A Lattice-based Provably Secure Multisignature Scheme in Quantum Random Oracle Model*

Masayuki Fukumitsu[1] and Shingo Hasegawa[2]

[1] Faculty of Information Media, Hokkaido Information University,
Nishi-Nopporo 59-2 Ebetsu, Hokkaido, 069–8585 Japan.
fukumitsu@do-johodai.ac.jp
[2] Graduate School of Information Sciences, Tohoku University,
41 Kawauchi, Aoba-ku, Sendai, Miyagi, 980–8576 Japan.
shingo.hasegawa.b7@tohoku.ac.jp

Abstract. The multisignature schemes are attracted to utilize in some cryptographic applications such as the blockchain. Though the lattice-based constructions of multisignature schemes exist as quantum-secure multisignature, a multisignature scheme whose security is proven in the *quantum* random oracle model (QROM), rather than the classical random oracle model (CROM), is not known.
In this paper, we propose a first lattice-based multisignature scheme whose security is proven in QROM. Although our proposed scheme is based on the Dilithium-QROM signature, whose security is proven in QROM, their proof technique cannot be directly applied to the multisignature setting. The difficulty of proving the security in QROM is how to program the random oracle in the security proof. To solve the problems in the security proof, we develop several proof techniques in QROM. First, we employ the searching query technique by Targi and Unruh to convert the Dilithium-QROM into the multisignature setting. For the second, we develop a new programming technique in QROM since the conventional programming techniques seem not to work in the multisignature setting of QROM. We combine the programming technique by Unruh with the one by Liu and Zhandry. The new technique enables us to program the random oracle in QROM and construct the signing oracle in the security proof.

Keywords: Lattice Cryptography · Multisigature · Quantum Random Oracle Model · CRYSTALS-Dilithium

1 Introduction

The multisignature scheme [IN83] is a variant of digital signature schemes in the sense that a group of signers can prove their authenticity of a single message. One of the advantages is that these can reduce the size of an issued *multisignature* compared with that of all signatures generated by each signer individually. This property is suitable for IoT devices, and the multisignature schemes are recently employed in the blockchain to realize a transaction by a multi-user.

Although many multisignature schemes were proposed, most of them are based on the discrete logarithm assumption or the integer factoring assumption. Such schemes have the threat by quantum computers because Shor [Sho99] presented the quantum algorithm for breaking these assumptions. The lattice-based multisignature schemes are one of the promising candidates for quantum-resistant multisignature schemes. El Bansarkhani and Sturm [EBS16] proposed the first lattice-based multisignature scheme for the constant number of signers. Their multisignature scheme is based on a lattice-based standard signature scheme by Güneysu, Lyubashevsky and Pöppelmann (GLP) [GLP12], which is a Fiat-Shamir-type signature scheme [FS87]. Its security is proven from the Ring Short Integer Solution assumption. Fukumitsu and Hasegawa [FH19] recently enhanced their multisignature scheme concerning the tightness of security reduction. The main idea of their construction is replacing the GLP signature scheme part with the Abdalla-Fouque-Lyubashevsky-Tibouchi (AFLT) signature scheme [AFLT16] which is known as a tightly-secure Fiat-Shamir-type signature scheme from the Ring Learning with Errors assumption.

The security of these lattice-based multisignature schemes was proven only in the classical random oracle model (CROM). In CROM, an adversary is supposed to obtain any hash value from the random oracle. However, a classical query is only allowed. In order to show that the schemes have the security against quantum computers, an adversary should enable quantum queries to the random oracle. Boneh, Dagdelen, Fischlin, Lehmann, Schaffner and Zhandry [BDF+11] proposed the quantum random oracle model (QROM). In this model, an adversary can provide a quantum query to the random oracle. The security of several cryptographic schemes, e.g. [BDF+11,Zha12,Unr15,TU16,KLS18,Zha19,DFMS19,LZ19] has been proven in this model. However, there is no multisignature scheme that is provably secure in QROM.

* The preliminary version of this paper appeared in [FH20].

1.1 Our Contribution

Overview We proposed a first multisignature scheme that is provably secure in QROM. The construction of our scheme is started with replacing the AFLT signature scheme part of the Fukumitsu-Hasegawa multisignature scheme with a variant of the candidates of the post-quantum cryptographic standard [NIS17], the Dilithium-QROM [KLS18]. The Dilithium-QROM achieves the provable security in QROM under the Modulo-LWE assumption. We aim to prove the plain-public key (ppk) security of our proposed scheme in QROM. The ppk security [BN06] is considered as the standard security notion of the multisignature scheme. It means that on a given challenge public key $\mathbf{pk}^*$, any probabilistic polynomial-time (PPT), especially quantum polynomial-time (QPT), adversary $\mathcal{A}$ cannot forge a new multisignature σ^* under a group of signers which contains the signer owning $\mathbf{pk}^*$ beyond negligible probability. During that, $\mathcal{A}$ can adaptively obtain multisignatures under an arbitrarily chosen pairs of a message μ and a group of signers which contains the signer owning $\mathbf{pk}^*$ from the signing oracle simulated by a challenger, where the challenger plays the role of the signer owning $\mathbf{pk}^*$ while $\mathcal{A}$ does that of the co-signers.

Proof Technique The main idea of proving the security of our proposed multisignature scheme is employing the proof technique by Kiltz, Lyubashevsky, and Schaffner [KLS18]. They proved the security of the Fiat-Shamir-type signature scheme in QROM. Their main feature is to naturally extend the security proof in CROM by [KW03,AFLT16], which is called a *lossy ID technique*, to the QROM case. The result suggests that the security proof given in CROM by the lossy ID technique can be converted to the one in QROM. On the other hand, the security of the original scheme [FH19] of our proposed scheme was proven by using the lossy ID technique. Therefore, we expect that we can prove the security of our multisignature scheme in QROM by extending the security proof by [KLS18] to the multisignature scheme case. However, such a natural extension of [KLS18] seems not to work well. We then describe the details of the problems and their solutions.

We first briefly recap the structure of Fiat-Shamir-type signature schemes and the security proof by [KLS18]. A signature generated by such types consists of a pair (w, z) such that z is determined by using a hash value $c = H_2(w, \mu)$ of the first component w and a message μ and a secret key of a signer. Their proof is divided into three steps. In the first step, the signing oracle in the security game is replaced with a simulator that no longer uses a secret key. To achieve security in QROM, all signatures that the signing oracle replied to $\mathcal{A}$ are determined at the beginning of the game by utilizing a random function. In the second step, a public key given to $\mathcal{A}$ is switched into a "lossy" one which intuitively means a public key that has no corresponding secret key. In the third step, they evaluate the upper bound of the winning probability of the "changed" game at the previous steps. Then, they convert the challenger of the changed game into an unbounded algorithm that solves the generic search problem with bounded probabilities (GSPB) [KLS18]. They also showed that any unbounded quantum algorithm solves GSPB with negligible probability. These imply that the winning probability of the original security game is almost bounded by the probability of distinguishing the distribution of a regular public key and that of a lossy public key.

According to these steps, we attempt to prove the security of the proposed multisignature scheme. Namely, we shall construct a simulator of the signing oracle and evaluate the security game for a lossy public key by converting the challenger of the changed game into an algorithm for solving GSPB. However, it turns out that two problems obstruct the naive application of [KLS18] to our multisignature case. Furthermore, we have to overcome them to succeed in the security proof.

The first one is how to search queries to the random oracle by $\mathcal{A}$ in the QROM case. Intuitively says, a multisignature during our signing protocol consists of the summation $(\boldsymbol{w}, \boldsymbol{z}) = \sum(\boldsymbol{w}_v, \boldsymbol{z}_v)$ of all solo signatures $(\boldsymbol{w}_v, \boldsymbol{z}_v)$ of the Dilithium-QROM issued by all signers individually. Following the signing protocols by [EBS16,FH19], each signer broadcasts the hash value $\boldsymbol{g}_v = H_0(\boldsymbol{w}_v)$ of $\boldsymbol{w}_v$ to the co-signers before broadcasting $\boldsymbol{w}_v$. By utilizing this process, we aims to construct a simulator which recovers each of $\boldsymbol{w}_v$ from $\boldsymbol{g}_v$ as in [EBS16,FH19] before broadcasting $\boldsymbol{w}_v$. In the CROM case, this is done by using the hash table of H_0. However, this technique cannot be immediately utilized in the QROM case as mentioned in [BDF+11,Zha19]. This is because hash queries and these responses in QROM are in a superposition, and then recording such values to the hash table is in general difficult. To solve this problem, we employ the technique by Targi and Unruh [TU16] which is used to prove the security of a variant of the Fujisaki-Okamoto transformation [FO13] in QROM. They overcame the difficultly by simulating the random oracle using an efficiently invertible function, i.e., a polynomial. Namely, it recovers a hash query to the random oracle by computing the root of the polynomial. Since the polynomial has multiple roots, to determine a hash query from the multiple roots, they utilize a one-way injection of a public key encryption scheme under fixed public key and randomness. To employ their method, we involve a one-way injective function in our signing protocol. We also note that the post-quantum public-key encryption schemes and the one-way injective functions such as [dC17,GPV08,PW08] can be adapted as a one-way injective function.

The second one is how one can program the hash values in QROM. The simulator of the signing oracle, which plays the role of the signer owning $\mathbf{pk}^*$, is required to issue a solo signature of the Dilithium-QROM without the secret key corresponding to $\mathbf{pk}^*$. Recall that [KLS18] succeeded in constructing such a simulator by determining the signatures of all messages queried by $\mathcal{A}$ at the beginning of the game. However, this technique cannot be applied immediately in our case. This is because in the signing protocol of the multisignature scheme, a multisignature $(\boldsymbol{w}, \boldsymbol{z})$ is issued by combining each of solo signatures $(\boldsymbol{w}_v, \boldsymbol{z}_v)$ from all co-signers which are played by $\mathcal{A}$, and hence the simulator no longer issues all multisignatures up-front. In order to resolve this problem, we employ a programing technique proposed by Unruh [Unr15]. Their technique supports programming a hash value on H_2 of an input value known just before the programming. Although their technique seems suitable in our case, it requires that an input value to be hashed by H_2 must be determined just before the programming by using new random coins. On the other hand, the input values $\boldsymbol{w}$ and μ are determined before broadcasting $\boldsymbol{w}_v$ in our construction, and hence these values include no new random coin. This is a new problem to be addressed. For this new problem, we combine the programming technique by Liu and Zhandry [LZ19]. Their technique realizes the programming of an arbitrary value as a hash value. By using the technique by [LZ19], the new random coins are programmed as a hash value given by employing a new hash function H_1, and then the technique by [Unr15] can be applied to our multisignature case by adding the new random coins as the input value of H_2. Our new programming technique, which combines the two conventional techniques [Unr15,LZ19] enables us to construct the simulator in the multisiganture case even in QROM, and then realizes the natural extension of the security proof in CROM into the QROM.

1.2 Future Works

We employ a new cryptographic assumption called the *rMLWE assumption* which is a lattice analog of the rDCK assumption [BBE+18] to simulate the signing oracle. Showing the validity of the new assumption or removing it from the security proof is important future work.

Another important future work is concerning the parameters of our multisignature scheme. Although we will show the bound and the relationship among the parameters in Table 1, the concrete recommended values have not been discussed. We will determine them so that our proposed scheme takes significance in the real world.

2 Preliminaries

Let $\mathcal{A}$ be an algorithm. $y \leftarrow \mathcal{A}(x)$ means that $\mathcal{A}$ outputs y on input x when $\mathcal{A}$ is deterministic. When $\mathcal{A}$ is probabilistic, we denote this by $y \leftarrow \mathcal{A}(x; r)$, where r is an internal coin of $\mathcal{A}$. And then, $\mathcal{A}(x)$ is a random variable over the choice of r. For any Boolean formula B, $\neg B$ means that B is not satisfied. Let $\mathbb{N}$ be the set of natural numbers. A function ϵ in κ is said to be *negligible* if for any polynomial p, there exists $\kappa_0 \in \mathbb{N}$ such that $\epsilon(\kappa) < 1/p(\kappa)$ for any $\kappa \geq \kappa_0$.

Let $\mathbb{Z}$, $\mathbb{Z}_n$, and $\mathbb{F}_q$ be the ring of integers, that of residues and the finite field over q respectively, where $n \in \mathbb{N}$ and q is a prime or a power of 2. Consider a finite set X. For a probabilistic distribution D over X, $x \in_D X$ means that x is chosen according to D. $x \in_{\mathrm{U}} X$ means that x is chosen uniformly at random from X. Let Uni be a probabilistic algorithm that outputs $x \in_{\mathrm{U}} X$ on a given set X. Although the length of the representation of X may be beyond polynomial in a security parameter, we use Uni only in unbounded algorithms in this paper. For any real number $0 \leq \lambda \leq 1$, B_λ is the Bernoulli distribution, namely $\Pr_{x \in_{B_\lambda} \{0,1\}}[x = 1] = \lambda$. For a set X, $|X|$ stands for the number of elements in X, while $|a|$ does for the absolute value of a for any number a.

2.1 Quantum Computation

The state $|\psi\rangle$ of n qubits is expressed by $|\psi\rangle = \sum_{s \in \{0,1\}^n} \alpha_s |s\rangle$, where each of α_s is a complex number such that $\sum_{s \in \{0,1\}^n} |\alpha_s|^2 = 1$. And, $\{|s\rangle\}_{s \in \{0,1\}^n}$ is called *computational basis*. The qubit is said to be *in superposition* if there exists $s \in \{0,1\}^n$ such that $0 < |\alpha_s| < 1$, whereas it is *classical* otherwise. Any string $s \in \{0,1\}^n$ can be measured in the computational basis with probability $|\alpha_s|^2$. The evolution of a quantum system in a state $|\psi\rangle$ is defined by using a unitary matrix U of size $2^n \times 2^n$. For the detail, please refer to a textbook such as [NC00].

Consider an oracle O which returns an m-bit string on input n-bit string. Following [BDF+11], the quantum access to O is expressed by a unitary matrix U_O which maps $|x\rangle|\psi\rangle$ to $|x\rangle|\psi \oplus O(x)\rangle$, where $x \in \{0,1\}^n$, $\psi \in \{0,1\}^m$ and $\oplus$ denotes the bitwise exclusive-or operation. For an quantum algorithm $\mathcal{A}$, $\mathcal{A}^{|O\rangle}$ means that $\mathcal{A}$ is allowed to access the oracle O in superposition.

$\underline{\mathbf{Exp}^{\mathsf{GSPB}}_{\lambda,\mathcal{A}}(\kappa)}$

(1) $(\{\lambda(x)\}_{x\in X}, \mathrm{st}) \leftarrow \mathcal{A}_1(1^\kappa)$
(2) return 0 if $\exists x \in X$ s.t. $\lambda(x) > \lambda$.
(3) For each $x \in X$, set $g(x) \in_{B_{\lambda(x)}} \{0,1\}$
(4) $x \leftarrow \mathcal{A}_2^{|g\rangle}(\mathrm{st})$
(5) return $g(x)$

Fig. 1. The generic search problem with bounded probabilities $0 \le \lambda \le 1$

$\underline{\mathbf{Exp}^{\mathsf{Reprog}}_{\mathfrak{D},\mathfrak{R},b',\mathcal{A}}(\kappa)}$

(1) $H \in_{\mathrm{U}} \mathsf{Fun}(\mathfrak{D},\mathfrak{R})$.
(2) $\mathrm{st}_0 \leftarrow \mathcal{A}_0^{|H\rangle}(1^\kappa)$.
(3) $(w, \mathrm{st}_c) \leftarrow \mathcal{A}_C(\mathrm{st}_0)$.
(4) $c = H(w)$ if $b' = 1$, or $c \in_{\mathrm{U}} \mathfrak{R}$ and set $H(w) = c$ otherwise.
(5) return $\mathcal{A}_2^{|H\rangle}(c, \mathrm{st}_c)$.

