2^d$
return $(r - r_L)/2^d$
Algorithm 1: $\mathsf{Power2Round}_q$

Input: $h \in \{0,1\}, r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N}$
$m = (q-1)/\alpha$
$(r_H, r_L) \leftarrow \mathsf{Decompose}_q(r, \alpha)$
if $h = 0$ **then**
  return r_H
else if $r_L > 0$ **then**
  return $(r_H + 1) \mod^+ m$
else
  return $(r_H - 1) \mod^+ m$
end
Algorithm 2: $\mathsf{UseHint}_q$

Input: $z \in \mathbb{Z}_q, r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N}$
$r_H \leftarrow \mathsf{HighBits}_q(r, \alpha)$
$v_H \leftarrow \mathsf{HighBits}_q(r + z, \alpha)$
if $r_H = v_H$ **then**
  return 0
else
  return 1
end
Algorithm 3: $\mathsf{MakeHint}_q$

Input: $r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N}$
$r = r \mod^+ q$
$r_L = r \mod^\pm \alpha$
if $r - r_L = q - 1$ **then**
  return $(0, r_L - 1)$
else
  return $((r - r_L)/\alpha, r_L)$
end
Algorithm 4: $\mathsf{Decompose}_q$

Input: $r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N}$
$(r_H, r_L) \leftarrow \mathsf{Decompose}_q(r, \alpha)$
return r_H
Algorithm 5: $\mathsf{HighBits}_q$

Input: $r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N}$
$(r_H, r_L) \leftarrow \mathsf{Decompose}_q(r, \alpha)$
return r_L
Algorithm 6: $\mathsf{LowBits}_q$

Fig. 1. Supporting algorithms

Lemma 4 (Proposition 1 in [12]). *Let X_1, X_2, Y be finite sets, $\mathcal{D}$ be any distinguisher making at most q queries to H, and R be the number of reprogramming points. The underlying hash function is $\mathsf{H}: X_1 \times X_2 \to Y$.*

$$\left|\mathsf{Pr}[\mathsf{REPROG}_1^{\mathcal{D}} = 1] - \mathsf{Pr}[\mathsf{REPROG}_0^{\mathcal{D}} = 1]\right| \leq \frac{3R}{2}\sqrt{\frac{q}{|X_1|}},$$

in which the REPROG_b game for $b = 0/1$ is defined by

1. $\mathcal{O}_0 \leftarrow_\$ Y^{X_1 \times X_2}$.
2. $\mathcal{O}_1 := \mathcal{O}_0$.
3. $b' \leftarrow \mathcal{A}^{|\mathcal{O}_b\rangle, \mathsf{Repro}}$.
4. *Return* b'.

where the $\mathsf{Repro}(x_2)$ oracle is reprogramming x_2 with a random prefix x_1 to a random number y:

1. $(x_1, y) \leftarrow_\$ X_1 \times Y$.
2. $\mathcal{O}_1 := \mathcal{O}_1^{(x_1||x_2)\to y}$.

3. Return x_1.

Although the Lemma 4 obtains a better bound in reprogramming, it is impossible to simulate the adversary in two stages. It will be unavoidable to get a q^2 loss when calling for a more general case. Hence, we use the following Lemma from [9] to complete the reduction from CCB to ν-SelfTargetMSIS.

Lemma 5 (Theorem 2 in [9]). *Let X and Y be two finite sets, and adversary $\mathcal{A}$ be an arbitrary oracle quantum algorithm making at most Q queries to the random oracle $\mathsf{H} : X \to Y$. The adversary outputs some $x \in X$ and a (possibly quantum) output z. The two-stage algorithm $\mathcal{S}^{\mathcal{A}}$ outputs $x \in X$ in the first stage, takes $\Theta \in Y$ as input to the second stage, and subsequently outputs a (possibly quantum) z, so that for any $x_0 \in X$ and any (possibly quantum) predicate V:*

$$\begin{aligned}\mathsf{Pr}[x = x_0 \wedge \mathsf{V}(x, \Theta, z) : (x, z) \leftarrow \langle \mathcal{S}^{\mathcal{A}}, \Theta \rangle] \\ \geq \frac{1}{(2Q+1)^2} \mathsf{Pr}[x = x_0 \wedge \mathsf{V}(x, \mathsf{H}(x), z) : (x, z) \leftarrow \mathcal{A}^{\mathsf{H}}].\end{aligned}$$

Furthermore, $\mathcal{S}$ runs in time polynomial in Q, $\log |X|$ and $\log |Y|$.

3 SelfTargetMSIS variant on ν hash target

The reduction from a single target SelfTargetMSIS to MLWE was given in [13]. The first step is to define a "plain" version of SelfTargetMSIS, known as Plain-SelfTargetMSIS, where the input matrix is not given in Hermite Normal Form. Next, it defines two experiments: the chosen coordinate binding experiment CCB and the collapsing experiment Collapse. The chain of reduction works as:

$$\begin{aligned}\mathsf{SelfTargetMSIS}_{\mathsf{H},\tau,m,l,\gamma} &\leftarrow \mathsf{Plain\text{-}SelfTargetMSIS}_{\mathsf{H},\tau,m,m+l,\gamma} \\ &\leftarrow \mathsf{CCB}_{\tau,m,m+l,\gamma} \leftarrow \mathsf{Collapse}_{m,m+l,\gamma} \leftarrow \mathsf{MLWE}_{m+l,m,\eta},\end{aligned}$$

of which the first transition requires $q \equiv 1 \pmod{2n}$ and the final transition requires $q > 16$ and $2\gamma\eta n(m+l) < \lfloor q/32 \rfloor$.

The above result could not be directly used in a multisignature scheme, because the original SelfTargetMSIS and Plain-SelfTargetMSIS only target one hash result, but multisignature further requires each party to answer an individual challenge c_i independently. More importantly, the adversary is able to control what the public keys of the malicious parties are. So, we propose a variant of the original SelfTargetMSIS to a ν-targets ν-SelfTargetMSIS.

To prove the hardness of ν-SelfTargetMSIS, we have to follow the route of [13], and consequently propose ν-Plain-SelfTargetMSIS. We can reduce the CCB experiment to our ν-Plain-SelfTargetMSIS. Hence, we can reuse the transition from CCB, Collapse to MLWE in [13].

Definition 7 (ν-SelfTargetMSIS). *For ν as a positive integer, $m, k, \gamma \in \mathbb{N}$ and $\mathsf{H} : \{0,1\}^* \to B_\tau$, the advantage that any adversary breaking the ν-SelfTargetMSIS is defined as*

$$\mathbf{Adv}^{\nu\text{-SelfTargetMSIS}}_{\mathsf{H},\tau,m,k,\gamma}(\mathcal{A}) := \Pr\left[\begin{array}{l|l} \|\mathbf{y}\|_\infty \le \gamma \wedge \|\mathbf{y}'\|_\infty \le \gamma & \mathbf{A} \leftarrow \mathcal{R}_q^{m\times k}; \\ \wedge\, \mathsf{H}(i, \mathbf{A}', [\mathbf{I}_m|\mathbf{A}]\mathbf{y} + \mathbf{A}'\mathbf{y}', M) = c_i & (\mathbf{y}, \mathbf{A}', \mathbf{y}', M) \leftarrow \mathcal{A}^{|\mathsf{H}\rangle}(\mathbf{A}) \\ \wedge\, \mathbf{y}_{m+k} = c_1 \wedge \mathbf{y}'_i = c_{i+1} & \end{array}\right],$$

in which $i \in [\nu - 1]$, $\mathbf{y} \in \mathcal{R}_q^{m+k}$, $\mathbf{A}' \in \mathcal{R}_q^{m\times(\tau-1)}$ and $\mathbf{y}' \in \mathcal{R}_q^{\nu-1}$.