Fig. 2. The reprogram problem in QROM

We recall the generic search problem with bounded probabilities (GSPB), which is defined in [KLS18] (Fig. 1). Intuitively, the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the GSPB game aims to find a string $x \in X$ such that $g(x) = 1$. Here, $\mathcal{A}$ can declare the probability $\lambda(x)$ that $g(x) = 1$ for each $x \in X$ under the constraint that $\lambda(x) < \lambda$ for some designated $0 \le \lambda \le 1$, and then the challenge C determines the function $g(x)$ according to $B_{\lambda(x)}$ for each $x \in X$.

Lemma 1 ([KLS18, Lemma 2.1], Generic Search Problem with Bounded Probabilities). *For any real number $0 \le \lambda \le 1$, and any unbounded quantum algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ making at most Q queries to $g : X \to \{0,1\}$, we have* $\Pr\left[\mathbf{Exp}^{\mathsf{GSPB}}_{\lambda,\mathcal{A}}(\kappa) = 1\right] \le 8\lambda(Q+1)^2$.

2.2 Quantum Random Oracle Model

In this subsection, we explain the notion of the quantum random oracle model and the ways to simulate it. Let $\mathsf{Fun}(\mathfrak{D},\mathfrak{R})$ be the set of all functions $H : \mathfrak{D} \to \mathfrak{R}$. Then, the *quantum random oracle model (QROM)* is a security model in which any adversary $\mathcal{A}$ obtains hash values from the random oracle by accessing the oracle in superposition. For a hash function $H \in \mathsf{Fun}(\mathfrak{D},\mathfrak{R})$, we write $\mathcal{A}^{|H\rangle}$ to denote that $\mathcal{A}$ can access to the random oracle H in superposition.

The several ways to simulate the random oracle are known. We recall the techniques of replacing the random oracle with several functions and that of involving the compressed oracle proposed by [Zha19].

The first way to simulate the random oracle is to replace the random oracle with a random function $H \in_{\mathrm{U}} \mathsf{Fun}(\mathfrak{D},\mathfrak{R})$. In this way, a challenger selects H at the beginning of the security game, and replies $H(x)$ on a given query x in superposition. The random oracle is simulated perfectly by this technique, because the uniformly random choice of a function from $\mathsf{Fun}(\mathfrak{D},\mathfrak{R})$ implies that the hash value $H(x)$ is uniformly distributed over $\mathfrak{R}$ for any value x. The following lemma says that we can replace a hash value with a random value even in QROM. Namely, we can use the programming technique with random values in this simulation. The *collision-entropy* [Unr15] of a random variable X is defined by $-\log \Pr[X = X']$, where X' is a random variable X' which is independent of X, but has the same distribution.

Lemma 2 ([Unr15]). *Let $\mathfrak{D}$, $\mathfrak{R}$ be finite sets, and let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_C, \mathcal{A}_2)$ be an algorithm such that $\mathcal{A}_0$ and $\mathcal{A}_2$ are quantum algorithms which can access to the random oracle, and $\mathcal{A}_C$ is a probabilistic classical algorithm. Assume that $\mathcal{A}_C(\mathrm{st}_0)$ has collision-entropy at least ι for any input st_0. For $\mathbf{Exp}^{\mathsf{Reprog}}_{\mathfrak{D},\mathfrak{R},b',\mathcal{A}}$ depicted in Fig. 2, and the number q_A of queries by $\mathcal{A}_0$, we have*

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{Reprog}}_{\mathfrak{D},\mathfrak{R},0,\mathcal{A}}(1^\kappa) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{Reprog}}_{\mathfrak{D},\mathfrak{R},1,\mathcal{A}}(1^\kappa) = 1\right]\right| \le (4 + \sqrt{2})\sqrt{q_A}2^{-\iota/4}.$$

Moreover, the following lemma guarantees that H can be replaced with a $2Q$-wise independent function in the simulation of the random oracle.

Lemma 3 ([Zha12]). *Let $\mathcal{A}$ be a quantum algorithm, which can access the random oracle in superposition at most Q times. Then, a simulator of the random oracle which uses a function $H \in_{\mathrm{U}} \mathsf{Fun}(\{0,1\}^n, \{0,1\}^m)$ is perfectly indistinguishable from the one which uses a $2Q$-wise independent function from $\mathcal{A}$. A random polynomial f_{2Q} of degree $2Q$ over the finite field $\mathbb{F}_{2^m}$ is $2Q$-wise independent.*

Note that the roots of f_{2Q} can be computed in polynomial-time.

The second way to replace the random oracle is the use of the compressed oracle [Zha19]. The concept of compressed oracles was introduced to simulate the random oracle, as in the case of the classical ROM. Namely, a database of all pairs of a random oracle query and its hash value is utilized in the simulation. We now briefly

recall the notion of the compressed Fourier oracle and the compressed phase oracle, respectively. For the detail, please refer to the papers such as [Zha19,LZ19]. Consider that $\mathfrak{R} = \{0,1\}^\theta$, and a superposition of databases D which contains pairs (x, y) of an input and an output of a hash function H Hence, $(x, y) \in D$ means that $H(x) = y$. At the beginning of a security game, D is initialized to a pure state over empty databases. For each query, the *compressed Fourier oracle* returns its hash value in superposition after operating $D \oplus (x, y)$. The operation $D \oplus (x, y)$ is defined in the following way:

- D if $y = 0$,
- $D \cup \{(x, y)\}$ else if $y \neq 0 \wedge D(x) = \perp$,
- $D \setminus \{(x, y)\}$ else if there exists x such that $D(x) = y' \wedge y + y' \equiv 0 \pmod{2^\theta}$, or
- $D \setminus \{(x, y)\} \cup \{(x, y + y')\}$ otherwise.

The *compressed phase oracle* is intuitively obtained by applying the quantum Fourier transformation to D. For a tuple $|x, y, D\rangle$, it proceeds as follows:

- If there exists y' such that $(x, y') \in D$, then y' part is updated to $y + y'$. Namely, a phase $\omega_{\mathfrak{R}}^{yy'}$ will be added to the state, where $\omega_{\mathfrak{R}} = e^{2\pi i/|\mathfrak{R}|}$.
- Otherwise, (x, y) is added to D. Namely, a superposition is appended to $|x\rangle \otimes \sum_{y'} \omega_{\mathfrak{R}}^{yy'} |y'\rangle$.
- Finally, it checks that there exists no $(x, 0)$ by converting it into the compressed Fourier oracle model. If there are such pairs, then these are deleted.

Moreover, any QPT algorithm can simulate both the compressed Fourier oracle and the compressed phase oracle straightforwardly.

Lemma 4 ([Zha19]). *Let $\mathcal{A}$ be a quantum algorithm which makes queries to a random oracle $H : \mathfrak{D} \to \{0,1\}^n$, and outputs tuples $(x_1, \ldots, x_k, y_1, \ldots, y_k, z)$. Let R be a collection of such tuples. Suppose that with probability p, $\mathcal{A}$ outputs a tuple such that (1) the tuple is in R and (2) for all i, $H(x_i) = y_i$. Now consider running $\mathcal{A}$ with each of the compressed Fourier oracle and the compressed phase oracle, and suppose that the database D is measured after $\mathcal{A}$ produces its output. Let p' be the probability that the conditions (1) and (2) hold in this running case. Then, we have $\sqrt{p} \leq \sqrt{p'} + \sqrt{k/2^n}$.*

Liu and Zhandry [LZ19] recently introduced a new compressed oracle, which is called *almost compressed Fourier oracle*, by combining the compressed Fourier oracle with the compressed phase oracle. In the use of the *almost compressed Fourier oracle*, some points are called *special points*. Moreover, they are dealt with the (uncompressed) phase oracle. On the other hand, other points are dealt with by the compressed Fourier oracle. Namely, the entire initial state of the oracle is expressed by $\sum_{x,y} \alpha_{x,y} |x, y\rangle \otimes (|D_0\rangle \otimes \sum_r |r\rangle)$, where D_0 is an empty database. For each query, the pair (x, y) of the input and the output is dealt with the one on the compressed Fourier oracle if x is not a special point. Otherwise, the second part of the oracle is updated as a phase oracle: $\alpha_{x,y,D,y'} |x, y\rangle \otimes |D\rangle \otimes \sum_r \omega_{\mathfrak{R}}^{y'r} |r\rangle \Longrightarrow \alpha_{x,y,D,y'} |x, y\rangle \otimes |D\rangle \otimes \sum_r \omega_{\mathfrak{R}}^{(y'+y)r} |r\rangle$.

As a programming technique on the almost compressed Fourier oracle, the following lemma holds. This lemma can be extended to a case for several special points as noted in [LZ19].

Lemma 5 ([LZ19, Corollary 7]). *Consider a random oracle which maps an element in $\{0,1\}^n$ into the one in $\{0,1\}^n$. Assume that an adversary $\mathcal{A}$ is interacting with a simulator of an almost compressed phase oracle where the i-th oracle query x^* is regarded as the special point. Instead of appending $\sum_y |y\rangle$ into the database for the i-th query, $|r\rangle$ for randomly chosen r is appended. Then, $\mathcal{A}$ and the simulator continue the running, and eventually the simulator measures the output registers. Then, $\mathcal{A}$ cannot distinguish such a replacement.*

More precisely, consider the following situation. Let S be a set of all possible entire states w of $\mathcal{A}$ and compressed database $D \cup \{(x^*, r)\}$. We define a measurement as follows: Let γ be the probability that the measurement gives 0 in the game where $\sum_y |y\rangle$ is appended into D as the hash value of x^*, and γ' be the one where $|r\rangle$ is appended. Then, we have $\gamma = \gamma'$.

2.3 Lattice

Consider $q, n \in \mathbb{N}$. Let R and R_q be the rings $\mathbb{Z}[X]/(X^n + 1)$ and $\mathbb{Z}_q/(X^n + 1)$, respectively. An element $a = \sum_{i=0}^{n-1} a_i x^i$ in R or R_q is represented by its coefficients vector $(a_0, a_1, \ldots, a_{n-1})$. Regular font letters stand for elements in R or R_q, while bold font letters stand for vectors and matrices over R or R_q.

For $\alpha \in \mathbb{N}$, $r = x \bmod^+ \alpha$ denotes the ordinary residue of the division over α, namely $0 \leq r < \alpha$, while $r = x \bmod^\pm \alpha$ does the *centered* residue such that $-\alpha/2 < r \leq \alpha/2$ when α is even, or $-(\alpha-1)/2 \leq r \leq (\alpha-1)/2$ otherwise. Hereafter, each element x in $\mathbb{Z}_q$ including coefficients of $a \in R_q$ is represented by the centered

Init C generates $\mathbf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ and C's key pair $(\mathbf{pk}^*, \mathbf{sk}^*) \leftarrow \mathsf{KGen}(\mathbf{pp})$. Then C sends $(\mathbf{pp}, \mathbf{pk}^*)$ to $\mathcal{F}$.

Sign For a query $(\mu^{(i)}, \mathsf{PK}^{(i)})$ of a message $\mu^{(i)}$ and a public key set $\mathsf{PK}^{(i)} = \left\{\mathbf{pk}_u^{(i)}\right\}_{u=1}^{U}$ including $\mathbf{pk}^*$, C runs the protocol by playing the role of the signer which generates $\mathbf{pk}^*$ and then returns a multisignature $\sigma^{(i)}$ to $\mathcal{F}$. We assume without loss of generality that $\mathbf{pk}_1^{(i)} = \mathbf{pk}^*$. Here, $\mathcal{F}$ plays the role of the co-signers.

Challenge For a final output $(\mathsf{PK}^*, \mu^*, \sigma^*)$ of $\mathcal{F}$, $\mathcal{F}$ is said to *win this game* if the following conditions hold:

(1) $\mathbf{pk}^* \in \mathsf{PK}^*$,
(2) (μ^*, PK^*) is not queried in **Sign** phase, and
(3) $\mathsf{Ver}(\mathbf{pp}, \mathsf{PK}^*, \mu^*, \sigma^*) = 1$.

Fig. 3. The description of the ppk game

residue instead of the ordinary one. For any $r \in \mathbb{Z}_q$, $|r|_\infty$ is defined by $|r \bmod^{\pm} q|$. By using this notation, any element $r = x \bmod^{\pm} \alpha$ satisfies that $|r|_\infty \leq \alpha/2$ for any natural number $\alpha \leq q$ and integer x. The L^∞-norm and the L^2-norm of a polynomial $a = (a_0, a_1, \ldots, a_{n-1}) \in R_q$ are represented by $|a|_\infty = \max_{0 \leq i \leq n-1} |a_i|_\infty$ and $|a|_2 = \sqrt{\sum_{i=0}^{n-1} |a_i|_\infty^2}$, respectively. The L^∞-norm and the L^2-norm of a vector $\boldsymbol{a} = (a_1, \ldots, a_\ell) \in R_q^\ell$ for any $\ell \in \mathbb{N}$ are represented by $|\boldsymbol{a}|_\infty = \max_{1 \leq i \leq \ell} |a_i|_\infty$ and $|\boldsymbol{a}|_2 = \sqrt{\sum_{i=1}^{\ell} |a_i|_\infty^2}$, respectively. For any $\eta \in \mathbb{N}$, let S_η be a set of all elements $a \in R$ such that $|a|_\infty \leq \eta$.

Consider $k, \ell \in \mathbb{N}$, a probability distribution D over R_q. For any polynomial T_{MLWE} and any function ϵ_{MLWE}, the $(T_{\mathrm{MLWE}}, \epsilon_{\mathrm{MLWE}}, D)$-*Modulo-LWE assumption* ($(T_{\mathrm{MLWE}}, \epsilon_{\mathrm{MLWE}}, D)$-MLWE assumption) states that for any QPT adversary $\mathcal{A}$ whose running time is at most T_{MLWE}, it holds that $\left|P_0^{\mathrm{MLWE}} - P_1^{\mathrm{MLWE}}\right| \leq \epsilon_{\mathrm{MLWE}}(\kappa)$ for any κ, where

$$P_0^{\mathrm{MLWE}} = \Pr\left[\mathcal{A}(\boldsymbol{A}, \boldsymbol{t}) = 1 : \boldsymbol{A} \in_{\mathrm{U}} R_q^{k \times \ell}, \boldsymbol{s}_1, \in_D R_q^\ell, \boldsymbol{s}_2 \in_D R_q^k, \boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}_1 + \boldsymbol{s}_2\right],$$
$$P_1^{\mathrm{MLWE}} = \Pr\left[\mathcal{A}(\boldsymbol{A}, \boldsymbol{t}) = 1 : \boldsymbol{A} \in_{\mathrm{U}} R_q^{k \times \ell}, \boldsymbol{t} \in_{\mathrm{U}} R_q^k\right].$$

We also consider another assumption which is an analog of the rejected-DCK assumption [BBE+18] to the MLWE case. Let

$$\mathcal{CH} = \left\{c \in R \mid |c|_\infty = 1 \wedge |c|_2 = \sqrt{46}\right\}.$$

For any polynomial T_{rMLWE}, any function $\epsilon_{\mathrm{rMLWE}}$ and any constants γ' and β, the $(T_{\mathrm{rMLWE}}, \epsilon_{\mathrm{rMLWE}}, D, \gamma', \beta)$-*rejected-MLWE assumption* ($(T_{\mathrm{rMLWE}}, \epsilon_{\mathrm{rMLWE}}, D, \gamma', \beta)$-rMLWE assumption) states that for any QPT adversary $\mathcal{A}$ whose running time is at most T_{rMLWE}, it holds that $\left|P_0^{\mathrm{rMLWE}} - P_1^{\mathrm{rMLWE}}\right| \leq \epsilon_{\mathrm{rMLWE}}(\kappa)$ for any κ, where

$$P_0^{\mathrm{rMLWE}} = \Pr\left[\begin{array}{ll} & \boldsymbol{A} \in_{\mathrm{U}} R_q^{k \times \ell}, \boldsymbol{s} \in_D R_q^\ell, \\ \mathcal{A}(\boldsymbol{A}, \boldsymbol{w}, c) = 1 : & \boldsymbol{y} \in_{\mathrm{U}} S_{\gamma'-1}^\ell, \quad \mid \boldsymbol{y} + c\boldsymbol{s} \geq \gamma' - \beta \\ & \boldsymbol{w} = \boldsymbol{A}\boldsymbol{y}, c \in_{\mathrm{U}} \mathcal{CH} \end{array}\right],$$
$$P_1^{\mathrm{rMLWE}} = \Pr\left[\begin{array}{ll} & \boldsymbol{A} \in_{\mathrm{U}} R_q^{k \times \ell}, \boldsymbol{s} \in_D R_q^\ell, \\ \mathcal{A}(\boldsymbol{A}, \boldsymbol{w}, c) = 1 : & \boldsymbol{y} \in_{\mathrm{U}} S_{\gamma'-1}^\ell, \quad \mid \boldsymbol{y} + c\boldsymbol{s} \geq \gamma' - \beta \\ & \boldsymbol{w} \in_{\mathrm{U}} R_q^k, c \in_{\mathrm{U}} \mathcal{CH} \end{array}\right].$$

2.4 Multisigature Scheme

We now introduce the notion of the multisignature and its security. A *multisignature scheme* consists of the following four tuples $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$. Setup and KGen are PPT algorithms. $\mathsf{Setup}(1^\kappa)$ returns a system parameter $\mathbf{pp}$ on a security parameter κ. Each signer generates a pair $(\mathbf{sk}, \mathbf{pk})$ of a secret key $\mathbf{sk}$ and a public key $\mathbf{pk}$ by $\mathsf{KGen}(\mathbf{pp})$. The signing process can be done by running the designated multiparty protocol during multiple signers to issue a signature σ on a message μ and a set PK of public keys. Each signer executes the interactive polynomial-time algorithm $\mathsf{Sig}(\mathbf{pp}, \mathbf{sk}, \mathsf{PK}, \mu)$ in order to execute this protocol. Ver is a deterministic polynomial-time algorithm which returns 1 on input $(\mathbf{pp}, \mathsf{PK}, \mu, \sigma)$ if σ is a valid signature for (PK, μ).

Correctness Let $\mathsf{MSig} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sig}, \mathsf{Ver})$ be a multisignature scheme. For any $\mathbf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ and any message μ, consider the situation where for each $1 \leq v \leq U$, a v-th signer generates own key pair $(\mathbf{sk}_v, \mathbf{pk}_v) \leftarrow \mathsf{KGen}(\mathbf{pp})$, executes $\mathsf{Sig}(\mathbf{pp}, \mathbf{sk}_v, \mathsf{PK}, \mu)$ in order to run the protocol, and then obtains a signature σ, where $\mathsf{PK} = \{\mathbf{pk}_u\}_{u=1}^{U}$. MSig satisfies the *correctness* if it always holds $\mathsf{Ver}(\mathbf{pp}, \mathsf{PK}, \mu, \sigma) = 1$.

$\mathsf{Power2Round}_q(r \in \mathbb{Z}_q, d \in \mathbb{N})$:
(1) $r = r \bmod^+ q$.
(2) $r_{\mathsf{L}} = r \bmod^\pm 2^d$.
(3) return $(r - r_{\mathsf{L}})/2^d$.

$\mathsf{UseHint}_q(h \in \{0,1\}, r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N})$:
(1) $m = (q-1)/\alpha$.
(2) $(r_{\mathsf{H}}, r_{\mathsf{L}}) \leftarrow \mathsf{Decompose}_{\mathsf{q}}(r, \alpha)$.
(3) return r_{H} if $h = 0$.
(4) return $(r_{\mathsf{H}} + 1) \bmod^+ m$ if $r_{\mathsf{L}} > 0$.
(5) return $(r_{\mathsf{H}} - 1) \bmod^+ m$ if $r_{\mathsf{L}} \leq 0$.

$\mathsf{MakeHint}_q(z \in \mathbb{Z}_q, r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N})$:
(1) $r_{\mathsf{H}} \leftarrow \mathsf{HighBits}_q(r, \alpha)$.
(2) $v_{\mathsf{H}} \leftarrow \mathsf{HighBits}_q(r + z, \alpha)$.
(3) return 0 if $r_{\mathsf{H}} = v_{\mathsf{H}}$, or 1 otherwise.

$\mathsf{Decompose}_{\mathsf{q}}(r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N})$:
(1) $r = r \bmod^+ q$.
(2) $r_{\mathsf{L}} = r \bmod^\pm \alpha$.
(3) return $(r_{\mathsf{H}}, r_{\mathsf{L}}) = (0, r_{\mathsf{L}} - 1)$ if $r - r_{\mathsf{L}} = q - 1$.
(4) $r_{\mathsf{H}} = (r - r_{\mathsf{L}})/\alpha$.
(5) return $(r_{\mathsf{H}}, r_{\mathsf{L}})$.

$\mathsf{HighBits}_q(r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N})$:
(1) $(r_{\mathsf{H}}, r_{\mathsf{L}}) \leftarrow \mathsf{Decompose}_{\mathsf{q}}(r, \alpha)$
(2) return r_{H}

$\mathsf{LowBits}_q(r \in \mathbb{Z}_q, \alpha \in 2\mathbb{N})$:
(1) $(r_{\mathsf{H}}, r_{\mathsf{L}}) \leftarrow \mathsf{Decompose}_{\mathsf{q}}(r, \alpha)$
(2) return r_{L}

Fig. 4. Supporting algorithms [DKL+18]

Security The plain public key (ppk) security [BN06] for multisignatures is defined by the ppk game between the challenger C and the forger $\mathcal{F}$ (Fig 3).

Definition 6. *Let T be a polynomial, ϵ be a function and U be the number of signers. A multisignature scheme* MSig *with U signers is (T, U, ϵ, Q_S)*-ppk secure *if for any forger $\mathcal{F}$ running in time T which makes at most Q_S queries in* **Sign** *phase, the probability that $\mathcal{F}$ wins the ppk game is ϵ.*

Especially, MSig *is said to be $(T, U, \epsilon, Q_S, Q_0, Q_1, \ldots)$*-ppk secure in QROM *if* MSig *is (T, U, ϵ, Q_S)-ppk secure and the number of queries to the random oracle H_i is at most Q_i for all i.*

3 A Dilithium-based Multisignature Scheme

We propose a multisignature scheme that is based on the Dilithium-QROM signature scheme [KLS18]. In their scheme, several supporting algorithms are employed to optimize efficiency. We start by introducing several supporting algorithms in the next subsection.

3.1 Supporting Algorithms

The basic strategy of the Dilithium-QROM signature scheme is to reduce the key size and the signature size is to ignore the residues of values by dividing them by some designated even number $\alpha < q$. In order to apply such a strategy to a signature scheme, the Dilithium-QROM signature scheme employs several supporting algorithms which are summarized in Fig. 4.

We note that the supporting algorithms can be applied to vectors or polynomials rather than scalars, by applying each of these elements in the vectors. For example, $\mathsf{Decompose}_{\mathsf{q}}(a, \alpha)$ for $a = (a_0, a_1, \ldots, a_{n-1}) \in R_q$ means that $\mathsf{Decompose}_{\mathsf{q}}$ is applied to each coefficients of a with the same modulus α. Namely, we have

$$\mathsf{Decompose}_{\mathsf{q}}(a, \alpha) = (\mathsf{Decompose}_{\mathsf{q}}(a_0, \alpha), \ldots, \mathsf{Decompose}_{\mathsf{q}}(a_{n-1}, \alpha)).$$

We can employ the same manner for a vector over R_q. For a vector $\boldsymbol{h} = (h_0, \ldots, h_{n-1}) \in \{0,1\}^n$, $\mathsf{UseHint}_q(\boldsymbol{h}, a, \alpha)$ means $(\mathsf{UseHint}_q(h_0, a_0, \alpha), \ldots, \mathsf{UseHint}_q(h_{n-1}, a_{n-1}, \alpha))$. $\mathsf{MakeHint}_q$ is also applied to the same manner.

3.2 Proposed Scheme

We present our Dilithium-based multisignature scheme in this subsection. We consider the number U of signers and parameters $(q, n, k, \ell, d, \gamma, \gamma', \eta, \beta, \theta)$ which are defined as Table 1. Let Sam be an algorithm which outputs a matrix $\boldsymbol{A} \in_{\mathrm{U}} R_q^{k \times \ell}$ on input $\varrho \in \{0,1\}^{256}$. And, let H_0, H_1 and H_2 be functions that $H_0 : \mathfrak{D}_0 \to \mathfrak{R}_0$, $H_1 : \mathfrak{D}_1 \to \{0,1\}^\theta$ and $H_2 : \mathfrak{D}_2 \to \mathcal{CH}$, respectively. We use a one-way injective function OW. The proposed multisignature scheme is given in Fig. 5.

Table 1. Parameters of the proposed multisignature scheme, where $\boldsymbol{A}$, $\boldsymbol{t}$, $\boldsymbol{w}_{\mathsf{H}}$, $\boldsymbol{s}_1$, $\boldsymbol{s}_2$ and H_1 will be defined in Fig. 5.

U	# of signers	constant
q	ring modulus	$q \equiv 5 \pmod 8$
n	ring dimension	
(k, ℓ)	dimension of matrix $\boldsymbol{A}$	$\ell \leq k$
d	dropped bit from $\boldsymbol{t}$	$46 \cdot 2^{d-1} \leq \gamma,\ 2^d < 4U(\gamma' - \beta) - 4$
γ	max. coefficient of $\boldsymbol{w}_{\mathsf{H}}$	$q > 4U\gamma,\ q \equiv 1 \pmod{2U\gamma},\ \beta \ll \gamma,\ 4U\gamma + 2 \leq \sqrt{q/2}$
γ'	$\approx$ max. sig. coefficient	$\beta \ll \gamma',\ 2\gamma' \leq \sqrt{q/2},\ 2U(\gamma' - \beta) - 2 \leq \sqrt{q/2}$
η	max. coefficients of $\boldsymbol{s}_1, \boldsymbol{s}_2$	
β	$\beta = 46\eta$	$\forall c \in \mathcal{CH}, \forall s \in S_\eta, \lvert cs\rvert_\infty < \beta$
θ	length of hash values of H_1	polynomial

$\mathsf{Setup}(1^\kappa)$ returns $\varrho \in_{\mathrm{U}} \{0,1\}^{256}$.

$\mathsf{KGen}(\varrho)$ returns a key pair $(\mathbf{sk}_v, \mathbf{pk}_v)$ which is computed as follows:

(1) $\boldsymbol{A} = \mathsf{Sam}(\varrho)$.
(2) $\mathbf{sk}_v = (\boldsymbol{s}_{v,1}, \boldsymbol{s}_{v,2}) \in_{\mathrm{U}} S_\eta^\ell \times S_\eta^k$.
(3) $\boldsymbol{t}_v = \boldsymbol{A}\boldsymbol{s}_{v,1} + \boldsymbol{s}_{v,2}$.
(4) decomposite $\boldsymbol{t}_v$ into $(\boldsymbol{t}_{v,\mathsf{H}}, \boldsymbol{t}_{v,\mathsf{L}})$ according to $\mathsf{Power2Round}_q$ such that $\boldsymbol{t}_v = 2^d \cdot \boldsymbol{t}_{v,\mathsf{H}} + \boldsymbol{t}_{v,\mathsf{L}}$.
(5) $\mathbf{pk}_v = (\boldsymbol{t}_{v,\mathsf{H}}, \boldsymbol{t}_{v,\mathsf{L}})$.

$\mathsf{Sig}(\varrho, \mathsf{PK}, \mathbf{sk}_v, \mu)$ returns $\sigma = (\boldsymbol{w}_{\mathsf{H}}, \boldsymbol{z}, \boldsymbol{h})$ on the message μ under the public key set $\mathsf{PK} = \{\mathbf{pk}_u\}_{u=1}^U$ by running the following protocol, where $\mathbf{pk}_u = (\boldsymbol{t}_{u,\mathsf{H}}, \boldsymbol{t}_{u,\mathsf{L}})$ for each $1 \leq u \leq U$. The following protocol is described in the viewpoint of the signer owning the public key $\mathbf{pk}_v$ which corresponds to $\mathbf{sk}_v$. If the time of the iteration exceeds E which is the expected time of the iteration, it returns $\sigma = \perp$.

1st stage proceed to the followings, and then broadcast $(\boldsymbol{g}_v, \boldsymbol{p}_v)$ to the co-signers:
(1.1) $\boldsymbol{A} = \mathsf{Sam}(\varrho)$; $\boldsymbol{y}_v \in_{\mathrm{U}} S_{\gamma'-1}^\ell$; $\boldsymbol{w}_v = \boldsymbol{A}\boldsymbol{y}_v$.
(1.2) $\boldsymbol{g}_v = H_0(\boldsymbol{w}_v)$; $\boldsymbol{p}_v = \mathsf{OW}_{\boldsymbol{g}_v}(\boldsymbol{w}_v)$.

2nd stage After receiving $\{(\boldsymbol{g}_u, \boldsymbol{p}_u)\}_{u \neq v}$, broadcast $\boldsymbol{w}_v$ to the co-signers.

3rd stage After receiving $\{\boldsymbol{w}_u\}_{u \neq v}$, proceed as follows, and then broadcast $\boldsymbol{z}_v$ to the co-signers:
(3.1) abort if $\neg(\boldsymbol{g}_u = H_0(\boldsymbol{w}_u) \wedge \boldsymbol{p}_u = \mathsf{OW}_{\boldsymbol{g}_u}(\boldsymbol{w}_u))$ for some $u \neq v$.
(3.2) $\boldsymbol{w} = \sum_{u=1}^U \boldsymbol{w}_u$; $\boldsymbol{w}_{\mathsf{H}} = \mathsf{HighBits}_q(\boldsymbol{w}, 2U\gamma)$.
(3.3) $\tau_v = H_1(\mathbf{pk}_v, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK}, \mu)$.
(3.4) $c_v = H_2(\mathbf{pk}_v, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK} \setminus \{\mathbf{pk}_v\}, \mu, \tau_v)$.
(3.5) $\boldsymbol{z}_v = \boldsymbol{y}_v + c_v \boldsymbol{s}_{v,1}$.
(3.6) restart if $\neg(\lvert \boldsymbol{z}_v \rvert_\infty < \gamma' - \beta)$.

4th stage After receiving $\{\boldsymbol{z}_u\}_{u \neq v}$, proceed as follows:
(4.1) $\boldsymbol{z} = \sum_{u=1}^U \boldsymbol{z}_u$.
(4.2) restart if $\neg\Big(\Big\lVert \mathsf{LowBits}_q\Big(\boldsymbol{A}\boldsymbol{z} - \sum_{u=1}^U c_u \boldsymbol{t}_u, 2U\gamma\Big)\Big\rVert_\infty < U(\gamma - \beta)\Big)$.
(4.3) $\boldsymbol{h} = \mathsf{MakeHint}_q\Big(-\sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{L}}, \boldsymbol{A}\boldsymbol{z} - 2^d \sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{H}}, 2U\gamma\Big)$.