Definition 8 (ν-Plain-SelfTargetMSIS). *For ν as a positive integer, $m, k, \gamma \in \mathbb{N}$ and $\mathsf{H} : \{0,1\}^* \to B_\tau$, the advantage that any adversary breaking the ν-Plain-SelfTargetMSIS is defined as*

$$\mathbf{Adv}^{\nu\text{-Plain-SelfTargetMSIS}}_{\mathsf{H},\tau,m,k,\gamma}(\mathcal{A}) := \Pr\left[\begin{array}{l|l} \|\mathbf{y}\|_\infty \le \gamma \wedge \|\mathbf{y}'\|_\infty \le \gamma & \mathbf{A} \leftarrow \mathcal{R}_q^{m\times k}; \\ \wedge\, \mathsf{H}(i, \mathbf{A}', \mathbf{A}\mathbf{y} + \mathbf{A}'\mathbf{y}', M) = c_i & (\mathbf{y}, \mathbf{A}', \mathbf{y}', M) \leftarrow \mathcal{A}^{|\mathsf{H}\rangle}(\mathbf{A}) \\ \wedge\, \mathbf{y}_k = c_1 \wedge \mathbf{y}'_i = c_{i+1} & \end{array}\right],$$

in which $i \in [\nu - 1]$, $\mathbf{y} \in \mathcal{R}_q^{k}$, $\mathbf{A}' \in \mathcal{R}_q^{m\times(\tau-1)}$, and $\mathbf{y}' \in \mathcal{R}_q^{\nu-1}$.

3.1 Reducing ν-Plain-SelfTargetMSIS to ν-SelfTargetMSIS

Proposition 1. *Suppose $q = 1 \pmod{2n}$ and $m, k, \gamma, \tau \in \mathbb{N}$, $\mathsf{H}_i : \{0,1\}^* \to B_\tau$. If there is an adversary $\mathcal{A}$ making Q queries to solve ν-SelfTargetMSIS$_{\mathsf{H},\tau,m,k,\gamma}$, there is an adversary $\mathcal{B}$ solving ν-Plain-SelfTargetMSIS$_{\mathsf{H},\tau,m,m+k,\gamma}$ with Q queries with loss at least n/q^k.*

Following up with Lemma 2 and Proposition 1 in [13], the proof is trivial to get by $\mathcal{R}_q \cong \mathbb{Z}_q^n$. And we remark that the randomly sampled part of our scheme is $\mathbf{A}$, while $\mathbf{A}'$ only eliminates the probability of having no row-echelon form, other than enlarging it. So the probability bound of distinguishing two cases is less than n/q^k.

3.2 Reducing CCB to ν-Plain-SelfTargetMSIS

We first review the experiment CCB. Let $\tau, m, l, \gamma \in \mathbb{N}$. The advantage of a quantum algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for winning $\mathsf{CCB}_{\tau,m,l,\gamma}$ is defined as the probability that the experiment $\mathsf{CCB}_{\tau,m,l,\gamma}$ in Algorithm 7 outputs 1.

The existing reduction from $\mathsf{CCB}_{\tau,m,l,\gamma}$ to Plain-SelfTargetMSIS$_{\mathsf{H},\tau,m,l,\gamma}$ in [13] relies on measure-and-reprogram lemma (Lemma 5) to build up a two-stage simulator of an adversary. The lemma seems to be unsuitable to adopt in our

$\mathbf{A} \xleftarrow{\$} \mathcal{R}_q^{m\times l}$.
$(\mathbf{z}, \mathsf{T}) \leftarrow \mathcal{A}_1(\mathbf{A})$, where $\mathbf{z} \in \mathcal{R}_q^m$ and T is auxiliary register.
$c_i \xleftarrow{\$} B_\tau$
$y \leftarrow \mathcal{A}_2(\mathsf{T}, c)$ where $y \in \mathcal{R}_q^l$.
if $\mathbf{Ay} = \mathbf{z}, \|\mathbf{y}\|_\infty \leq \gamma$, *and,* $y_l = c$ **then**
  return 1
end
return 0

Algorithm 7: Experiment $\mathsf{CCB}_{\tau,m,l,\gamma}$.

reduction because it is unclear whether involving more than one $(x, \mathsf{H}(x))$ in verification would impact the lemma.

On the other hand, [9] provides a variant supporting multiple inputs $\mathbf{x}$ and reprogramming the corresponding outputs $\mathbf{\Theta}$. However, the multiplicative loss on such reduction is too high, i.e. $1/(2Q+1)^{2n}$. So finding a way with minimal loss is preferable. To begin with it, we first dive into the core of Lemma 5.

Revisiting Lemma 5. The supporting technique of this lemma is Lemma 1 in [9]. The adversary $\mathcal{A}$ in the model is described as $A_1, A_2, \cdots, A_Q$ with the initial state $|\phi_0\rangle$. Subsequently, the state after Q queries turns out to be $|\phi_Q^{\mathsf{H}}\rangle := \mathcal{A}^{\mathsf{H}}|\phi_0\rangle := A_Q\mathcal{O}^{\mathsf{H}}\cdots A_1\mathcal{O}^{\mathsf{H}}|\phi_0\rangle$, of which $\mathcal{O}^{\mathsf{H}} : |c\rangle|x\rangle|y\rangle \mapsto |c\rangle|x\rangle|y \oplus c \cdot \mathsf{H}(x)\rangle$. The model gives controlled queries on $c \in \{0,1\}$. Moreover, it uses the notation of reprogramming as

$$\mathsf{H} * \Theta x_0 : x \mapsto \begin{cases} \Theta & \text{if } x = x_0, \\ \mathsf{H}(x) & \text{Otherwise.} \end{cases}$$

where the H is the hash function modeled as a quantum random oracle. The abbreviation is $(\mathcal{A}_{i\to Q}^{\mathsf{H}*\Theta x})(\mathcal{A}_{0\to i}^{\mathsf{H}})|\phi_0\rangle = (\mathcal{A}_{i\to Q}^{\mathsf{H}*\Theta x})|\phi_i^{\mathsf{H}}\rangle$, meaning that the adversary is started with $|\phi_0\rangle$ and the oracle answers the first i queries with H and reprograms on x to Θ for the remaining queries. Consequently, the underlying lemma could be expressed as Lemma 6.