$\mathsf{Ver}(\varrho, \mathsf{PK}, \mu, (\boldsymbol{w}_{\mathsf{H}}, \boldsymbol{z}, \boldsymbol{h}))$ returns 1 if the following holds, where $\boldsymbol{A} = \mathsf{Sam}(\varrho)$, $\mathsf{PK} = \{\mathbf{pk}_u\}_{u=1}^U$, $\mathbf{pk}_u = (\boldsymbol{t}_{u,\mathsf{H}}, \boldsymbol{t}_{u,\mathsf{L}})$, $\tau_u = H_1(\mathbf{pk}_u, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK}, \mu)$ and $c_u = H_2(\mathbf{pk}_u, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK} \setminus \{\mathbf{pk}_u\}, \mu, \tau_u)$, for each $1 \leq u \leq U$:
(a) $\lvert \boldsymbol{z} \rvert_\infty < U(\gamma' - \beta)$.
(b) $\boldsymbol{w}_{\mathsf{H}} = \mathsf{UseHint}_q(\boldsymbol{h}, \boldsymbol{x}, 2U\gamma)$, where $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{z} - 2^d \sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{H}}$.

Fig. 5. Proposed multisignature scheme

3.3 Correctness

In this subsection, we show the correctness of our proposed multisignature scheme. We use the following lemmas for the supporting algorithms.

Lemma 7 (Lemma 4.1 [KLS18]). *Suppose that q and α are positive integers such that $q > 2\alpha$, $q \equiv 1 \pmod \alpha$ and α is even. Let $\boldsymbol{r}, \boldsymbol{z}$ be vectors over R_q satisfying that $\lvert \boldsymbol{z} \rvert_\infty \leq \alpha/2$, and let $\boldsymbol{h}, \boldsymbol{h}'$ be vectors of bits. Then, we have the followings:*

(1) $\mathsf{UseHint}_q(\mathsf{MakeHint}_q(\boldsymbol{z}, \boldsymbol{r}, \alpha), \boldsymbol{r}, \alpha) = \mathsf{HighBits}_q(\boldsymbol{r} + \boldsymbol{z}, \alpha)$.
(2) If $\boldsymbol{v}_{\mathsf{H}} = \mathsf{UseHint}_q(\boldsymbol{h}, \boldsymbol{r}, \alpha)$, *then* $\lvert \boldsymbol{r} - \boldsymbol{v}_{\mathsf{H}}\alpha \rvert_\infty \leq \alpha + 1$.
(3) For any $\boldsymbol{h}, \boldsymbol{h}'$, *if* $\mathsf{UseHint}_q(\boldsymbol{h}, \boldsymbol{r}, \alpha) = \mathsf{UseHint}_q(\boldsymbol{h}', \boldsymbol{r}, \alpha)$, *then* $\boldsymbol{h} = \boldsymbol{h}'$.

Lemma 8 (Lemma 4.2 [KLS18]). *If* $\lvert \boldsymbol{s} \rvert_\infty \leq \beta$ *and* $\big\lVert \mathsf{LowBits}_q(\boldsymbol{r}, \alpha) \big\rVert_\infty < \alpha/2 - \beta$, *then* $\mathsf{HighBits}_q(\boldsymbol{r}, \alpha) = \mathsf{HighBits}_q(\boldsymbol{r} + \boldsymbol{s}, \alpha)$.

Verifying Conditions Let $\varrho \leftarrow \mathsf{Setup}(1^\kappa)$. Consider the following situation. For each $1 \le v \le U$, a v-th signer generates own key pair $(\mathbf{sk}_v, \mathbf{pk}_v) \leftarrow \mathsf{KGen}(\varrho)$, executes $\mathsf{Sig}(\varrho, \mathsf{PK}, \mathbf{sk}_v, \mu)$, and then obtains a signature $\sigma = (\boldsymbol{w}_1, \boldsymbol{z}, \boldsymbol{h})$, where $\mathsf{PK} = \{\mathbf{pk}_u\}_{u=1}^U$. Assume that $\sigma \neq \perp$. Namely, the protocol has passed the conditions in the processes (3.6) and (4.2). This implies that $|\boldsymbol{z}_v|_\infty < \gamma' - \beta$ for any $1 \le v \le U$ and $\left|\mathsf{LowBits}_q\left(\boldsymbol{A}\boldsymbol{z} - \sum_{u=1}^U c_u \boldsymbol{t}_u, 2U\gamma\right)\right|_\infty < U(\gamma - \beta)$. It follows that

$$|\boldsymbol{z}|_\infty = \left|\sum_{u=1}^U \boldsymbol{z}_u\right|_\infty \le \sum_{u=1}^U |\boldsymbol{z}_u|_\infty < \sum_{u=1}^U (\gamma' - \beta) = U(\gamma' - \beta),$$

from the processes (3.6) and (4.1). Then the verifying condition (a) of Ver is satisfied.

For each $1 \le u \le U$, since $\boldsymbol{z}_u = \boldsymbol{y}_u + c_u \boldsymbol{s}_{u,1}$, $\boldsymbol{w}_u = \boldsymbol{A}\boldsymbol{y}_u$ and $\boldsymbol{t}_u = 2^d \cdot \boldsymbol{t}_{u,\mathsf{H}} + \boldsymbol{t}_{u,\mathsf{L}} = \boldsymbol{A}\boldsymbol{s}_{u,1} + \boldsymbol{s}_{u,2}$, we have

$$\boldsymbol{A}\boldsymbol{z}_u - c_u \boldsymbol{t}_u = \boldsymbol{A}(\boldsymbol{y}_u + c_u \boldsymbol{s}_{u,1}) - c_u \boldsymbol{t}_u = \boldsymbol{A}\boldsymbol{y}_u + c_u(\boldsymbol{A}\boldsymbol{s}_{u,1} - \boldsymbol{t}_u) = \boldsymbol{w}_u - c_u \boldsymbol{s}_{u,2}. \tag{1}$$

It follows that

$$\boldsymbol{A}\boldsymbol{z} - \sum_{u=1}^U c_u \boldsymbol{t}_u = \boldsymbol{A}\sum_{u=1}^U \boldsymbol{z}_u - \sum_{u=1}^U c_u \boldsymbol{t}_u = \sum_{u=1}^U (\boldsymbol{A}\boldsymbol{z}_u - c_u \boldsymbol{t}_u) = \sum_{u=1}^U (\boldsymbol{w}_u - c_u \boldsymbol{s}_{u,2}) = \boldsymbol{w} - \sum_{u=1}^U c_u \boldsymbol{s}_{u,2}. \tag{2}$$

By the setting $46 \cdot 2^{d-1} \le \gamma$ in Table 1, we have

$$\left|\sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{L}}\right|_\infty \le \sum_{u=1}^U \left|c_u \boldsymbol{t}_{u,\mathsf{L}}\right|_\infty \le \sum_{u=1}^U 46 \cdot 2^{d-1} \le \sum_{u=1}^U \gamma \le U\gamma.$$

By using the settings $q > 4U\gamma$ and $q \equiv 1 \pmod{2U\gamma}$ in Table 1, it follows from Lemma 7 and Eq. (2) that

$$\begin{aligned}
&\mathsf{UseHint}_q\left(\boldsymbol{h}, \boldsymbol{A}\boldsymbol{z} - 2^d \sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{H}}, 2U\gamma\right) \\
&= \mathsf{UseHint}_q\left(\mathsf{MakeHint}_q\left(-\sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{L}}, \boldsymbol{A}\boldsymbol{z} - 2^d \sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{H}}, 2U\gamma\right), \boldsymbol{A}\boldsymbol{z} - 2^d \sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{H}}, 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{z} - 2^d \sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{H}} - \sum_{u=1}^U c_u \boldsymbol{t}_{u,\mathsf{L}}, 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{z} - \sum_{u=1}^U c_u \left(2^d \cdot \boldsymbol{t}_{u,\mathsf{H}} + \boldsymbol{t}_{u,\mathsf{L}}\right), 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{z} - \sum_{u=1}^U c_u \boldsymbol{t}_u, 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{w} - \sum_{u=1}^U c_u \boldsymbol{s}_{u,2}, 2U\gamma\right).
\end{aligned} \tag{3}$$

Since the setting $|c\boldsymbol{s}|_\infty < \beta$ implies that $\left|\sum_{u=1}^U c_u \boldsymbol{s}_{u,2}\right|_\infty < U\beta$ and the assumption that the protocol does not restart at (4.2) of the signing process, Lemma 8 implies that

$$\mathsf{HighBits}_q\left(\boldsymbol{w} - \sum_{u=1}^U c_u \boldsymbol{s}_{u,2}, 2U\gamma\right) = \mathsf{HighBits}_q(\boldsymbol{w}, 2U\gamma) = \boldsymbol{w}_\mathsf{H}. \tag{4}$$

Putting together Eqs. (3) and (4), we can show that the verifying condition (b) of Ver holds.

Expected Time of Iteration In order to evaluate the expected time E of iteration on Sig, we introduce the following lemma.

Lemma 9 ([KLS18, Lemma 4.3]). *Let $c \in \mathcal{CH}$ and $\boldsymbol{s} \in S_\eta^\ell$. Assume that $|c\boldsymbol{s}|_\infty < \beta$. Then,*

$$\textit{For any } \boldsymbol{z} \in S^\ell_{\gamma'-\beta-1}, \Pr_{\boldsymbol{y} \in_\mathrm{U} S^\ell_{\gamma'-1}}[\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s}] = \frac{1}{\left|S^\ell_{\gamma'-1}\right|}, \tag{5}$$

$$\Pr_{\boldsymbol{y} \in_\mathrm{U} S^\ell_{\gamma'-1}}\left[\boldsymbol{y} + c\boldsymbol{s} \in S^\ell_{\gamma'-\beta-1}\right] = \frac{\left|S^\ell_{\gamma'-\beta-1}\right|}{\left|S^\ell_{\gamma'-1}\right|}. \tag{6}$$

Putting together Eqs. (5) and (6), the following holds.

Lemma 10. *Let $c \in \mathcal{CH}$ and $\boldsymbol{s} \in S^{\ell}_{\eta}$. Assume that $|c\boldsymbol{s}|_{\infty} < \beta$. For any $\boldsymbol{z} \in S^{\ell}_{\gamma'-\beta-1}$, we have*

$$\Pr_{\boldsymbol{y} \in_{\mathrm{U}} S^{\ell}_{\gamma'-1}} \left[\boldsymbol{z} = \boldsymbol{y} + c\boldsymbol{s} \mid \boldsymbol{y} + c\boldsymbol{s} \in S^{\ell}_{\gamma'-\beta-1} \right] = \frac{1}{\left| S^{\ell}_{\gamma'-\beta-1} \right|}. \tag{7}$$

We first evaluate the probability that the protocol does not restart at (3.6) for each signer. Since $\beta \ll \gamma'$, Eq. (6) implies that

$$\begin{aligned} \Pr_{\boldsymbol{y} \in_{\mathrm{U}} S^{\ell}_{\gamma'-1}} \left[\boldsymbol{y} + c\boldsymbol{s} \in S^{\ell}_{\gamma'-\beta-1} \right] &= \frac{\left| S^{\ell}_{\gamma'-\beta-1} \right|}{\left| S^{\ell}_{\gamma'-1} \right|} \\ &= \left(\frac{2(\gamma'-\beta-1)+1}{2(\gamma'-1)+1} \right)^{n\ell} \\ &= \left(1 - \frac{\beta}{\gamma'-1/2} \right)^{n\ell} \\ &\approx e^{-n\ell\beta/\gamma'}. \end{aligned} \tag{8}$$

We next evaluate the probability that the protocol does not restart at (4.2). It follows from Lemma 10 that $\boldsymbol{z}_u$ is uniformly distributed over $S^{\ell}_{\gamma'-\beta-1}$ provided that $\boldsymbol{z}_u \in S^{\ell}_{\gamma'-\beta-1}$. In a smilar manner to [KLS18, Lemma 4.4], this implies that for any B, $\boldsymbol{A}\boldsymbol{z} - B \bmod {}^{\pm} 2U\gamma$ can be regarded as it is uniformly distributed over $S^{k}_{U\gamma-1}$. Since $\beta \ll \gamma$, we have

$$\begin{aligned} \Pr\left[\left\| \mathsf{LowBits}_q(\boldsymbol{A}\boldsymbol{z} - \sum_{u=1}^{U} c_u \boldsymbol{t}_u, 2U\gamma) \right\|_{\infty} < U(\gamma-\beta) \right] &= \frac{\left| S^{k}_{U(\gamma-\beta)-1} \right|}{\left| S^{k}_{U\gamma-1} \right|} \\ &= \left(1 - \frac{\beta}{\gamma - 1/2U} \right)^{nk} \\ &\approx e^{-nk\beta/\gamma}. \end{aligned} \tag{9}$$

Thus the probability that the signing protocol does not restart at the processes (3.6) and (4.2) is evaluated by $\left(e^{-n\ell\beta/\gamma'}\right)^{U} e^{-nk\beta/\gamma} = e^{-n\beta(\ell U/\gamma' + k/\gamma)}$. And then, we can set the parameter E, which is the expected time of iteration, to $e^{n\beta(\ell U/\gamma' + k/\gamma)}$.

4 Security

In this section, we show the security proof of the proposed multisignature scheme. Before that we prepare the auxiliary lemma.

Lemma 11 ([KLS18, Lemma 4.6]). *Assume that $q \equiv 5 \pmod 8$. Let α_1, α_2 be positive integers less than $\sqrt{q/2}$, and let C be a set of elements in $R \setminus \{0\}$ with coefficients less than $\sqrt{q/2}$. Let $2^d < 2\alpha_1$. Then,*

$$\Pr_{\boldsymbol{A} \in_{\mathrm{U}} R_q^{k \times \ell}, \boldsymbol{t} \in_{\mathrm{U}} R_q^{k}} \left[\exists (\boldsymbol{z}_1, \boldsymbol{z}_2, c) \in S^{\ell}_{\alpha_1} \times S^{k}_{\alpha_2} \times C \text{ s.t. } \boldsymbol{A}\boldsymbol{z}_1 + \boldsymbol{z}_2 = c\boldsymbol{t}_1 \cdot 2^d \right] \leq 2|C| \left(\frac{(2\alpha_1+1)^{\ell}(2\alpha_2+1)^{k}}{q^{k}} \right)^{n},$$

where $\boldsymbol{t}_1 = \mathsf{Power2Round}_q(\boldsymbol{t}, d)$.

We now state our theorem.

Theorem 12. *Let $(q, n, k, \ell, d, \gamma, \gamma', \eta, \beta, \theta)$ be parameters as in Table 1, and let OW be a one-way injective function such that any QPT adversary breaks the one-wayness with probability at most ϵ_{OW}. Assume that the $\left(T_{\mathrm{MLWE}}, \epsilon_{\mathrm{MLWE}}, \mathsf{Uni}(S_\eta)\right)$-MLWE assumption and $\left(T_{\mathrm{rMLWE}}, \epsilon_{\mathrm{rMLWE}}, \mathsf{Uni}(S_\eta), \gamma', \beta\right)$-rMLWE assumption hold. Then, the proposed multisignature signature scheme is $(T, U, \epsilon, Q_S, Q_0, Q_1, Q_2)$-ppk secure in QROM,*

where

$$T = T_{\mathrm{MLWE}} - O(Q_S EU),\ T = T_{\mathrm{rMLWE}} - O(Q_S EU),$$
$$\epsilon < \epsilon_{\mathrm{MLWE}} + \frac{2\sqrt{Q_1}}{2^{\theta/2}} + \frac{Q_S E(4+\sqrt{2})\sqrt{Q_2 + (Q_S E - 1)U}}{2^{\theta/4}} + Q_S E \epsilon_{\mathrm{rMLWE}}$$
$$+ 8 \cdot \left(\frac{1}{|\mathcal{CH}|} + 2|\mathcal{CH}|^2 \left(\frac{32U^2\gamma'\gamma}{q} \right)^{kn} \right) \cdot (Q_2 + Q_S UE + 1)^2 + \epsilon_{\boldsymbol{w}_{\mathsf{H}}} + \epsilon_{\mathsf{OW}} + \mathrm{negl}.$$

Here, $\epsilon_{\boldsymbol{w}_{\mathsf{H}}}$ denotes the probability that $\boldsymbol{w}_{\mathsf{H}} = \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right)$ over the choices of $\boldsymbol{A} \in_{\mathrm{U}} R_q^{k\times\ell}$ and $\boldsymbol{y} \in_{\mathrm{U}} S^{\ell}_{\gamma'-1}$ for any $\boldsymbol{w}_{\mathsf{H}}$ which would be output by $\mathsf{HighBits}_q(\boldsymbol{w}, 2U\gamma)$ and any fixed set $\{\boldsymbol{w}_u\}_{u=2}^{U}$.