Lemma 6 (Lemma 1 in [9]). *Let $\mathcal{A}$ be a Q-query quantum algorithm. For any $\mathsf{H} : X \to Y$, any $x \in X$, $\Theta \in Y$, and any projection $\Pi_{x,\Theta}$, it holds*

$$\mathbb{E}_{i,b}\left[\|(|x\rangle\langle x| \otimes \Pi_{x,\Theta})(\mathcal{A}_{i+b\to Q}^{\mathsf{H}*\Theta x})(\mathcal{A}_{i\to i+b}^{\mathsf{H}})X|\phi_i^{\mathsf{H}}\rangle\|_2^2\right] \geq \frac{\left[\|(|x\rangle\langle x| \otimes \Pi_{x,\Theta})|\phi_Q^{\mathsf{H}*\Theta x}\rangle\|_2^2\right]}{(2Q+1)^2},$$

where $(i,b) \in \{0,\cdots,Q-1\} \times \{0,1\} \cup \{Q,0\}$.

Here, $|x\rangle\langle x|$ measures the final output register X to obtain x, and the projection $\Pi_{x,\Theta}$ is the measurement of some other register Z relying on $\mathsf{H}(x)$. Besides, the measurement of the hash query register is written as X in this formula.

In other words, the left-hand side is the expectation of measuring the query register to get x and then deciding to reprogram $\mathsf{H}(x)$ with Θ depending on a random choice b, and finally the output register X is measured and coincides with x.

On the right-hand side of Lemma 6, the measurement probability that could be treated as the probability of outputting a valid z passing the verification V, as long as Θ is random.

$$\mathsf{Pr}[x = x_0 \wedge \mathsf{V}(x, \mathsf{H}(x), z) : (x, z) \leftarrow \mathcal{A}^{\mathsf{H}}] = \||(x_0\rangle\langle x_0| \otimes \Pi_{x_0,\mathsf{H}(x_0)})|\phi_Q^{\mathsf{H}*\Theta\mathsf{x}}\rangle\|_2^2.$$

Now, we will adopt the Lemma 6 to show the reduction from CCB to ν-Plain-$\mathsf{SelfTargetMSIS}$.

Proposition 2. *Let $\nu, m, l, \gamma, \tau \in \mathbb{N}$. Suppose there is an adversary $\mathcal{A}$ solving ν-Plain-$\mathsf{SelfTargetMSIS}_{\mathsf{H},\tau,m,l,\gamma}$ with at most Q queries with advantage ϵ, there exists a two-stage quantum algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ solving and winning the experiment $\mathsf{CCB}_{\tau,m,l,\gamma}$ with advantage at least $\epsilon/(2Q+1)^2$.*

Proof. To complete the reduction, we expect a 2-stage simulator $\mathcal{S}_1^{\mathcal{A}}$ and $\mathcal{S}_2^{\mathcal{A}}$, such that $(\mathbf{A}', z := \mathbf{A}\mathbf{y} + \mathbf{A}'\mathbf{y}', M, T) \leftarrow \mathcal{S}_1^{\mathcal{A}}$ and $(\mathbf{y}, \mathbf{y}') \leftarrow \mathcal{S}_2^{\mathcal{A}}(T, c_1)$ such that $\|\mathbf{y}\|_\infty \leq \gamma \wedge \|\mathbf{y}'\|_\infty \leq \gamma \wedge \mathbf{y}_k = c_1 \wedge \mathbf{y}'_i = c_{i+1}$.

Here, we only need to program $\mathsf{H}(x_1)$ to $\Theta := c_1$ and remain $\{\mathsf{H}(x_i)\}_{i>1}$ the same, which means neither H nor $\mathsf{H} * \Theta x$ will change the result of $\{\mathsf{H}(x_i)\}_{i>1}$. All we need to care about is whether $(1, \mathbf{A}', \mathbf{A}\mathbf{y} + \mathbf{A}'\mathbf{y}', M)$ has been queried or not. Moreover, when we measure the output register X for $x_1 := (1, \mathbf{A}', \mathbf{A}\mathbf{y} + \mathbf{A}'\mathbf{y}', M)$, we immediately determine $x_i := (i, \mathbf{A}', \mathbf{A}\mathbf{y} + \mathbf{A}'\mathbf{y}', M)$ for $i > 2$.

So we can use almost the same strategy as the CCB to Plain-$\mathsf{SelfTargetMSIS}$ reduction, that we pick a random $(i, b) \in \{0, \cdots, Q-1\} \times \{0, 1\} \cup \{Q, 0\}$ and measure the query register after the i-th query, and we then choose to program the hash on or after the current query, of the measured result. And the expectation of the outcome probability that the output register X measures the same as the query register, is indeed $\mathbb{E}_{i,b}\left[\|(|x\rangle\langle x| \otimes \Pi_{x,\Theta})(\mathcal{A}_{i+b\to Q}^{\mathsf{H}*\Theta x})(\mathcal{A}_{i\to i+b}^{\mathsf{H}})X|\phi_i^{\mathsf{H}}\rangle\|_2^2\right]$.

Consequently, there is a pair of simulators $(\mathcal{S}_1^{\mathcal{A}}, \mathcal{S}_2^{\mathcal{A}})$, of which $\mathcal{S}_2^{\mathcal{A}}$ receives a $\Theta := c_1$ as an input, and then outputs a valid solution to ν-Plain-$\mathsf{SelfTargetMSIS}$. The expectation of the probability is $\frac{1}{(2Q+1)^2}$ times the probability of outputting a single transcript, as in Lemma 6.

Moreover, the algorithms $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ could be constructed by $(\mathcal{S}_1^{\mathcal{A}}, \mathcal{S}_2^{\mathcal{A}})$ with the following modification.

- $\mathcal{B}_1$: calls $\mathcal{S}_1^{\mathcal{A}}$ to get $(\mathbf{A}', z, M, T)$ and outputs $T' := T$ and $z' := z - \mathbf{A}'\mathbf{y}'$. Here $\mathbf{y}'_i = c_{i+1} = \mathsf{H}(i, \mathbf{A}', z, M)$ are all fixed.
- $\mathcal{B}_2$: on receiving c_1, calls $\mathcal{S}_2^{\mathcal{A}}(T', c)$ to get $(\mathbf{y}, \mathbf{y}')$ which fulfills $z = \mathbf{A}\mathbf{y} + \mathbf{A}'\mathbf{y}' \wedge \|\mathbf{y}\|_\infty \leq \gamma \wedge \|\mathbf{y}'\|_\infty \leq \gamma \wedge \mathbf{y}_k = c_1 \wedge \mathbf{y}'_i = c_{i+1}$. Thus, it is able to output $\mathbf{y}$ such that $\mathbf{A}\mathbf{y} = z'$.

□

Parameter	Explanation	Limitation
ν	The number of signers	
n	Ring dimension	
q	Ring Modulus	$q \equiv 1 \pmod{2n}$, $q > 16$
(k, l)	Dimensions of $\mathbf{A}$	$l \leq k$
d	Dropped bits from $\mathbf{t}$	$\tau \cdot 2^{d-1} \leq \gamma_2$
γ_1	Max signature coefficient (approx.)	$\beta \ll \gamma_1$
γ_2	The limit of making hints	$q > 4\nu\gamma_2$, $q \equiv 1 \pmod{2\nu\gamma_2}$, $\beta \ll \gamma_2$
η	Max coefficients of $\mathbf{s}_1, \mathbf{s}_2$	
β	The bound of $\|cs\|_\infty$, $s \in S_\eta$	$\beta = \tau\eta$
τ	The number of 1 in challenge space	

Table 1. Parameters for QROM-Dilithium-Musig

3.3 Security of ν-SelfTargetMSIS

By putting Proposition 1 in our paper, with Proposition 3 in [13] (reduction from the experiment Collapse to CCB) and Proposition 4 in [13] (reduction from the experiment CCB to MLWE) together, we get the following theorem on the security of ν-SelfTargetMSIS.

Theorem 1. *Let $m, k, \tau, \gamma, \eta \in \mathbb{N}$. Suppose $q > 16$, $q \equiv 1 \pmod{2n}$, and $2\gamma\eta n(m+k) < \lfloor q/32 \rfloor$. If there is an adversary making Q quantum queries to solve the problem ν-SelfTargetMSIS$_{\mathsf{H},\tau,m,k,\gamma}$ with advantage ϵ and $\mathsf{H} : \{0,1\}^* \to B_\tau$, there exists a quantum algorithm that solves MLWE$_{m+k,m,\eta}$ with advantage*

$$\frac{\epsilon - nq^{-k}}{4(2Q+1)^2}\left(\frac{\epsilon - nq^{-k}}{4(2Q+1)^2} - \frac{1}{|B_\tau|}\right) - \frac{1}{4 \cdot 3^w},$$

for all $w \in \mathbb{N}$.

4 QROM-Dilithium-MuSig

In this section, we propose the QROM-Dilithium-MuSig, which is highly efficient to run NTT because of $q \equiv 1 \pmod{2n}$. All the parameter requirements follow Table 1. Moreover, we consider the challenge space as $\mathsf{ChSet} = B_\tau$, and thus the $|\mathsf{ChSet}| = 2^\tau \binom{n}{\tau}$.

4.1 Construction

Setup. On input a security parameter 1^λ, it outputs the public parameters as shown in Table 1. This part also requires the generation of the public matrix $\mathbf{A}$. A safe generation requires all participants to efficiently check the trustworthiness of $\mathbf{A}$. Here we refer to [1] for generating a common reference string without trusted setup. It also defines two hash functions $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^*$ and $\mathsf{H}_2 : \{0,1\}^* \to B_\tau$.

Key generation. On input the public parameter, it samples secret keys by the MLWE parameters $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$. It then constructs the public key and compresses it using $\mathsf{Power2Round}_q$ method, i.e. $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$ and sets $(\mathbf{t}_H, \mathbf{t}_L) := \mathsf{Power2Round}_q(\mathbf{t}, d)$. It will finally output $\mathsf{pk} := (\mathbf{t}_H, \underline{\mathbf{t}_L}), \mathsf{sk} := (\mathbf{s}_1, \mathbf{s}_2)$. Here, $\underline{\mathbf{t}_L}$ is the part used in signing only because of the key compression technique.

Signing. Assuming that the multi-signature is targeting the message μ, the size of signers is ν, the member index is ranged from 1 to ν, and the current party is indexed by v. Each party takes part in the signing procedure with the secret key of the current party sk_v and the collection of the public keys $\mathbf{PK}$.

Sign_1. The signer generates its own random masks by sampling $\mathbf{y}_v \leftarrow S_{\gamma_1 - 1}^\ell$ first, computes $\mathbf{w}_v = \mathbf{A}\mathbf{y}_v$, and makes a commitment $\mathbf{g}_v = \mathsf{H}_1(\mathsf{pk}_v, \mathbf{w}_v)$. It sends out $(\mathsf{state}_1 := (\mathbf{PK}, \mathbf{y}_v, \mathbf{w}_v), \mathbf{g}_v)$.

Sign_2. On receiving the previous state state_1 and all the commitments denoted as $\{\mathbf{g}_u\}_{u \in [\nu], u \neq v}$ from other parties, it outputs a reveal of the commitment as $\mathbf{w}_v$ together with the current state, i.e. $(\mathsf{state}_2 := (\mathbf{PK}, \mathbf{y}_v, \{\mathbf{g}_u\}_{u \in [\nu]}), \mathbf{w}_v)$.

Sign_3. It takes the previous state state_2 and all the revealing $\{\mathbf{w}_u\}_{u \in [\nu]}$ as inputs. And it performs outputs $(\mathsf{state}_3 := (\mathbf{PK}, \mathbf{w}_H), \mathbf{z}_v)$ with the following.

1. Abort if $\mathbf{g}_u \neq \mathsf{H}_1(\mathsf{pk}_u, \mathbf{w}_u)$ for some $u \in [\nu]$.
2. Compute $\mathbf{w} = \sum_{u=1}^{\nu} \mathbf{w}_u$, $\mathbf{w}_H = \mathsf{HighBits}_q(\mathbf{w}, 2\nu\gamma_2)$.
3. Get a challenge $c_v = \mathsf{H}_2(v, \mathbf{w}_H, \mu, \mathbf{PK})$.
4. Compute $\mathbf{z}_v = \mathbf{y}_v + c_v \mathbf{s}_{v,1}$.
5. Abort when $\|\mathbf{z}_v\|_\infty \geq \gamma_1 - \beta$.

Sign_4. On receiving the previous state state_3 and a series of $\{\mathbf{z}_u\}_{u \in [\nu]}$ from each individual signer, it sums up all of the $\mathbf{z}_u$ together by $\mathbf{z} = \sum_{u=1}^{\nu} \mathbf{z}_u$. It restarts the whole signing when $\|\mathsf{LowBits}_q(\mathbf{w}_v - c_v \mathbf{s}_{v,2}, 2\nu\gamma_2)\|_\infty \geq \nu(\gamma_2 - \beta)$. It then computes the hint of the signature by $\mathbf{h} = \mathsf{MakeHint}_q(-\sum_{u=1}^{\nu} c_u \mathbf{t}_{u,L}, \mathbf{A}\mathbf{z} - 2^d \sum_{u=1}^{\nu} c_u \mathbf{t}_{u,H}, 2\nu\gamma_2)$. It outputs the signature $\sigma = (\mathbf{w}_H, \mathbf{z}, \mathbf{h})$.

Verification. The verification takes the public keys of all the parties $\mathbf{PK} = \{\mathsf{pk}_u\}_{u \in [u]}$, and signatures in the form $\sigma = (\mathbf{w}_H, \mathbf{z}, \mathbf{h})$. It reconstructs $\mathbf{x} = \mathbf{A}\mathbf{z} - 2^d \sum_{u=1}^{\nu} c_u \mathbf{t}_{u,H}$ where $c_u = \mathsf{H}_2(u, \mathbf{w}_H, \mu, \mathbf{PK})$. And it later outputs 1 if the following two conditions are fulfilled, and it outputs 0 otherwise.

- $\|\mathbf{z}\|_\infty < \nu(\gamma_1 - \beta)$,
- $\mathbf{w}_H = \mathsf{UseHint}_q(\mathbf{h}, \mathbf{x}, 2\nu\gamma_2)$.