For the probability $\epsilon_{\boldsymbol{w}_{\mathsf{H}}}$, the following lemma holds.

Lemma 13. *It holds that*

$$\epsilon_{\boldsymbol{w}_{\mathsf{H}}} \leq \frac{1}{(2\gamma'-1)^{n\ell}} + \left(\frac{(4U\gamma+1)(4\gamma'+1)}{q} \right)^{nk}.$$

Proof. For fixed $\boldsymbol{w}_{\mathsf{H}}$ and $\boldsymbol{A}$, let $\mathsf{Set}_{\boldsymbol{w}_{\mathsf{H}}}$ denote the set of all vectors $\boldsymbol{y}$ such that $\boldsymbol{w}_{\mathsf{H}} = \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right)$. Assume that $\left|\mathsf{Set}_{\boldsymbol{w}_{\mathsf{H}}}\right| = 1$. Then, we can obtain the probability that $\boldsymbol{w}_{\mathsf{H}} = \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right)$ is evaluated by $1/\left|S^{\ell}_{\gamma'-1}\right| = (2\gamma'-1)^{-n\ell}$, where the probability is taken over the choice of $\boldsymbol{y} \in_{\mathrm{U}} S^{\ell}_{\gamma'-1}$.

We now consider the probability ξ that $\left|\mathsf{Set}_{\boldsymbol{w}_{\mathsf{H}}}\right| \geq 2$ holds for any $\boldsymbol{w}_{\mathsf{H}}$, where the probability is taken over the choice of $\boldsymbol{A} \in_{\mathrm{U}} R_q^{k\times\ell}$.

Assume that there exist two vectors $\boldsymbol{y}$ and $\boldsymbol{y}'$ which satisfy the followings:

- $\boldsymbol{y} \neq \boldsymbol{y}'$; and
- $\mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right) = \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y}' + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right)$.

We let

$$(\boldsymbol{w}_{\mathsf{H}}, \boldsymbol{w}_{\mathsf{L}}) = \mathsf{Decompose}_q\left(\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right);\ \text{and}$$
$$\left(\boldsymbol{w}'_{\mathsf{H}}, \boldsymbol{w}'_{\mathsf{L}}\right) = \mathsf{Decompose}_q\left(\boldsymbol{A}\boldsymbol{y}' + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right).$$

If $\mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right) = \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y}' + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right)$, then it holds that $\boldsymbol{w}_{\mathsf{H}} = \boldsymbol{w}'_{\mathsf{H}}$. Therefore, we have

$$\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u = 2U\gamma \cdot \boldsymbol{w}_{\mathsf{H}} + \boldsymbol{w}_{\mathsf{L}},\ |\boldsymbol{w}_{\mathsf{L}}|_\infty \leq U\gamma;\ \text{and}$$
$$\boldsymbol{A}\boldsymbol{y}' + \sum_{u=2}^{U} \boldsymbol{w}_u = 2U\gamma \cdot \boldsymbol{w}_{\mathsf{H}} + \boldsymbol{w}'_{\mathsf{L}},\ \left|\boldsymbol{w}'_{\mathsf{L}}\right|_\infty \leq U\gamma.$$

By letting $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{y}'$ and $\tilde{\boldsymbol{w}} = \boldsymbol{w}_{\mathsf{L}} - \boldsymbol{w}'_{\mathsf{L}}$, we have $\boldsymbol{A}\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{w}} = 0$, $|\tilde{\boldsymbol{y}}|_\infty \leq 2\gamma'$ and $|\tilde{\boldsymbol{w}}|_\infty \leq 2U\gamma$. By the settings of γ in Table 1, $2U\gamma < \sqrt{q/2}$ follows. Then we have

$$\xi \leq \left(\frac{(4U\gamma+1)(4\gamma'+1)}{q} \right)^{nk},$$

by using a similar manner to [KLS18, Lemma 4.7]. Let $\mathsf{W}_{\boldsymbol{w}_{\mathsf{H}}}$ be the event that $\boldsymbol{w}_{\mathsf{H}} = \mathsf{HighBits}_q\left(\sum_{u=1}^{U} \boldsymbol{w}_u, 2U\gamma\right)$ holds for any $\boldsymbol{w}_{\mathsf{H}}$. Then we have

$$\begin{aligned}
\epsilon_{\boldsymbol{w}_{\mathsf{H}}} &= \Pr\left[\mathsf{W}_{\boldsymbol{w}_{\mathsf{H}}}\right] \\
&\leq \Pr\left[\mathsf{W}_{\boldsymbol{w}_{\mathsf{H}}} \wedge \left|\mathsf{Set}_{\boldsymbol{w}_{\mathsf{H}}}\right| = 1\right] + \Pr\left[\mathsf{W}_{\boldsymbol{w}_{\mathsf{H}}} \wedge \left|\mathsf{Set}_{\boldsymbol{w}_{\mathsf{H}}}\right| \geq 2\right] \\
&\leq \Pr\left[\mathsf{W}_{\boldsymbol{w}_{\mathsf{H}}} \mid \left|\mathsf{Set}_{\boldsymbol{w}_{\mathsf{H}}}\right| = 1\right] + \Pr\left[\left|\mathsf{Set}_{\boldsymbol{w}_{\mathsf{H}}}\right| \geq 2\right] \\
&\leq \frac{1}{(2\gamma'-1)^{n\ell}} + \xi \\
&\leq \frac{1}{(2\gamma'-1)^{n\ell}} + \left(\frac{(4U\gamma+1)(4\gamma'+1)}{q} \right)^{nk}.
\end{aligned}$$

Init C proceeds as follows:

(1) $H_0 \in_{\mathsf{U}} \mathsf{Fun}(\mathfrak{D}_0, \mathfrak{R}_0)$;
(2) $H_2 \in_{\mathsf{U}} \mathsf{Fun}(\mathfrak{D}_2, \mathcal{CH})$.
(3) $\varrho \leftarrow \mathsf{Setup}(1^\kappa)$.
(4) $(\mathbf{sk}^*, \mathbf{pk}^*) \leftarrow \mathsf{KGen}(\varrho)$, where $\mathbf{sk}^* = \left(\boldsymbol{s}_1^*, \boldsymbol{s}_2^*\right)$ and $\mathbf{pk}^* = \left(\boldsymbol{t}_{\mathsf{H}}^*, \boldsymbol{t}_{\mathsf{L}}^*\right)$.
(5) send $(\varrho, \mathbf{pk}^*)$ to $\mathcal{F}$.

H_0 **oracle** On $|\boldsymbol{w}\rangle|\psi\rangle$, reply $|\boldsymbol{w}\rangle|\psi \oplus H_0(\boldsymbol{w})\rangle$.

H_1 **oracle** On $|\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK}, \mu\rangle|\tau\rangle$, reply the hash value by using a database D.

H_2 **oracle** On $|(\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, P, \mu, \tau)\rangle|\psi\rangle$, reply $|(\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, P, \mu, \tau)\rangle|\psi \oplus H_2((\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, P, \mu, \tau))\rangle$.

Sign When $\mathcal{F}$ makes an i-th query $(\mathsf{PK}^{(i)}, \mu^{(i)})$ of a set $\mathsf{PK}^{(i)} = \{\mathbf{pk}_u^{(i)}\}_{u=1}^U$ and a message $\mu^{(i)}$ such that $\mathbf{pk}_1^{(i)} = \mathbf{pk}^*$, C finally returns $\sigma^{(i)} = (\boldsymbol{w}_{\mathsf{H}}^{(i)}, \boldsymbol{z}^{(i)}, \boldsymbol{h}^{(i)})$ after running the following protocol, where $\mathbf{pk}_u^{(i)} = (\boldsymbol{t}_{u,\mathsf{H}}^{(i)}, \boldsymbol{t}_{u,\mathsf{L}}^{(i)})$ for each $2 \le u \le U$:

1st stage proceed as follows, and then send $(\boldsymbol{g}_1^{(i)}, \boldsymbol{p}_1^{(i)})$ to $\mathcal{F}$:

(1.1) $\boldsymbol{A} = \mathsf{Sam}(\varrho)$; $\boldsymbol{y}_1^{(i)} \in_{\mathsf{U}} S_{\gamma'-1}^{\ell}$; $\boldsymbol{w}_1^{(i)} = \boldsymbol{A}\boldsymbol{y}_1^{(i)}$.
(1.2) $\boldsymbol{g}_1^{(i)} = H_0(\boldsymbol{w}_1^{(i)})$; $\boldsymbol{p}_1^{(i)} = \mathsf{OW}_{\boldsymbol{g}_1^{(i)}}(\boldsymbol{w}_1^{(i)})$.

2nd stage After receiving $\{(\boldsymbol{g}_u^{(i)}, \boldsymbol{p}_u^{(i)})\}_{u=2}^U$, send $\boldsymbol{w}_1^{(i)}$ to $\mathcal{F}$.

3rd stage After receiving $\{\boldsymbol{w}_u^{(i)}\}_{u=2}^U$, proceed as follows, and then send $\boldsymbol{z}_1^{(i)}$ to $\mathcal{F}$:

(3.1) abort if $\neg(\boldsymbol{g}_u^{(i)} = H_0(\boldsymbol{w}_u^{(i)}) \wedge \boldsymbol{p}_u^{(i)} = \mathsf{OW}_{\boldsymbol{g}_u^{(i)}}(\boldsymbol{w}_u^{(i)}))$ for some $2 \le u \le U$.
(3.2) $\boldsymbol{w}^{(i)} = \sum_{u=1}^U \boldsymbol{w}_u^{(i)}$; $\boldsymbol{w}_{\mathsf{H}}^{(i)} = \mathsf{HighBits}_q(\boldsymbol{w}^{(i)}, 2U\gamma)$.
(3.3) $\tau_1^{(i)} = H_1(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)})$.
(3.4) $c_1^{(i)} = H_2(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)} \setminus \{\mathbf{pk}^*\}, \mu^{(i)}, \tau_1^{(i)})$.
(3.5) $\boldsymbol{z}_1^{(i)} = \boldsymbol{y}_1^{(i)} + c_1^{(i)}\boldsymbol{s}_1^*$.
(3.6) restart if $\neg(|\boldsymbol{z}_1^{(i)}|_\infty < \gamma' - \beta)$.

4th stage After receiving $\{\boldsymbol{z}_u^{(i)}\}_{u=2}^U$, proceed as follows:

(4.1) $\boldsymbol{z}^{(i)} = \sum_{u=1}^U \boldsymbol{z}_u^{(i)}$.
(4.2) restart if $\neg(|\mathsf{LowBits}_q(\boldsymbol{A}\boldsymbol{z}^{(i)} - \sum_{u=1}^U c_u^{(i)}\boldsymbol{t}_u^{(i)}, 2U\gamma)|_\infty < U(\gamma - \beta))$.
(4.3) $\boldsymbol{h}^{(i)} = \mathsf{MakeHint}_q(-\sum_{u=1}^U c_u^{(i)}\boldsymbol{t}_{u,\mathsf{L}}^{(i)}, \boldsymbol{A}\boldsymbol{z}^{(i)} - 2^d \sum_{u=1}^U c_u^{(i)}\boldsymbol{t}_{u,\mathsf{H}}^{(i)}, 2U\gamma)$.

Challenge Given $(\mathsf{PK}^*, \mu^*, \sigma^*)$ by $\mathcal{F}$, C returns 1 if the followings hold, where $\mathsf{PK}^* = \{\mathbf{pk}_u^*\}_{u=1}^U$, $\sigma^* = \left(\boldsymbol{w}_{\mathsf{H}}^*, \boldsymbol{z}^*, \boldsymbol{h}^*\right)$, and for each $1 \le u \le U$, $\mathbf{pk}_u^* = \left(\boldsymbol{t}_{u,\mathsf{H}}^*, \boldsymbol{t}_{u,\mathsf{L}}^*\right)$, $\tau_u^* = H_1\left(\mathbf{pk}_u^*, \boldsymbol{w}_{\mathsf{H}}^*, \mathsf{PK}^*, \mu^*\right)$, and $c_u^* = H_2\left(\mathbf{pk}_u^*, \boldsymbol{w}_{\mathsf{H}}^*, \mathsf{PK}^* \setminus \{\mathbf{pk}_u^*\}, \mu^*, \tau_u^*\right)$:

- $\mathbf{pk}^* \in \mathsf{PK}^*$,
- (μ^*, PK^*) does not queried,
- σ^* is valid, i.e.
  (a) $|\boldsymbol{z}^*|_\infty < U(\gamma' - \beta)$, and
  (b) $\boldsymbol{w}_{\mathsf{H}}^* = \mathsf{UseHint}_q(\boldsymbol{h}^*, \boldsymbol{x}^*, 2U\gamma)$, where $\boldsymbol{x}^* = \boldsymbol{A}\boldsymbol{z}^* - 2^d \sum_{u=1}^U c_u^*\boldsymbol{t}_{u,\mathsf{H}}^*$.

Fig. 6. $\Im_0$

□

Proof (Theorem 12). The theorem is proven by the hybrid argument. Let $\mathcal{F}$ be a forger against the proposed multisignature scheme. We denote by Win_k the event that $\mathcal{F}$ wins $\Im_k$ for each $0 \le k \le 6$ except $k = 5$, and $\mathsf{Win}_{5,t}$ the event that $\mathcal{F}$ wins $\Im_{5,t}$ for each $0 \le t \le Q_s E$.

$\Im_0$ *(ppk game with simulating the random oracles)* This game coincides with the ppk game of the proposed multisignature scheme as in Fig. 6. Here, the hash functions H_0 and H_2 are simulated by random functions, whereas H_1 is done by the almost compressed Fourier oracle. As mentioned in Subsection 2.2, the random oracles are perfectly simulated by using random functions H_0 and H_2. Since H_1 is simulated by the almost compressed random oracle with the database D, there is a possibility that some tuple $(\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK}, \mu)$ is not contained in D, but $H_1(\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK}, \mu) = \tau$ for some τ in the case where H_1 is a perfectly random function. It follows from Lemma 4 and the $(T, U, \epsilon, Q_S, Q_0, Q_1, Q_2)$-ppk security of the proposed multisignature scheme that $\sqrt{\epsilon} \le \sqrt{\Pr[\mathsf{Win}_0]} + \sqrt{Q_1/2^\theta}$. Since $\epsilon \le 1$, we have

$$\Pr[\mathsf{Win}_0] \ge \left(\sqrt{\epsilon} - \sqrt{\frac{Q_1}{2^\theta}}\right)^2 = \epsilon + \frac{Q_1}{2^\theta} - 2\sqrt{\frac{Q_1}{2^\theta} \cdot \epsilon} \ge \epsilon - 2\sqrt{\frac{Q_1}{2^\theta}}. \tag{10}$$

$\Im_1$ At (1) of **Init** phase, C chooses H_0 uniformly at random from a set of all polynomials of degree $2(Q_0 + Q_S UE)$, instead of $H_0 \in_{\mathsf{U}} \mathsf{Fun}(\mathfrak{D}_0, \mathfrak{R}_0)$. Since $\mathcal{F}$ makes at most Q_0 queries, and C computes at most UE hash values of

2nd stage After receiving $\{(\boldsymbol{g}_u^{(i)}, \boldsymbol{p}_u^{(i)})\}_{u=2}^{U}$, proceed as follows, and then send $\boldsymbol{w}_1^{(i)}$ to $\mathcal{F}$:

(2.1) for each $2 \le u \le U$,

(2.1a) compute the roots $\{\boldsymbol{r}_{u,j}^{(i)}\}_j$ of $H_0 - \boldsymbol{g}_u^{(i)}$,

(2.1b) find $\boldsymbol{r}_u^{(i)} \in \{\boldsymbol{r}_{u,j}^{(i)}\}_j$ such that $\boldsymbol{p}_u^{(i)} = \mathsf{OW}_{\boldsymbol{g}_u^{(i)}}(\boldsymbol{r}_u^{(i)})$, and

(2.1c) abort if there is no such $\boldsymbol{r}_u^{(i)}$.