4.2 Correctness

In verification, it is easy to see $\|\mathbf{z}\| \leq \nu(\gamma_1 - \beta)$. And it remains to check $\mathbf{w}_H = \mathsf{UseHint}_q(\mathbf{h}, \mathbf{x}, 2\nu\gamma_2)$.

$$\begin{aligned}\mathsf{UseHint}_q(\mathbf{h}, \mathbf{x}, 2\nu\gamma_2) =& \mathsf{UseHint}_q(\mathsf{MakeHint}_q(-\sum_{u=1}^{\nu} c_u \mathbf{t}_{u,L}, \mathbf{x}, 2\nu\gamma_2), \mathbf{x}, 2\nu\gamma_2) \\ =& \mathsf{HighBits}_q(-\sum_{u=1}^{\nu} c_u \mathbf{t}_{u,L} + \mathbf{x}, 2\nu\gamma_2),\end{aligned}$$

where

$$\mathbf{x} = \mathbf{A}\mathbf{z} - 2^d \sum_{u=1}^{\nu} c_u \mathbf{t}_{u,H} = \mathbf{w} + \sum_{u=1}^{\nu} c_u(\mathbf{A}\mathbf{s}_{u,1} - 2^d \mathbf{t}_{u,H}).$$

And by $\|\sum_{u=1}^{\nu} c_u \mathbf{t}_{u,L}\|_\infty \leq \nu\tau \cdot 2^{d-1} \leq \nu\gamma_2$ and applying Lemma 2,

$$\begin{aligned}
\mathsf{UseHint}_q(\mathbf{h}, \mathbf{x}, 2\nu\gamma_2) =& \mathsf{HighBits}_q(\mathbf{w} + \sum_{u=1}^{\nu} c_u(\mathbf{A}\mathbf{s}_{u,1} - 2^d \mathbf{t}_{u,H} - c_u \mathbf{t}_{u,L}), 2\nu\gamma_2) \\
=& \mathsf{HighBits}_q(\mathbf{w} - \sum_{u=1}^{\nu} c_u \mathbf{s}_{u,2}, 2\nu\gamma_2) \\
=& \mathsf{HighBits}_q(\mathbf{A}\mathbf{z} - \sum_{u=1}^{\nu} c_u \mathbf{t}_u, 2\nu\gamma_2).
\end{aligned}$$

By the signing procedure $\|\mathsf{LowBits}_q(\mathbf{A}\mathbf{z} - \sum_{u=1}^{\nu} c_u \mathbf{t}_u, 2\nu\gamma_2)\|_\infty < \nu(\gamma_2 - \beta)$ and $\|\sum_{u=1}^{\nu} c_u \mathbf{s}_{u,2}\|_\infty \leq \nu\beta$, we get

$$\begin{aligned}
\mathsf{UseHint}_q(\mathbf{h}, \mathbf{x}, 2\nu\gamma_2) &= \mathsf{HighBits}_q(\mathbf{w} - \sum_{u=1}^{\nu} c_u \mathbf{s}_{u,2}, 2\nu\gamma_2) \\
&= \mathsf{HighBits}_q(\mathbf{w}, 2\nu\gamma_2) \\
&= \mathbf{w}_H.
\end{aligned}$$

4.3 Security

In the security reduction, we assume that the forger has the quantum query to hash functions H_1 and H_2, while it can only access the signing oracles classically just like the real-world classical communication channel. The main proof follows [11]. However, we remark that the original proof is relying on the lossy-key generation where $q \equiv 5 \pmod 8$, but our scheme relies on $q \equiv 1 \pmod{2n}$.

Without loss of generality, we assume the adversary is going to interactively query with the first party that acted by the simulator, which would involve the pk_1 without knowing sk_1. And all other parties are controlled by the adversary.

Lemma 7. *If the multisignature scheme QROM-Dilithium-Musig is secure in MS-UF-NMA model, the QROM-Dilithium-Musig is also secure in MS-UF-CMA with the advantage loss* $\frac{3Q_sE}{2}\sqrt{\frac{Q_H+Q_s\nu E}{2^\tau\binom{n}{\tau}}} + Q_sE\epsilon^{\mathsf{rMLWE}}_{k,l,\gamma,\beta,\eta,\tau}$*, of which* Q_s *is the number of queries on signing,* Q_H *is the number of queries on hash,* E *is the expectation of the repeating time in signing and* $\epsilon^{\mathsf{rMLWE}}_{k,l,\gamma_1,\beta,\eta,\tau}$ *is the probability rMLWE that could be distinguished.*

Proof. The major technique of achieving unforgeability is on game-hopping. Here is the definition of the original game.

Setup. On input 1^λ, the simulator sets up $\mathbf{A}$, and the key pair for the first party, i.e. $(\mathsf{pk}^*, \mathsf{sk}^*) := (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(\mathbf{A})$. All public parameters and pk^* are given to the adversary $\mathcal{A}$.

Oracles. Hash functions $\mathsf{H}_1, \mathsf{H}_2$ are simulated by quantum random oracles. The adversary is able to make Q_1 queries to H_1 and Q_H queries to H_2.

SigningOracle. Suppose the i-th singing query is denoted by $(\mathbf{PK}^{(i)}, \mu^{(i)})$. If pk^* is not contained by $\mathbf{PK}^{(i)}$, just answer the query normally. Otherwise, the index of the target keys pk^* is set to the first one without loss of generality (i.e., $v = 1$). Moreover, we assume that each state of the signing procedure is continuous and each state_i would be unique as if the adversary is working with a normal user.

- Sign_1. The simulator invokes the original Sign_1 and sends $\mathbf{g}_1^{(i)}$ to $\mathcal{A}$ as Party 1.
- Sign_2. The simulator receives $\{\mathbf{g}_u^{(i)}\}_{u\in[2,\nu]}$ from $\mathcal{A}$ and works as the party 1 to invokes the original Sign_2 and sends $\mathbf{w}_1^{(i)}$ to $\mathcal{A}$.
- Sign_3. The simulator receives $\{\mathbf{w}_u^{(i)}\}_{u\in[2,\nu]}$ from $\mathcal{A}$, and invokes the original Sign_3 and sends $\mathbf{z}_1^{(i)}$ to $\mathcal{A}$.
- Sign_4. The simulator receives $\{\mathbf{z}_u^{(i)}\}_{u\in[2,\nu]}$ from $\mathcal{A}$, and works as the party 1 to invoke the original Sign_4 and sends $(\mathbf{w}_H^{(i)}, \mathbf{z}^{(i)}, \mathbf{h}^{(i)})$ to $\mathcal{A}$.

Challenge. The forger generates a valid signature of any message on the tuple $(\mathbf{PK}^*, (\mathbf{w}_H^*, \mathbf{z}^*, \mathbf{h}^*), \mu^*)$ such that $\mathsf{Verify}(\mathbf{PK}^*, (\mathbf{w}_H^*, \mathbf{z}^*, \mathbf{h}^*), \mu^*) = 1$ and $\mathsf{pk}^* \in \mathbf{PK}^*$.

The target of the game hopping is to replace the keys of the first party by the underlying MS-UF-NMA key and make the simulator able to answer any query even without the secret key.

Game_0: The original game.

Game_1: It is the same as Game_0, except:

- Oracles. The hash function H_1 is replaced by a $2(Q_1 + Q_s\nu E)$ degree polynomial function $\mathcal{H}_1$, of which Q_1 is the number of queries that H_1 is asked directly and Q_s is the number of queries that signing oracle is invoked.