(2.2) $\boldsymbol{w}^{(i)} = \boldsymbol{w}_1^{(i)} + \sum_{u=2}^{U} \boldsymbol{r}_u^{(i)}$.

(2.3) $\boldsymbol{w}_{\mathsf{H}}^{(i)} = \mathsf{HighBits}_q(\boldsymbol{w}^{(i)}, 2U\gamma)$.

(2.4) $\tau_1^{(i)} = H_1(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)})$.

(2.5) $c_1^{(i)} = H_2(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)} \setminus \{\mathbf{pk}^*\}, \mu^{(i)}, \tau_1^{(i)})$.

3rd stage After receiving $\{\boldsymbol{w}_u^{(i)}\}_{u=2}^{U}$, proceed as follows, and then send $\boldsymbol{z}_1^{(i)}$ to $\mathcal{F}$:

(3.1) abort if $\neg(\boldsymbol{r}_u^{(i)} = \boldsymbol{w}_u^{(i)})$ for some $2 \le u \le U$.

(3.2) $\boldsymbol{z}_1^{(i)} = \boldsymbol{y}_1^{(i)} + c_1^{(i)} \boldsymbol{s}_1^*$.

(3.3) restart if $\neg(\left|\boldsymbol{z}_1^{(i)}\right|_\infty < \gamma' - \beta)$.

Fig. 7. Description of $\Im_2$ changed from $\Im_1$

H_0 for each signing oracle query, it follows from Lemma 3 that

$$\Pr[\mathsf{Win}_1] = \Pr[\mathsf{Win}_0]. \tag{11}$$

$\Im_2$ In this game, $c_1^{(i)}$ is computed at **2nd stage** in **Sign** phase, instead of **3rd stage** in $\Im_1$. In order to accomplish this change, $\boldsymbol{w}_{\mathsf{H}}^{(i)}$ is also required to be computed at **2nd stage**. Hence, **2nd stage** and **3rd stage** in **Sign** phase are replaced with Fig. 7.

Before showing the relationship between Win_1 and Win_2, we note the process (2.1a). As mentioned in Subsection 2.2, the roots of the polynomials of degree $2(Q_0 + Q_S UE)$ can be computed in polynomial time, and hence the process (2.1a) can be done in polynomial time.

Assume that C does not abort at (2.1c) on $\Im_2$. In this case, the injectivity of OW implies that the condition $\boldsymbol{r}_u^{(i)} = \boldsymbol{w}_u^{(i)}$ checked in (3.1) on $\Im_2$ is equivalent to the condition $\boldsymbol{g}_u^{(i)} = H_0(\boldsymbol{w}_u^{(i)}) \wedge \boldsymbol{p}_u^{(i)} = \mathsf{OW}_{\boldsymbol{g}_u^{(i)}}(\boldsymbol{w}_u^{(i)})$ checked in (3.1) on $\Im_1$. Thus we can consider that C also does not abort in (3.1) on $\Im_2$ under the assumption that C does not abort at (2.1c).

We now evaluate the abort probability in (2.1c) on $\Im_2$. To proceed to **Sign** phase, $\mathcal{F}$ must send a set $\left\{\left(\boldsymbol{g}_u^{(i)}, \boldsymbol{p}_u^{(i)}\right)\right\}_{u=2}^{U}$ to C, such that there exist $\boldsymbol{w}_u^{(i)}$'s which satisfy $\boldsymbol{g}_u^{(i)} = H_0\left(\boldsymbol{w}_u^{(i)}\right)$ and $\boldsymbol{p}_u^{(i)} = \mathsf{OW}_{\boldsymbol{g}_u^{(i)}}\left(\boldsymbol{w}_u^{(i)}\right)$. It follows from the injectivity of OW that there is at most one $\boldsymbol{w}_u^{(i)}$ for each i and u. Therefore there must exist $\boldsymbol{r}^{(i)}$ which satisfies $\boldsymbol{p}_u^{(i)} = \mathsf{OW}_{\boldsymbol{g}_u^{(i)}}\left(\boldsymbol{r}_u^{(i)}\right)$, and hence we can consider that C never abort in (2.1c).

We have

$$\Pr[\mathsf{Win}_2] = \Pr[\mathsf{Win}_1], \tag{12}$$

$\Im_3$ At (2.4) during each execution of **Sign** in this game, C is changed to abort if D have already contained the tuple $\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)}\right)$. Otherwise, $\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)}\right)$ is regarded as a special point on the almost compressed Fourier oracle, and then C appends $\sum_{\tau_1^{(i)}} \left|\tau_1^{(i)}\right\rangle$ into the database D as the hash value of $\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)}\right)$.

In order to evaluate the abort probability due to the change in $\Im_3$, we focus on the probability that $\mathcal{F}$ finds $\boldsymbol{w}_{\mathsf{H}}^{(i)}$ before querying $\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)}\right)$. There are two ways that $\mathcal{F}$ finds $\boldsymbol{w}_{\mathsf{H}}^{(i)}$.

The first one is to find it from $\boldsymbol{g}_1^{(i)}$ and $\boldsymbol{p}_1^{(i)}$, because $\boldsymbol{g}_1^{(i)}$ and $\boldsymbol{p}_1^{(i)}$ are computed from $\boldsymbol{w}_1^{(i)}$. Since H_0 is a polynomial of degree $2(Q_0 + Q_S UE)$ and its coefficients are hidden to $\mathcal{F}$, it is difficult for $\mathcal{F}$ to determine $\boldsymbol{w}_1^{(i)}$ from H_0 [TU16]. Moreover, it follows from the one-wayness of OW that such a vector cannot be determined from OW. Therefore, the abort probability due to these reasons is at most $\epsilon_{\mathsf{OW}} + \mathrm{negl}$.

The second one is that $\boldsymbol{w}_{\mathsf{H}}^{(i)}$ has been found at the process (2.3) for a randomly chosen $\boldsymbol{y}_1^{(i)} \in_{\mathrm{U}} S_{\gamma'-1}^{\ell}$ and any $\left\{\boldsymbol{w}_u^{(i)}\right\}_{u=2}^{U}$ given from $\mathcal{F}$. This can be evaluated by the probability $\epsilon_{\boldsymbol{w}_{\mathsf{H}}}$ that $\boldsymbol{w}_{\mathsf{H}} = \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{y} + \sum_{u=2}^{U} \boldsymbol{w}_u, 2U\gamma\right)$ holds for any $\boldsymbol{w}_{\mathsf{H}}$ which would be output by $\mathsf{HighBits}_q(\boldsymbol{w}, 2U\gamma)$ and any fixed set $\{\boldsymbol{w}_u\}_{u=2}^{U}$, where the probability is taken over the choices of $\boldsymbol{A} \in_{\mathrm{U}} R_q^{k\times\ell}$ and $\boldsymbol{y} \in_{\mathrm{U}} S_{\gamma'-1}^{\ell}$.

Then we have

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \le \epsilon_{\mathsf{OW}} + \epsilon_{\boldsymbol{w}_{\mathsf{H}}} + \mathrm{negl}, \tag{13}$$

$\mathcal{A}_0(1^\kappa)$ executes $\Im_{5,t}$ by playing the role of C and interacting with $\mathcal{F}$ just until the $(t+1)$-th execution of (2.1c), and then returns $\mathrm{st} = \left(H_0, \mathbf{sk}^*, \mathbf{pk}^*, \mathsf{PK}^{(i)}, \mu^{(i)}, \boldsymbol{y}_1^{(i)}, \boldsymbol{w}_1^{(i)}, \left\{\boldsymbol{g}_u^{(i)}\right\}_{u=1}^U, \left\{\boldsymbol{p}_u^{(i)}\right\}_{u=1}^U, \left\{\boldsymbol{r}_u^{(i)}\right\}_{u=2}^U\right)$.

$\mathcal{A}_C(\mathrm{st})$ executes (2.2), (2.3) and (2.4) on the $(t+1)$-th execution, and then outputs $\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)} \setminus \{\mathbf{pk}^*\}, \mu^{(i)}, \tau_1^{(i)}\right)$ and st.

$\mathcal{A}_2(c_1^{(i)}, \mathrm{st})$ executes $\Im_{5,t}$ until the end, and then returns C's final output.

Fig. 8. Adversary $\mathcal{A}$ of the reprogram problem from $\Im_{5,t}$

1st stage proceed as follows, and then send $(\boldsymbol{g}_1^{(i)}, \boldsymbol{p}_1^{(i)})$ to $\mathcal{F}$:

(1.1) $\boldsymbol{A} = \mathsf{Sam}(\varrho)$.
(1.2) $c_1^{(i)} \in_{\mathrm{U}} \mathcal{CH}$.
(1.3) $\boldsymbol{z}_1^{(i)} \in_{\mathrm{U}} S^{\ell}_{\gamma'-\beta-1}$.
(1.4) $\nu \in_{B_{\omega_1}} \{0,1\}$, where $\omega_1 = 1 - e^{-n\ell\beta/\gamma'}$.
(1.5) $\boldsymbol{w}_1^{(i)} = \boldsymbol{A}\boldsymbol{z}_1^{(i)} - c_1^{(i)}(\boldsymbol{t}^* - \boldsymbol{s}_2^*)$ if $\nu = 0$, or $\boldsymbol{w}_1^{(i)} \in_{\mathrm{U}} R_q$ otherwise.
(1.6) $\boldsymbol{g}_1^{(i)} = H_0(\boldsymbol{w}_1^{(i)})$; $\boldsymbol{p}_1^{(i)} = \mathsf{OW}_{\boldsymbol{g}_1^{(i)}}(\boldsymbol{w}_1^{(i)})$.

2nd stage After receiving $\{(\boldsymbol{g}_u^{(i)}, \boldsymbol{p}_u^{(i)})\}_{u=2}^U$, proceeds as follows, and then send $\boldsymbol{w}_1^{(i)}$ to $\mathcal{F}$:

(2.1) for each $2 \le u \le U$,
  (2.1a) compute the roots $\{\boldsymbol{r}_{u,j}^{(i)}\}_j$ of $H_0 - \boldsymbol{g}_u^{(i)}$,
  (2.1b) find $\boldsymbol{r}_u^{(i)} \in \{\boldsymbol{r}_{u,j}^{(i)}\}_j$ such that $\boldsymbol{p}_u^{(i)} = \mathsf{OW}_{\boldsymbol{g}_u^{(i)}}(\boldsymbol{r}_u^{(i)})$,
  (2.1c) abort if there is no such $\boldsymbol{r}_u^{(i)}$.
(2.2) $\boldsymbol{w}^{(i)} = \boldsymbol{w}_1^{(i)} + \sum_{u=2}^U \boldsymbol{r}_u^{(i)}$.
(2.3) $\boldsymbol{w}_{\mathsf{H}}^{(i)} = \mathsf{HighBits}_q(\boldsymbol{w}^{(i)}, 2U\gamma)$.
(2.4) abort if D has already contained $(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)})$.
(2.5) $\tau_1^{(i)} \in_{\mathrm{U}} \{0,1\}^\theta$; append $((\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)}), \tau_1^{(i)})$ to D.
(2.6) set $H_2(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)} \setminus \{\mathbf{pk}^*\}, \mu^{(i)}, \tau_1^{(i)}) = c_1^{(i)}$.

3rd stage After receiving $\{\boldsymbol{w}_u^{(i)}\}_{u=2}^U$, proceed as follows, and then send $\boldsymbol{z}_1^{(i)}$ to $\mathcal{F}$:

(3.1) abort if $\neg(\boldsymbol{r}_u^{(i)} = \boldsymbol{w}_u^{(i)})$ for some $2 \le u \le U$.
(3.2) restart if $\nu = 1$.

Fig. 9. Description of $\Im_6$ changed from $\Im_{5,Q_S E}$

$\Im_4$ At (2.4) during each execution of **Sign** in this game, C is changed to appends $\left|\tau_1^{(i)}\right\rangle$ for $\tau_1^{(i)} \in_{\mathrm{U}} \{0,1\}^\theta$ into the database as the hash value of $\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)}, \mu^{(i)}\right)$, instead of $\sum_{\tau_1^{(i)}} \left|\tau_1^{(i)}\right\rangle$ in $\Im_3$. From Lemma 5, this change does not affect the probability of the measurement of D. Then we have

$$\Pr[\mathsf{Win}_4] = \Pr[\mathsf{Win}_3]. \tag{14}$$

$\Im_{5,t}$ In this game, we consider the process (2.5) of **Sign**. **Sign** phase is queried at most Q_s times by $\mathcal{F}$. Moreover, for each query by $\mathcal{F}$, the process (2.5) is executed at most E times. Then the total number of the execution of (2.5) during the game is at most $Q_s E$. Fix an index $0 \le t \le Q_s E$. In $\Im_{5,t}$, on the k-th execution of (2.5) for $1 \le k \le t$, C sets the hash value $c_1^{(i)}$ as follows: chooses $c_1^{(i)} \in_{\mathrm{U}} \mathcal{CH}$ and set $H_2\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^{(i)}, \mathsf{PK}^{(i)} \setminus \{\mathbf{pk}^*\}, \mu^{(i)}, \tau_1^{(i)}\right) = c_1^{(i)}$.

When $t = 0$, the replacement on $c_1^{(i)}$ does not happen, and hence we have

$$\Pr[\mathsf{Win}_{5,0}] = \Pr[\mathsf{Win}_4]. \tag{15}$$

Otherwise, we employ Lemma 2 to evaluate the difference between the probability of $\mathsf{Win}_{5,t}$ and that of $\mathsf{Win}_{5,t+1}$. We construct an algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_C, \mathcal{A}_2)$ depicted in Fig. 8. Observe that $\mathbf{Exp}^{\mathsf{Reprog}}_{\mathfrak{D}_2,\mathcal{CH},1,\mathcal{A}}$ and $\mathbf{Exp}^{\mathsf{Reprog}}_{\mathfrak{D}_2,\mathcal{CH},0,\mathcal{A}}$ coincide with $\Im_{5,t}$ and $\Im_{5,t+1}$, respectively. Since $\mathcal{F}$ queries to H_2 at most Q_2 times and C acted by $\mathcal{A}_0$ queries at most tU times, the number of access H_2 by $\mathcal{A}_0$ is at most $Q_2 + tU$. The uniform choice of $\tau_1^{(i)}$ over $\{0,1\}^\theta$ by $\mathcal{A}_C$ at (2.4) implies that the collision-entropy can be evaluated by θ.

Then, for $0 \le t \le Q_s E - 1$, it follows from Lemma 2 that

$$\left|\Pr[\mathsf{Win}_{5,t+1}] - \Pr[\mathsf{Win}_{5,t}]\right| \le (4 + \sqrt{2})\sqrt{Q_2 + tU}\,2^{-\theta/4}. \tag{16}$$

$\Im_6$ In this game, the way of generating $\left(\boldsymbol{w}_{\mathsf{H}}^{(i)}, c_1^{(i)}, \boldsymbol{z}_1^{(i)}\right)$ is changed. Concretely, **1st stage**, **2nd stage** and **3rd stage** in **Sign** phase are replaced with Fig. 9.