By Lemma 1, we have:

$$|\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_0} = 1] - \mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_1} = 1]| \leq \mathsf{negl}(\lambda).$$

Game_2: It is the same as Game_1, except:

- Sign_2. The simulator receives $\{\mathbf{g}_u^{(i)}\}_{u\in[2,\nu]}$ from $\mathcal{A}$. To generate $\mathbf{w}_1^{(i)}$, the simulator extracts the original $\mathbf{w}_u^{(i)}$ for $u \in [2, \nu]$ by the computing the roots of $\mathcal{H}_1 - \mathbf{w}_u^{(i)}$ as $\mathbf{r}_u^{(i)}$. The probability that the number of roots greater than 2 is given by $\epsilon_{\mathsf{roots}}$. If there is more than one result, it just simply picks a random one. And it sends $\mathbf{w}_1^{(i)}$ to $\mathcal{A}$.
- Sign_3. The simulator receives $\{\mathbf{w}_u^{(i)}\}_{u\in[2,\nu]}$ from $\mathcal{A}$, and checks the extraction is indeed the original $\mathbf{w}_u^{(i)}$, by $\hat{\mathbf{w}}_u^{(i)} = \mathbf{w}_u$. It aborts if the extraction is not correct. Subsequently, it invokes the original Sign_3 and sends $\mathbf{z}_1^{(i)}$ to $\mathcal{A}$.

We append the public key into the hash input such that it would avoid multiple roots with high probability. The extraction loss of taking the polynomial roots is very small by using polynomial simulation and Lemma 1. Hence we have:

$$|\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_1}=1]-\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_2}=1]|\leq \mathsf{negl}(\lambda).$$

$\mathsf{Game}_{3,i}$ for $i=0,...,Q_S$: Before any hopping under the sub-game, the original sub-game $\mathsf{Game}_{3,0}$ is indeed the same as the previous game Game_2.

Following up with the previous state, $\mathsf{Game}_{3,i}$ transits from $\mathsf{Game}_{3,i-1}$. In this hopping, the simulator will pick a random challenge in the very beginning, by which the simulator is able to generate $\mathbf{w}$ it could be answered. And using Lemma 4, the hopping would succeed with non-negligible probability.

Here are the changes from $\mathsf{Game}_{3,i-1}$ to $\mathsf{Game}_{3,i}$. The i-th signing call is simulated by the following.

- Sign_1. The simulator picks a random challenge $c_1^{(i)}$ for the first party and $\mathbf{z}_1^{(i)}\leftarrow\mathcal{S}_{\gamma_1-\beta-1}$. Subsequently, it sets $\mathbf{w}_1^{(i)}=\mathbf{A}\mathbf{z}_1^{(i)}-c_1^{(i)}(\mathbf{t}^*-\mathbf{s}_2^*)$ with probability $1-e^{-nl\beta/\gamma_1}$ and $\mathbf{w}_1^{(i)}\leftarrow\mathcal{R}_q^k$. And it computes and sends $\mathbf{g}_1^{(i)}$ as the original algorithm.
- Sign_2. As $\{\mathbf{w}_u^{(i)}\}_{u\in[2,\nu]}$ are successfully extracted, the simulator construct $\mathbf{w}_H^{(i)}$. Then the simulator will reprogram $\mathsf{H}_2(1,\mathbf{w}_H^{(i)},\mu,\mathbf{PK}^{(i)})$ as $c_1^{(i)}$.
- Sign_3. If $\mathbf{w}_1^{(i)}$ is sampled uniformly, the simulator restarts the algorithm. Alternatively, it will proceed the original Sign_3.

Summing up all sub-games together, we get Q_sE points to reprogram and the total queried number is $Q_H+Q_s\nu E$. The random challenge space is B_τ of size $2^\tau\binom{n}{\tau}$. With Lemma 4, the differences turn to

$$\begin{aligned}\left|\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_{3,Q_s}}=1]-\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_2}=1]\right|=&\sum_{i=1}^{Q_s}\left|\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_{3,i}}=1]-\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_{3,i-1}}=1]\right|\\ \leq&\frac{3Q_sE}{2}\sqrt{\frac{Q_H+Q_s\nu E}{2^\tau\binom{n}{\tau}}}+Q_sE\epsilon^{\mathsf{rMLWE}}_{k,l,\gamma,\beta,\eta,\tau}.\end{aligned}$$

Game_4: It is the same as Game_{3,Q_s}, except that $\mathbf{s}_2^*$ will be replaced by an arbitrary $s^{(i)}\xleftarrow{\$}\mathcal{S}_\eta^k$ in Sign_1, such that:

- Sign_1. The simulator picks $\mathbf{s}^{(i)}\xleftarrow{\$}\mathcal{S}_\eta^k$, and a random challenge $c_1^{(i)}$ for the first party and $\mathbf{z}_1^{(i)}\leftarrow\mathcal{S}_{\gamma_1-\beta-1}$. Subsequently, it sets $\mathbf{w}_1^{(i)}=\mathbf{A}\mathbf{z}_1^{(i)}-c_1^{(i)}(\mathbf{t}^*-\mathbf{s}^{(i)})$. It computes and sends $\mathbf{g}_1^{(i)}$ as the original algorithm.

In this hopping, we evaluate the difference from the verification. It would be $\mathbf{w}_H^{(i)}=\mathsf{UseHint}_q(\mathbf{h}^{(i)},\mathbf{x}^{(i)},2\nu\gamma_2)$, where

$$\mathbf{x}^{(i)}=\mathbf{A}\sum_{u=1}^{\nu}\mathbf{z}_u^{(i)}-2^d\sum_{u=1}^{\nu}c_u^{(i)}\mathbf{t}_{u,H},$$

and in verification with Lemma 2 and $\| \sum_{u=1}^{\nu} c_u^{(i)} \mathbf{t}_{u,L} \|_\infty \leq \nu\gamma_2$,

$$\begin{aligned}\mathbf{w}_H^{(i),\mathsf{VERIFY}} =& \mathsf{UseHint}_q(\mathsf{MakeHint}_q(-\sum_{u=1}^{\nu} c_u^{(i)} \mathbf{t}_{u,L}, \mathbf{x}^{(i)}, 2\nu\gamma_2), \mathbf{x}^{(i)}, 2\nu\gamma_2) \\ =& \mathsf{HighBits}_q(\mathbf{A}\sum_{u=1}^{\nu} \mathbf{z}_u^{(i)} - \sum_{u=1}^{\nu} c_u^{(i)} \mathbf{t}_u, 2\nu\gamma_2).\end{aligned}$$