In order to discuss the difference between the probability of Win_{5,Q_sE} and that of Win_6, we estimate the restart probabilities in both games and the difference between the distributions of the communicated components $\left(\boldsymbol{g}_1^{(i)}, \boldsymbol{p}_1^{(i)}, \boldsymbol{w}_1^{(i)}, c_1^{(i)}, \boldsymbol{z}_1^{(i)}, \boldsymbol{z}^{(i)}, \boldsymbol{h}^{(i)}\right)$ in both games. We also check that the change of $\mathfrak{I}_6$ does not affect the validness of simulated signatures concerning verifying conditions when $\nu = 0$ determined at (1.4).

For the restart probability, the challenger C restarts $\mathfrak{I}_6$ if $\nu = 1$ which is determined at (1.4). Namely, the probability that $\nu = 1$ is $1 - e^{-n\ell\beta/\gamma'}$. On the other hand, it follows from Eq. (8) that the restart probability $1 - e^{-n\ell\beta/\gamma'}$ here is the same to the one at (3.3) in $\mathfrak{I}_{5,Q_sE}$ as defined in Fig. 7.

We now fix an execution of **Sign** phase and an iteration during the exection of **Sign** phase. We now evaluate the difference between the distribution of the communicated components $\left(\boldsymbol{g}_1^{(i)}, \boldsymbol{p}_1^{(i)}, \boldsymbol{w}_1^{(i)}, c_1^{(i)}, \boldsymbol{z}_1^{(i)}, \boldsymbol{z}^{(i)}, \boldsymbol{h}^{(i)}\right)$ in $\mathfrak{I}_{5,Q_sE}$ and that of $\mathfrak{I}_6$. Since $\left(\boldsymbol{g}_1^{(i)}, \boldsymbol{p}_1^{(i)}, \boldsymbol{z}^{(i)}, \boldsymbol{h}^{(i)}\right)$ is generated by $\left(\boldsymbol{w}_1^{(i)}, c_1^{(i)}, \boldsymbol{z}_1^{(i)}\right)$ with other components given by $\mathcal{F}$, the difference between the distributions of $\left(\boldsymbol{w}_1^{(i)}, c_1^{(i)}, \boldsymbol{z}_1^{(i)}\right)$ in both games implies those of $\left(\boldsymbol{g}_1^{(i)}, \boldsymbol{p}_1^{(i)}, \boldsymbol{z}^{(i)}, \boldsymbol{h}^{(i)}\right)$. It suffices to estimate the difference between the distributions of $\left(\boldsymbol{w}_1^{(i)}, c_1^{(i)}, \boldsymbol{z}_1^{(i)}\right)$ in both games.

We now assume that the restart does not occur. In both games, the element $c_1^{(i)}$ is chosen uniformly at random from $C\mathcal{H}$. The element $\boldsymbol{z}_1^{(i)}$ is set as $\boldsymbol{y}_1^{(i)} + c_1^{(i)}\boldsymbol{s}_1^*$ in (3.2) on $\mathfrak{I}_{5,Q_sE}$, while it is chosen uniformly at random from $S_{\gamma'-\beta-1}$ in (1.3) on $\mathfrak{I}_6$. We now have $\left\|\boldsymbol{z}_1^{(i)}\right\|_\infty < \gamma' - \beta$, since the restart condition at (3.3) of $\mathfrak{I}_{5,Q_sE}$ does not occur by the assumption. It follows from Lemma 10 that $\boldsymbol{z}_1^{(i)}$ on $\mathfrak{I}_{5,Q_sE}$ is distributed uniformly at random over $S^\ell_{\gamma'-\beta-1}$ under the assumption. $\boldsymbol{w}_1^{(i)}$ of $\mathfrak{I}_{5,Q_sE}$ can be seen as $\boldsymbol{A}\boldsymbol{z}_1^{(i)} - c_1^{(i)}\left(\boldsymbol{t}^* - \boldsymbol{s}_2^*\right)$ with $\boldsymbol{z}_1^{(i)} \in_{\mathrm{U}} S_{\gamma'-\beta-1}$ and $c_1^{(i)} \in_{\mathrm{U}} C\mathcal{H}$. It is the same with (1.5) in $\mathfrak{I}_6$.

Otherwise, when C restarts, the distributions in both games are identical except $\boldsymbol{w}_1^{(i)}$. C of $\mathfrak{I}_{5,Q_sE}$ broadcasts $\boldsymbol{w}_1^{(i)}$ which induces the aborts at (3.3) of $\mathfrak{I}_{5,Q_sE}$. On the other hand, C broadcasts $\boldsymbol{w}_1^{(i)}$ which is uniformly distributed over R_q in this case on $\mathfrak{I}_6$. Namely, the difference between such $\boldsymbol{w}_1^{(i)}$ is bounded by the rMLWE assumption. [3]

We finally estemate that the verifying conditions holds when $\nu = 0$. The process $\boldsymbol{z}_1^{(i)} \in_{\mathrm{U}} S^\ell_{\gamma'-\beta-1}$ of (1.3) and the requirement that $\left\|\boldsymbol{z}_u^{(i)}\right\|_\infty < \gamma' - \beta$ for each $2 \le u \le U$ imply that the verifying condition (a) holds. Since $\left(\boldsymbol{w}_1^{(i)}, c_1^{(i)}, \boldsymbol{z}_1^{(i)}\right)$ in $\mathfrak{I}_6$ satisfies that $\boldsymbol{A}\boldsymbol{z}_1^{(i)} - c_1^{(i)}\boldsymbol{t}^* = \boldsymbol{w}_1^{(i)} - c_1^{(i)}\boldsymbol{s}_2^*$ as in Eq. (1), the verifying condition (b) is always satisfied.

Since $\mathcal{A}$ makes at most Q_S queries to **Sign** phase and the iteration is happened at most E times for each **Sign** phase, we have

$$\left|\Pr\left[\mathsf{Win}_6\right] - \Pr\left[\mathsf{Win}_{5,Q_sE}\right]\right| \le Q_S E \epsilon_{\mathrm{rMLWE}}. \tag{17}$$

$\mathfrak{I}_7$ In this game, (1.5) of **1st stage** in **Sign** phase is replaced with $\boldsymbol{w}_1^{(i)} = \boldsymbol{A}\boldsymbol{z}_1^{(i)} - c_1^{(i)}(\boldsymbol{t}^* - \boldsymbol{s}^{(i)})$ where $\boldsymbol{s}^{(i)} \in_{\mathrm{U}} S^k_\eta$ instead of $\boldsymbol{w}_1^{(i)} = \boldsymbol{A}\boldsymbol{z}_1^{(i)} - c_1^{(i)}(\boldsymbol{t}^* - \boldsymbol{s}_2^*)$, if $\nu = 0$.

Since for any $S \in R_q$, the map $x \in R_q$ to $x + S \in R_q$ is bijective, the distributions of $\boldsymbol{w}_1^{(i)}$ in both games are identical for the same choices of $\boldsymbol{A} \in_{\mathrm{U}} R_q^{k\times\ell}$ and $\boldsymbol{z}_1^{(i)} \in_{\mathrm{U}} S^\ell_{\gamma'-\beta-1}$ between the both games.

We now check that the verifying conditions hold after the replacement of $\mathfrak{I}_7$. Since the set from which $\boldsymbol{z}_u^{(i)}$ is chosen does not changed, the verifying condition (a) holds. For the verifying condition (b), we have

$$\begin{aligned}
\boldsymbol{A}\boldsymbol{z}^{(i)} - \sum_{u=1}^{U} c_u^{(i)}\boldsymbol{t}_u^{(i)} &= \boldsymbol{A}\sum_{u=1}^{U}\boldsymbol{z}_u^{(i)} - \sum_{u=1}^{U} c_u^{(i)}\boldsymbol{t}_u^{(i)} \\
&= \sum_{u=1}^{U}\left(\boldsymbol{A}\boldsymbol{z}_u^{(i)} - c_u^{(i)}\boldsymbol{t}_u^{(i)}\right) \\
&= \boldsymbol{w}_1^{(i)} - c_1^{(i)}\boldsymbol{s}^{(i)} + \sum_{u=2}^{U}\left(\boldsymbol{w}_u^{(i)} - c_u^{(i)}\boldsymbol{s}_{u,2}\right) \\
&= \boldsymbol{w}^{(i)} - c_1^{(i)}\boldsymbol{s}^{(i)} - \sum_{u=2}^{U} c_u^{(i)}\boldsymbol{s}_{u,2}.
\end{aligned}$$

[3] [DOTT21] pointed out that $\boldsymbol{w}_1^{(i)}$ in the case where C restarts should be simulated strictly. We employ a method by [BBE+18] to deal with the case.

By $\left\|c_1^{(i)}\boldsymbol{s}^{(i)} + \sum_{u=2}^{U} c_u \boldsymbol{s}_{u,2}\right\|_\infty < U\beta$, as in Eqs. (3) and (4), we have

$$
\begin{aligned}
&\mathsf{UseHint}_q\left(\boldsymbol{h}^{(i)}, \boldsymbol{A}\boldsymbol{z}^{(i)} - 2^d \sum_{u=1}^{U} c_u^{(i)} \boldsymbol{t}_{u,\mathsf{H}}^{(i)}, 2U\gamma\right) \\
&= \mathsf{UseHint}_q\left(\mathsf{MakeHint}_q(-\sum_{u=1}^{U} c_u^{(i)} \boldsymbol{t}_{u,\mathsf{L}}^{(i)}, \boldsymbol{A}\boldsymbol{z}^{(i)} - 2^d \sum_{u=1}^{U} c_u^{(i)} \boldsymbol{t}_{u,\mathsf{H}}^{(i)}, 2U\gamma), \boldsymbol{A}\boldsymbol{z}^{(i)} - 2^d \sum_{u=1}^{U} c_u^{(i)} \boldsymbol{t}_{u,\mathsf{H}}^{(i)}, 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{z}^{(i)} - 2^d \sum_{u=1}^{U} c_u^{(i)} \boldsymbol{t}_{u,\mathsf{H}}^{(i)} - \sum_{u=1}^{U} c_u^{(i)} \boldsymbol{t}_{u,\mathsf{L}}^{(i)}, 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{z}^{(i)} - \sum_{u=1}^{U} c_u^{(i)} \left(2^d \cdot \boldsymbol{t}_{u,\mathsf{H}}^{(i)} + \boldsymbol{t}_{u,\mathsf{L}}^{(i)}\right), 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{A}\boldsymbol{z}^{(i)} - \sum_{u=1}^{U} c_u^{(i)} \boldsymbol{t}_u^{(i)}, 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{w}^{(i)} - c_1^{(i)}\boldsymbol{s}^{(i)} - \sum_{u=2}^{U} c_u^{(i)} \boldsymbol{s}_{u,2}, 2U\gamma\right) \\
&= \mathsf{HighBits}_q\left(\boldsymbol{w}^{(i)}, 2U\gamma\right) \\
&= \boldsymbol{w}_{\mathsf{H}}^{(i)},
\end{aligned}
$$

and the verifying condition (b) of Ver holds. Then, it holds

$$\Pr[\mathsf{Win}_7] = \Pr[\mathsf{Win}_6]. \tag{18}$$

$\Im_8$ In this game, $\mathbf{pk}^* = \left(\boldsymbol{t}_1^*, \boldsymbol{t}_0^*\right)$ is generated in a way that $\boldsymbol{t}^* = 2^d \cdot \boldsymbol{t}_1^* + \boldsymbol{t}_0^*$ for $\boldsymbol{t}^* \in_{\mathrm{U}} R_q^k$, instead of $\boldsymbol{t}^* = \boldsymbol{A}\boldsymbol{s}_1^* + \boldsymbol{s}_2^*$ in **Init** phase. Therefore, we have

$$|\Pr[\mathsf{Win}_8] - \Pr[\mathsf{Win}_7]| \leq \epsilon_{\mathrm{MLWE}}. \tag{19}$$

Upper bound of winning probability of $\Im_8$ We evaluate the upper bound of the winning probability of $\Im_8$ by using the GSPB game. We first construct a GSPB adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ from $\Im_8$ which is depicted in Fig. 10. On the evaluation of the winning probability of $\mathcal{A}$, we first fix $\mathbf{pk}^*$. $\mathcal{A}_1$ decides the probability that $g(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau) = 1$ from the fraction of $c \in \mathcal{CH}$ such that c induces the win of $\mathcal{F}$ on $\Im_8$. While $\mathcal{A}_2$ aims to find $\left(\boldsymbol{w}_{\mathsf{H}}^*, \mathsf{PK}^* \setminus \{\mathbf{pk}^*\}, C^*, \mu^*, \tau_1^*\right)$ such that $g\left(\boldsymbol{w}_{\mathsf{H}}^*, \mathsf{PK}^* \setminus \{\mathbf{pk}^*\}, C^*, \mu^*, \tau_1^*\right) = 1$ by playing $\Im_8$ with running $\mathcal{F}$ and simulating the random oracle H_2.

We now verify that H_2 depicted in Fig. 10 is indistinguishable from H_2 of $\Im_8$. Observe that H_2 depicted in Fig. 10 returns a hash value in the same way as $\Im_8$ when an input $(\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, P, \mu, \tau)$ satisfies $\mathbf{pk} \neq \mathbf{pk}^*$ or has been programmed. Otherwise, H_2 chooses a hash value by using Uni, where the internal random coins of Uni are given by the $2(Q_2 + Q_S UE)$-wise independent function $F(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau)$, and its value is chosen according to δ. In order to estimate the validness, we now consider that Uni is used with purely random coins instead of the output of F. Since $\lambda_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau)$ is defined by $\left|\mathcal{CH}_{\mathbf{pk}^*}^{\mathsf{Good}}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right|/|\mathcal{CH}|$, the hash value of H_2 in this case is uniformly distributed over $\mathcal{CH}$. We return to the situation where the output of F is used. It follows from Lemma 3 that $\mathcal{F}$ cannot distinguish the random oracle on H_2 in the case of Fig. 10 from that of $\Im_8$. Lemma 1 implies that under the fixed challenge public key $\mathbf{pk}^*$, we have

$$\Pr[\mathsf{Win}_8 \mid \mathbf{pk}^*] \leq \Pr\left[\mathbf{Exp}_{\lambda_{\mathbf{pk}^*}, \mathcal{A}}^{\mathsf{GSPB}}(\kappa) = 1 \mid \mathbf{pk}^*\right] \leq 8\lambda_{\mathbf{pk}^*}(Q_2 + Q_S UE + 1)^2, \tag{20}$$

where

$$\lambda_{\mathbf{pk}^*} = \max_{\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau} \lambda_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau).$$

From Eq. (20), the winning probability of $\Im_8$ can be evaluated by averaging Eq. (20) over the choice of $\mathbf{pk}^*$. Therefore, we have

$$\Pr[\mathsf{Win}_8] \leq 8\mathbb{E}\left[\lambda_{\mathbf{pk}^*}\right](Q_2 + Q_S UE + 1)^2, \tag{21}$$

$\underline{\mathcal{A}_1(1^\kappa)}$

(i) choose H_0 uniformly at random from a set of all polynomials of degree $2(Q_0 + Q_S UE)$, initialize a database D for H_1, choose $H_2' \in_{\mathrm{U}} \mathsf{Fun}(\mathfrak{D}_2, \mathcal{CH})$, and choose a $2(Q_2 + Q_S UE)$-wise independent functions F.
(ii) $\varrho \leftarrow \mathsf{Setup}(1^\kappa)$; $\boldsymbol{A} = \mathsf{Sam}(\varrho)$.
(iii) $\boldsymbol{t}^* \in_{\mathrm{U}} R_q^k$; $\mathbf{pk}^* = \left(\boldsymbol{t}_1^*, \boldsymbol{t}_0^*\right)$ satisfies that $\boldsymbol{t}^* = 2^d \cdot \boldsymbol{t}_{\mathsf{H}}^* + \boldsymbol{t}_{\mathsf{L}}^*$.
(iv) for all $\boldsymbol{w}_{\mathsf{H}}$, $C = (c_2, \ldots, c_U)$, $P = (\mathbf{pk}_2, \ldots, \mathbf{pk}_U)$, μ, τ,
(iv.1) set $\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)$ to the following:

$$\left\{ c \in \mathcal{CH} \mid \exists(\boldsymbol{z}, \boldsymbol{h}) \text{ s.t. } \begin{array}{l} |\boldsymbol{z}|_\infty \le U(\gamma' - \beta) \wedge \\ \boldsymbol{w}_{\mathsf{H}} \\ \quad = \mathsf{UseHint}_q\left(\boldsymbol{h}, \boldsymbol{A}\boldsymbol{z} - 2^d c \boldsymbol{t}_{\mathsf{H}}^* - 2^d \sum_{u=2}^{U} c_u \boldsymbol{t}_{u,\mathsf{H}}, 2U\gamma\right) \end{array} \right\},$$

where $\mathbf{pk}_u = (\boldsymbol{t}_{u,\mathsf{H}}, \boldsymbol{t}_{u,\mathsf{L}})$ and
(iv.2) $\lambda_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau) = \left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| / |\mathcal{CH}|$.
(v) return $\left(\left\{\lambda_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau)\right\}_{\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau}, \left(H_0, D, H_2', F, \varrho, \mathbf{pk}^*\right)\right)$.