It could be noticed that $\|c_1^{(i)}\mathbf{s}^{(i)} + \sum_{u=2}^{\nu} c_u^{(i)} \mathbf{s}_{u,2}\|_\infty \leq \nu\beta$ and $\|\mathsf{LowBits}_q(\mathbf{A}\sum_{u=1}^{\nu} \mathbf{z}_u^{(i)} - \sum_{u=1}^{\nu} c_u^{(i)} \mathbf{t}_u, 2\nu\gamma_2)\|_\infty < \nu(\gamma_2 - \beta)$, and as a direct result of Lemma 3:

$$\begin{aligned}\mathbf{w}_H^{(i),\mathsf{VERIFY}} =& \mathsf{HighBits}_q(\mathbf{A}\mathbf{z}_1^{(i)} - c_1^{(i)}\mathbf{t}^* + \mathbf{A}\sum_{u=2}^{\nu} \mathbf{z}_u^{(i)} - \sum_{u=2}^{\nu} c_u^{(i)} \mathbf{t}_u, 2\nu\gamma_2) \\ =& \mathsf{HighBits}_q(\mathbf{w}_1^{(i)} + \sum_{u=2}^{\nu} (\mathbf{A}\mathbf{z}_u^{(i)} - c_u^{(i)}(\mathbf{t}_u - \mathbf{s}_{u,2})), 2\nu\gamma_2).\end{aligned}$$

Thus, all the other parts are kept the same as the original game.

$$|\mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_3} = 1] - \mathsf{Pr}[\mathcal{A}^{\mathsf{Game}_4} = 1]| \leq \mathsf{negl}(\lambda).$$

Game_5: At the final game, we notice that the signing oracle queries could be perfectly simulated without using any private information of the first party. Now consider an algorithm $\mathcal{B}$ playing the MS-UF-NMA game and receiving $(\mathsf{pp}, \mathsf{pk}^*)$. Then $\mathcal{B}$ can run Game_5 with the MS-UF-CMA adversary $\mathcal{A}$ using $(\mathsf{pp}, \mathsf{pk}^*)$. If $\mathcal{A}$ wins by outputting $(\mathbf{PK}^*, \sigma^*, m^*)$, then $\mathcal{B}$ also wins by outputting $(\mathbf{PK}^*, \sigma^*, m^*)$. Hence we have:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{QROM\text{-}Dilithium\text{-}Musig}}(\mathcal{A}) \leq& \mathbf{Adv}^{\mathsf{MS\text{-}UF\text{-}NMA}}_{\mathsf{QROM\text{-}Dilithium\text{-}Musig}}(\mathcal{B}) \\ &+ \frac{3Q_sE}{2}\sqrt{\frac{Q_H + Q_s\nu E}{2^\tau \binom{n}{\tau}}} + Q_sE \cdot \epsilon^{\mathsf{rMLWE}}_{k,l,\gamma_1,\beta,\eta,\tau} + \mathsf{negl}(\lambda).\end{aligned}$$

□

Finally, we show that our multisignature scheme is MS-UF-NMA secure.

Lemma 8. *Let* $\zeta = \max\big(\nu(\gamma_1 - \beta), \nu(2\gamma_2 + 2^{d-1}\tau) + 1\big)$. *If there is any adversary who breaks MS-UF-NMA of* QROM-Dilithium-Musig*, there exists an adversary that breaks* $\mathsf{MLWE}_{k,\ell,\eta}$ *or* ν-$\mathsf{SelfTargetMSIS}_{\mathsf{H},\tau,k,\ell+1,\zeta}$.

Proof. From the point of verification, if there is any attack giving out $(\mathbf{PK}, \mathbf{w}_H, \mathbf{z}, \mathbf{h})$. The following equation must be correct for $\mathbf{x} = \mathbf{A}\mathbf{z} - 2^d \sum_{u=1}^{\nu} c_u \mathbf{t}_{u,H}$.

$$c_u = \mathsf{H}_2(u, \mathsf{UseHint}_q(\mathbf{h}, \mathbf{x}, 2\nu\gamma_2), \mu, \mathbf{PK}).$$

With the lemma 2, there is one $\mathbf{u}$ such that $\|\mathbf{u}\|_\infty \leq 2\nu\gamma_2 + 1$ and

$$\begin{aligned} c_u &= \mathsf{H}_2(u, \mathbf{x} + \mathbf{u}, \mu, \mathbf{PK}) \\ &= \mathsf{H}_2(u, \mathbf{Az} - 2^d \sum_{u=1}^{\nu} c_u \mathbf{t}_{u,H} + \mathbf{u}, \mu, \mathbf{PK}) \\ &= \mathsf{H}_2(u, \mathbf{Az} - \sum_{u=1}^{\nu} c_u(\mathbf{t}_u - \mathbf{t}_{u,L}) + \mathbf{u}, \mu, \mathbf{PK}) \\ &= \mathsf{H}_2(u, \mathbf{Az} - \sum_{u=1}^{\nu} c_u \mathbf{t}_u + (\sum_{u=1}^{\nu} c_u \mathbf{t}_{u,L} + \mathbf{u}), \mu, \mathbf{PK}). \end{aligned}$$

With $\|\mathbf{t}_{u,L}\|_\infty \leq 2^{d-1}$, there is another $\mathbf{u'} = \sum_{u=1}^{\nu} c_u \mathbf{t}_{u,L} + \mathbf{u}$ with the bound $\|\mathbf{u'}\|_\infty \leq (2\gamma_2 + 2^{d-1}\tau)\nu + 1$, where

$$c_u = \mathsf{H}_2(u, \mathbf{Az} - \sum_{u=1}^{\nu} c_u \mathbf{t}_u + \mathbf{u'}, \mu, \mathbf{PK}),$$

and it is equivalent to find out

$$\mathbf{y}_1 = \left[\mathbf{u'}\ \mathbf{z}\ \mathbf{c_1}\right]^\intercal, \mathbf{y}_2 = \left[\mathbf{c_2} \cdots\ \mathbf{c_U}\right]^\intercal,$$

and there is a

$$\mathbf{A}'_1 = \left[\mathbf{I}\ \mathbf{A}\ \mathbf{t_1}\right], \mathbf{A}'_2 = \left[\mathbf{t_2} \cdots\ \mathbf{t_U}\right],$$

such that

$$\mathsf{H}_2(u, \mathbf{A}'_1\mathbf{y}_1 + \mathbf{A}'_2\mathbf{y}_2, \mu, \mathbf{PK}) = c_u.$$

If the adversary could distinguish $\mathbf{A}'_1$ from the uniformly generated $[\mathbf{A}]$, it would break $\mathsf{MLWE}_{k,\ell,\eta}$. Otherwise, it turns out to be indeed the same form of τ-$\mathsf{SelfTargetMSIS}_{\mathsf{H},\tau,k,l+1,\zeta}$, except that $\mathbf{PK}$ contains more information than $\mathbf{A'}$ in τ-$\mathsf{SelfTargetMSIS}$. This change will not affect the security of such a problem, because $\mathbf{A}$ is fixed before the $\mathcal{A}$ could make an attack. Thus,

$$\mathbf{Adv}^{\mathsf{MS\text{-}UF\text{-}NMA}}_{\mathsf{QROM\text{-}Dilithium\text{-}Musig}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{MLWE}}_{k,\ell,\eta}(\mathcal{A}_1) + \mathbf{Adv}^{\nu\text{-}\mathsf{SelfTargetMSIS}}_{\mathsf{H},\tau,k,\ell+1,\zeta}(\mathcal{A}_2) + \mathsf{negl}(\lambda).$$

□

Comparison. We remark that our proof strategy has a better loss compared to the existing scheme [11]. Their scheme uses [21] to reprogram and further answers the signing queries, while our scheme uses Lemma 4. Because the simulation of [21] requires a $2Q$-wise independent function, and the loss is $(4+\sqrt{2})\sqrt{Q}2^{-\theta/4}$. Here θ is the input collision-entropy. And to further control the collision-entropy, they require an almost-compressed random oracle to simulate the hash input which works like $c_i := \mathsf{H}_1(\mathsf{H}_2(\cdot))$. By adding up with the almost compressed oracle failure probability, their loss is $\frac{2\sqrt{Q_1}}{2^{\theta/2}} + \frac{Q_s E(4+\sqrt{2})\sqrt{(Q_2+Q_s E-1)\nu}}{2^{\theta/4}}$ for the

reprogramming part, while our scheme gets $\frac{3Q_sE}{2}\sqrt{\frac{Q_H+Q_s\nu E}{2^\tau\binom{n}{\tau}}}$ instead. Here θ could be simply treated as the input length of H_2.