$\underline{\mathcal{A}_2^{|g\rangle}(H_0, D, H_2', F, \varrho, \mathbf{pk}^*)}$

(i) run $\mathcal{F}(\varrho, \mathbf{pk}^*)$ with emulating C of $\mathfrak{I}_8$ to finally obtain $(\mathsf{PK}^*, \mu^*, \sigma^*)$, where $\sigma^* = \left(\boldsymbol{w}_{\mathsf{H}}^*, \boldsymbol{z}^*, \boldsymbol{h}^*\right)$. Here, the random oracle on H_2 is simulated as below. Let $\mathsf{PK}^* = \{\mathbf{pk}^*, \mathbf{pk}_2^*, \ldots, \mathbf{pk}_U^*\}$
(ii) abort if $\neg(\mathbf{pk}^* \in \mathsf{PK}^* \wedge (\mu^*, \mathsf{PK}^*)$ does not queried $\wedge\ \mathsf{Ver}(\mathsf{PK}^*, \mu^*, \sigma^*) = 1)$.
(iii) for each $1 \le u \le U$, $\tau_u^* = H_1\left(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}^*, \mathsf{PK}^*, \mu^*\right)$.
(iv) for each $2 \le u \le U$, $c_u^* = H_2\left(\mathbf{pk}_u^*, \boldsymbol{w}_{\mathsf{H}}^*, \mathsf{PK}^* \setminus \{\mathbf{pk}_u^*\}, \mu^*, \tau_u^*\right)$.
(v) $C^* = \left(c_2^*, \ldots, c_U^*\right)$.
(vi) return $\left(\boldsymbol{w}_{\mathsf{H}}^*, \mathsf{PK}^* \setminus \{\mathbf{pk}^*\}, C^*, \mu^*, \tau_1^*\right)$.

$\underline{H_2^{|g\rangle}(\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, P, \mu, \tau)}$, where $P = (\mathbf{pk}_2, \ldots, \mathbf{pk}_U)$

(i) return $H_2'(\mathbf{pk}, \boldsymbol{w}_{\mathsf{H}}, P, \mu, \tau)$ if $\mathbf{pk} \neq \mathbf{pk}^*$.
(ii) return the reprogrammed value c if $(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}, P, \mu, \tau)$ has been programed in **Sign** phase.
(iii) $\mathsf{PK} = (\mathbf{pk}^*, \mathbf{pk}_2, \ldots, \mathbf{pk}_U)$.
(iv) for each $2 \le u \le U$, $\tau_u = H_1(\mathbf{pk}_u, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK}, \mu)$ and $c_u = H_2'(\mathbf{pk}_u, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK} \setminus \{\mathbf{pk}_u\}, \mu, \tau_u)$.
(v) $C = (c_2, \ldots, c_u)$.
(vi) $\tau = H_1(\mathbf{pk}^*, \boldsymbol{w}_{\mathsf{H}}, \mathsf{PK}, \mu)$.
(vii) $\delta = g(\boldsymbol{w}_{\mathsf{H}}, P, C, \tau, \mu)$.
(viii) return $c = \mathsf{Uni}\left(\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C); F(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau)\right)$ if $\delta = 1$.
(ix) return $c = \mathsf{Uni}\left(\mathcal{CH} \setminus \mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C); F(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau)\right)$ if $\delta = 0$.

Fig. 10. GSPB adversary $\mathcal{A}$ from $\mathfrak{I}_8$

where $\mathbb{E}\left[\lambda_{\mathbf{pk}^*}\right]$ denotes the expected value of $\lambda_{\mathbf{pk}^*}$ over the choice of $\mathbf{pk}^*$.

We next evaluate $\mathbb{E}\left[\lambda_{\mathbf{pk}}^*\right]$. As in (iv) of $\mathcal{A}_1$, $\lambda_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C, \mu, \tau)$ is defined by $\left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| / |\mathcal{CH}|$. This implies that $\mathbb{E}\left[\lambda_{\mathbf{pk}}^*\right]$ is at most the probability of Loss: Loss denotes the event that any unbounded adversary meets $c \in \mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)$ for an arbitrary chosen tuple $(\boldsymbol{w}_{\mathsf{H}}, P, C)$ when $c \in_{\mathrm{U}} \mathcal{CH}$, where $P = (\mathbf{pk}_2, \ldots, \mathbf{pk}_U)$ and $C = (c_2, \ldots, c_U)$. In order to evaluate the probability of Loss, we now show that $\left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| = 1$ with high probability over the choice of $\mathbf{pk}^*$. Namely, we estimate the probability that $\left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| \ge 2$.

Assume that for a tuple $(\boldsymbol{w}_{\mathsf{H}}, P, C)$ where $P = (\mathbf{pk}_2, \ldots, \mathbf{pk}_U)$ and $C = (c_2, \ldots, c_U)$, there exist different values $c, c' \in \mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)$. Namely, there exist $(\boldsymbol{z}, \boldsymbol{h})$ and $(\boldsymbol{z}', \boldsymbol{h}')$ such that $|\boldsymbol{z}|_\infty, |\boldsymbol{z}'|_\infty < U(\gamma' - \beta)$, and

$$\boldsymbol{w}_{\mathsf{H}} = \mathsf{UseHint}_q\left(\boldsymbol{h}, \boldsymbol{A}\boldsymbol{z} - 2^d c \boldsymbol{t}_{\mathsf{H}}^* - 2^d \sum_{u=2}^{U} c_u \boldsymbol{t}_{u,\mathsf{H}}, 2U\gamma\right); \text{ and}$$

$$\boldsymbol{w}_{\mathsf{H}} = \mathsf{UseHint}_q\left(\boldsymbol{h}', \boldsymbol{A}\boldsymbol{z}' - 2^d c' \boldsymbol{t}_{\mathsf{H}}^* - 2^d \sum_{u=2}^{U} c_u \boldsymbol{t}_{u,\mathsf{H}}, 2U\gamma\right),$$

where $\mathbf{pk}_u = (\boldsymbol{t}_{u,\mathsf{H}}, \boldsymbol{t}_{u,\mathsf{L}})$ for $2 \le u \le U$. It follows from Lemma 7 that

$$\left\| \boldsymbol{A}\boldsymbol{z} - 2^d c\boldsymbol{t}^*_{\mathsf{H}} - 2^d \sum_{u=2}^{U} c_u \boldsymbol{t}_{u,\mathsf{H}} - \boldsymbol{w}_{\mathsf{H}} \cdot 2U\gamma \right\|_\infty \le 2U\gamma + 1; \text{ and}$$
$$\left\| \boldsymbol{A}\boldsymbol{z}' - 2^d c'\boldsymbol{t}^*_{\mathsf{H}} - 2^d \sum_{u=2}^{U} c_u \boldsymbol{t}_{u,\mathsf{H}} - \boldsymbol{w}_{\mathsf{H}} \cdot 2U\gamma \right\|_\infty \le 2U\gamma + 1.$$

This implies that

$$\left| \boldsymbol{A}(\boldsymbol{z} - \boldsymbol{z}') - 2^d (c - c')\boldsymbol{t}^*_{\mathsf{H}} \right|_\infty \le 4U\gamma + 2.$$

This means that there exists $\boldsymbol{u} \in S^k_{4U\gamma+2}$ such that

$$\boldsymbol{A}(\boldsymbol{z} - \boldsymbol{z}') + \boldsymbol{u} = 2^d (c - c')\boldsymbol{t}^*_{\mathsf{H}}.$$

Let $\Gamma = \{c - c' \mid c, c' \in \mathcal{CH} \wedge c \neq c'\}$. Note that $|\Gamma| \le |\mathcal{CH}|^2$. By letting $\boldsymbol{z}^* = \boldsymbol{z} - \boldsymbol{z}' \in S^k_{2U(\gamma'-\beta)-2}$ and $c^* = c - c'$, we have

$$\boldsymbol{A}\boldsymbol{z}^* + \boldsymbol{u} = 2^d c^* \boldsymbol{t}^*_{\mathsf{H}}$$

By the settings $\ell \le k$ and $2^d < 4U(\gamma' - \beta) - 4$ in Table 1 and Lemma 11, it holds that

$$\begin{aligned}
&\Pr_{\boldsymbol{A} \in_{\mathrm{U}} R_q^{k \times \ell}, \boldsymbol{t}^* \in_{\mathrm{U}} R_q^k} \left[\begin{array}{l} \exists (\boldsymbol{z}^*, \boldsymbol{u}, c^*) \in S^{\ell}_{2U(\gamma'-\beta)-2} \times S^k_{4U\gamma+2} \times \Gamma \\ \text{s.t. } \boldsymbol{A}\boldsymbol{z}^* + \boldsymbol{u} = c^* \boldsymbol{t}^*_{\mathsf{H}} \cdot 2^d \end{array} \right] \\
&\le 2|\Gamma| \left(\frac{(2(2U(\gamma' - \beta) - 2) + 1)^\ell (2(4U\gamma + 2) + 1)^k}{q^k} \right)^n \\
&\le 2|\mathcal{CH}|^2 \left(\frac{(2(2U(\gamma' - \beta) - 2) + 1)^\ell (2(4U\gamma + 2) + 1)^k}{q^k} \right)^n \\
&= 2|\mathcal{CH}|^2 \left(\frac{(4U\gamma' - 4U\beta - 3)^\ell (8U\gamma + 5)^k}{q^k} \right)^n \\
&< 2|\mathcal{CH}|^2 \left(\frac{(4U\gamma')^k 8U\gamma^k}{q^k} \right)^n \\
&= 2|\mathcal{CH}|^2 \left(\frac{32U^2\gamma'\gamma}{q} \right)^{kn}.
\end{aligned} \tag{22}$$

On the other hand, the probability of Loss when $\left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| = 1$ is evaluated by $1/|\mathcal{CH}|$. This is because we now consider the event that $c \in \mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)$ for $c \in_{\mathrm{U}} \mathcal{CH}$. By Eq. (22), the probability of Loss is evaluated as follows: for arbitrary chosen tuple $(\boldsymbol{w}_{\mathsf{H}}, P, C)$,

$$\begin{aligned}
\Pr[\mathsf{Loss}] &= \Pr\left[\mathsf{Loss} \wedge \left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| = 1\right] + \Pr\left[\mathsf{Loss} \wedge \left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| \ge 2\right] \\
&\le \Pr\left[\mathsf{Loss} \mid \left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| = 1\right] + \Pr\left[\left|\mathcal{CH}^{\mathsf{Good}}_{\mathbf{pk}^*}(\boldsymbol{w}_{\mathsf{H}}, P, C)\right| \ge 2\right] \\
&< \frac{1}{|\mathcal{CH}|} + 2|\mathcal{CH}|^2 \left(\frac{32U^2\gamma'\gamma}{q} \right)^{kn}.
\end{aligned} \tag{23}$$

Putting together with Eqs. (21) and (23), we have

$$\begin{aligned}
\Pr[\mathsf{Win}_8] &\le 8\mathbb{E}\left[\lambda_{\mathbf{pk}^*}\right](Q_2 + Q_S UE + 1)^2 \\
&\le 8 \Pr[\mathsf{Loss}](Q_2 + Q_S UE + 1)^2 \\
&< 8 \cdot \left(\frac{1}{|\mathcal{CH}|} + 2|\mathcal{CH}|^2 \left(\frac{32U^2\gamma'\gamma}{q} \right)^{kn} \right) \cdot (Q_2 + Q_S UE + 1)^2,
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\epsilon &< \epsilon_{\mathrm{MLWE}} + \frac{2\sqrt{Q_1}}{2^{\theta/2}} + \frac{Q_S E(4 + \sqrt{2})\sqrt{Q_2 + (Q_S E - 1)U}}{2^{\theta/4}} + Q_S E \epsilon_{\mathrm{rMLWE}} \\
&\quad + 8 \cdot \left(\frac{1}{|\mathcal{CH}|} + 2|\mathcal{CH}|^2 \left(\frac{32U^2\gamma'\gamma}{q} \right)^{kn} \right) \cdot (Q_2 + Q_S UE + 1)^2 + \epsilon_{\boldsymbol{w}_{\mathsf{H}}} + \epsilon_{\mathsf{OW}} + \mathrm{negl}.
\end{aligned}$$

The proof is completed. □

Acknowledgements. We would like to thank anonymous reviewers for their valuable comments and suggestions. We are also grateful to Akira Takahashi for his fruitful comments on the security proof. A part of this work is supported by JSPS KAKENHI Grant Numbers 18K11288 and 19K20272.

References

[AFLT16] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly secure signatures from lossy identification schemes. *Journal of Cryptology*, 29(3):597–631, Jul 2016.

[BBE+18] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 354–384, Cham, 2018. Springer International Publishing.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 41–69, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 390–399, New York, NY, USA, 2006. ACM.

[dC17] Alexandre de Castro. Quantum one-way permutation over the finite field of two elements. *Quantum Information Processing*, 16(6), Apr 2017.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 356–383, Cham, 2019. Springer International Publishing.

[DKL+18] Léo Ducas, Eike Kiltz, Tancréde Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018.

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan A. Garay, editor, *Public-Key Cryptography – PKC 2021*, pages 99–130, Cham, 2021. Springer International Publishing.

[EBS16] Rachid El Bansarkhani and Jan Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security*, pages 140–155, Cham, 2016. Springer International Publishing.

[FH19] Masayuki Fukumitsu and Shingo Hasegawa. A tightly-secure lattice-based multisignature. In *Proceedings of the 6th on ASIA Public-Key Cryptography Workshop*, APKC '19, pages 3–11, New York, NY, USA, 2019. ACM.

[FH20] Masayuki Fukumitsu and Shingo Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model. In Khoa Nguyen, Wenling Wu, Kwok Yan Lam, and Huaxiong Wang, editors, *Provable and Practical Security*, pages 45–64, Cham, 2020. Springer International Publishing.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of cryptology*, 26(1):80–101, 2013.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 530–547, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. ACM.

[IN83] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignature. *NEC Research and Development*, 71:1–8, 1983.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 552–586, Cham, 2018. Springer International Publishing.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 155–164, New York, NY, USA, 2003. ACM.

[LZ19] Qipeng Liu and Mark Zhandry. Revisiting post-quantum Fiat-Shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 326–355, Cham, 2019. Springer International Publishing.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[NIS17] NIST. Post-quantum cryptography, 2017. Online: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography, Accessed: November 17, 2019.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 187–196, New York, NY, USA, 2008. ACM.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In Martin Hirt and Adam Smith, editors, *Theory of Cryptography*, pages 192–216, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 755–784, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[Zha12] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 758–775, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Zha19] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 239–268, Cham, 2019. Springer International Publishing.