Although it seems to be controllable that the loss decreases when the θ increases, the length-preserving property of [21] prevents it from choosing the θ freely. Moreover, we would expect the challenge space to be B_τ when using Dilithium, which makes it hard to choose a proper θ in reality. Instead, using Lemma 4 does not limit the relationship between the input space and the output space and could guide the parameter choosing in a better way.

5 Conclusion

In this paper, we introduce the ν-SelfTargetMSIS problem and present a reduction from CCB to it. Consequently, we could further show that ν-SelfTargetMSIS is at least as hard as the MLWE problem. Besides, we propose a multisignature scheme based on the one proposed by [11]. Our multisignature scheme can reach a much tighter loss by using ν-SelfTargetMSIS and [12], and it is proven in $q \equiv 1 \pmod{2n}$ such that it benefits from NTT acceleration. Thus, our scheme is more practical and is expected to be widely used.

References

1. Abram, D., Waters, B., Zhandry, M.: Security-preserving distributed samplers: How to generate any CRS in one round without random oracles. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023. Lecture Notes in Computer Science, vol. 14081, pp. 489–514. Springer (2023), https://doi.org/10.1007/978-3-031-38557-5_16
2. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Lecture Notes in Computer Science, vol. 11693, pp. 269–295. Springer (2019), https://doi.org/10.1007/978-3-030-26951-7_10
3. Bai, S., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS–dilithium: Algorithm specification and supporting documentation (version 3.1). Round-3: Submission to the NIST Post-Quantum Cryptography Standardization Project (2020), https://cryptojedi.org/papers/#dilithiumnistr3
4. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 41–69. Springer (2011), https://doi.org/10.1007/978-3-642-25385-0_3
5. Boschini, C., Takahashi, A., Tibouchi, M.: Musig-l: Lattice-based multi-signature with single-round online phase. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. Lecture Notes in Computer Science, vol. 13508, pp. 276–305. Springer (2022), https://doi.org/10.1007/978-3-031-15979-4_10
6. Chen, Y.: Dualms: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023. Lecture Notes in Computer Science, vol. 14085, pp. 716–747. Springer (2023), https://doi.org/10.1007/978-3-031-38554-4_23

7. Chung, K., Fehr, S., Huang, Y., Liao, T.: On the compressed-oracle technique, and post-quantum security of proofs of sequential work. In: Canteaut, A., Standaert, F. (eds.) EUROCRYPT 2021. Lecture Notes in Computer Science, vol. 12697, pp. 598–629. Springer (2021), https://doi.org/10.1007/978-3-030-77886-6_21
8. Damgård, I., Orlandi, C., Takahashi, A., Tibouchi, M.: Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In: Garay, J.A. (ed.) PKC 2021. Lecture Notes in Computer Science, vol. 12710, pp. 99–130. Springer (2021), https://doi.org/10.1007/978-3-030-75245-3_5
9. Don, J., Fehr, S., Majenz, C.: The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. Lecture Notes in Computer Science, vol. 12172, pp. 602–631. Springer (2020), https://doi.org/10.1007/978-3-030-56877-1_21
10. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(1), 238–268 (2018), https://doi.org/10.13154/TCHES.V2018.I1.238-268
11. Fukumitsu, M., Hasegawa, S.: A lattice-based provably secure multisignature scheme in quantum random oracle model. In: Nguyen, K., Wu, W., Lam, K., Wang, H. (eds.) ProvSec 2020. Lecture Notes in Computer Science, vol. 12505, pp. 45–64. Springer (2020), https://doi.org/10.1007/978-3-030-62576-4_3
12. Grilo, A.B., Hövelmanns, K., Hülsing, A., Majenz, C.: Tight adaptive reprogramming in the QROM. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. Lecture Notes in Computer Science, vol. 13090, pp. 637–667. Springer (2021), https://doi.org/10.1007/978-3-030-92062-3_22
13. Jackson, K.A., Miller, C.A., Wang, D.: Evaluating the security of crystals-dilithium in the quantum random oracle model. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024. Lecture Notes in Computer Science, vol. 14656, pp. 418–446. Springer (2024), https://doi.org/10.1007/978-3-031-58751-1_15
14. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. Lecture Notes in Computer Science, vol. 10822, pp. 552–586. Springer (2018), https://doi.org/10.1007/978-3-319-78372-7_18
15. Lai, Q., Liu, F.H., Lu, Y., Xue, H., Yu, Y.: Scalable two-round n-out-of-n and multi-signatures from lattices in the quantum random oracle model. Cryptology ePrint Archive, Paper 2024/1574 (2024), https://eprint.iacr.org/2024/1574
16. Liu, Q., Zhandry, M.: Revisiting post-quantum fiat-shamir. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Lecture Notes in Computer Science, vol. 11693, pp. 326–355. Springer (2019), https://doi.org/10.1007/978-3-030-26951-7_12
17. Lyubashevsky, V.: Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. Lecture Notes in Computer Science, vol. 5912, pp. 598–616. Springer (2009), https://doi.org/10.1007/978-3-642-10366-7_5
18. Lyubashevsky, V., Neven, G.: One-shot verifiable encryption from lattices. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017. Lecture Notes in Computer Science, vol. 10210, pp. 293–323 (2017), https://doi.org/10.1007/978-3-319-56620-7_11
19. Lyubashevsky, V., Seiler, G.: Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. Lecture Notes in Computer Science, vol. 10820, pp. 204–224. Springer (2018), https://doi.org/10.1007/978-3-319-78381-9_8

20. National Institute of Standards and Technology: Module-lattice-based digital signature standard. Tech. Rep. Federal Information Processing Standards Publications (FIPS) 204, U.S. Department of Commerce, Washington, D.C. (2024)
21. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. Lecture Notes in Computer Science, vol. 9057, pp. 755–784. Springer (2015), https://doi.org/10.1007/978-3-662-46803-6_25
22. Watrous, J.: Zero-knowledge against quantum attacks. SIAM J. Comput. **39**(1), 25–58 (2009), https://doi.org/10.1137/060670997
23. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 758–775. Springer (2012), https://doi.org/10.1007/978-3-642-32009-5_44
24. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. Lecture Notes in Computer Science, vol. 11693, pp. 239–268. Springer (2019), https://doi.org/10.1007/978-3-030-26951-7_9