Scalable Two-Round n-out-of-n and Multi-Signatures from Lattices in the Quantum Random Oracle Model

Qiqi Lai[1], Feng-Hao Liu[2],Yang Lu[1], Haiyang Xue[3],Yong Yu[1]

[1] School of Computer Science, Shaanxi Normal University, Xi'an, China.
laiqq@snnu.edu.cn, luyang@snnu.edu.cn, yuyong@snnu.edu.cn.
[2] Washington State University, Pullman, WA, USA. feng-hao.liu@wsu.edu.
[3] Singapore Management University, Singapore. haiyangxc@gmail.com.

Abstract. In this paper, we construct the first asymptotically efficient two-round n-out-of-n and multi-signature protocols from lattices in the quantum random oracle model (QROM), using the Fiat-Shamir with Aborts (FSwA) paradigm. Our protocols can be viewed as the improvement of the two-round protocols by Damgård et al. (JoC 2022). A notable feature of our protocols, compared to other counterparts in the classical random oracle model, is that each party performs an independent abort and still outputs a signature in exactly two rounds, making our schemes significantly more scalable.

From a technical perspective, the simulation of QROM and the efficient reduction from breaking underlying assumption to forging signatures are the essential challenges to achieving efficient QROM security for the previously related works. In order to conquer the former one, we adopt the quantum-accessible pseudorandom function (QPRF) to simulate QROM. Particularly, we show that there exists a QPRF, which is not only both invertible and programmable, but also its output space is separable, even against a quantum adversary. For the latter challenge, we tweak and apply the online extractability by Unruh (Eurocrypt 2015).

1 Introduction

Due to the applications in the decentralized scenarios, distributed signing protocols have recently received wide attentions. *Threshold signature* [20] and *multi-signature* [35] are two similar classes of distributed signing protocols.

In a nutshell, a t-out-of-n *threshold signature* involves a distributed key generation process, where each user obtains a share, say sk_i, of the single signing key sk, effectively distributing signing power among n participants, such that a message can only be signed if t or more participants agree to do so. The special case of n-out-of-n signature is achieved by setting $t = n$, which is also called as *decentralized signature* (DS).

On the other hand, a *multi-signature* (MS) allows a group of n users, each holding signing key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$, to collectively sign the same message and obtain a single signature. It differs from threshold signature in several ways: (i) there is no distributed key generation, as each participant generates their own key pairs; (ii) the group of participants is dynamically formed from certain sets; (iii) as a result, the verification process does not rely on a fixed public key, but rather on the list or aggregation of public keys of the participants.

State-of-the-art of Thershold/Multi signature. Numerous recent studies have focused on Schnorr-type schemes, such as the threshold version of ECDSA [14, 22, 58, 59], threshold Schnorr [39, 41], and multi-signing of Schnorr [3, 6, 48, 50, 53]. However, Schnorr-type constructions are vulnerable to quantum attacks, it is crucial to investigate the threshold and multi-signature of post-quantum alternatives. Significant attention has been directed towards lattice-based digital signatures, particularly variants of Dilithium [27, 28], which has been chosen as a standard by NIST [52].

Dilithium is based on the "Fiat-Shamir with Aborts" (FSwA) technique of Lyubashevsky [44], which employs rejection sampling to guarantee security. However, unlike Schnorr, thresholding Dilithium presents greater challenges due to the adoption of the "Aborts" and small Gaussian distributions. Several schemes with other technique routes have been proposed, but they either rely on costly primitives, such as fully homomorphic encryption in [2, 11, 33], or fail to achieve comparable efficiency and full security based on standard assumptions [30].

Damgård, Orlandi, Takahashi, and Tibouchi [18] address the challenge by using homomorphic trapdoor commitment schemes as a building block. They proposed threshold signatures (specifically, n-out-of-n) and multi-signatures in both two-round and three-round, denoted as $\mathsf{DS}_2/\mathsf{MS}_2$ and $\mathsf{DS}_3/\mathsf{MS}_3$, respectively. Notably, DS_2 and MS_2 are the first lattice-based two-round schemes.

Subsequently, $\mathsf{MulSig\text{-}L}$ by Boschini et al. [13] and DualMS by Chen [15] improved upon $\mathsf{DS}_2/\mathsf{MS}_2$. Compared with $\mathsf{DS}_2/\mathsf{MS}_2$, $\mathsf{MulSig\text{-}L}$ does not rely on additional lattice-based trapdoor commitments and benefits from preprocessing in the first round. DualMS further enhances the construction by utilizing a trapdoor-free "dual signing simulation" technique, resulting in smaller public keys, signatures, and reduces communication.

Common problems in lattice-based setting. All these two-round FSwA paradigm protocols with polynomial modulus can only been proven to be secure in the classical random oracle model (ROM), through using the currently known techniques. Additionally, all the protocols must be restarted until all participants pass the rejection sampling step (i.e., without abort) simultaneously. This will lead to an exponential increase (in the number of participants) in expected communication, computation, and rounds, which makes them "non-scalability".[4]

<u>QUANTUM ROM.</u> For post-quantum cryptographic constructions in the ROM, we need to further consider that the quantum adversary might conduct arbitrary

[4] Of course, we can trivially make all these existing two-round schemes "scalability", through directly using the noise flooding technique, instead of rejection sampling technique. But, the corresponding cost is super-polynomial modulus, which will results in bad efficiency. Thus, we just focus on FSwA paradigm constructions with polynomial modulus in this paper.

superposition queries to ROM, which is further called QROM and first proposed by Boneh et al. in [10]. Currently, for DS and MS, only three-round schemes DS_3 and multi-signature in [30] achieve security in the QROM by utilizing the technique of lossy identification [38]. The QROM security for all known existing two-round schemes, DS_2/MS_2, MulSig-L, and DualMS, remains an open problem.

Independent Abort. Another drawback is that, due to the requirement of abort, the final round of these schemes will not output the final signature until non-abort happens to all of the participants. Thus, the probability of successfully outputting a signature decreases with an exponential in the number n of parties. Specifically, assuming the non-abort probability of each party is $1/M$, the success probability will be reduced to $1/M^n$ when n parties are involved. In this case, the expected communication round will increase to M^n from 2. While parallel executions can be applied to reduce the round, it still results in an increase in the expected communication and computation overheads. Particularly, the whole protocol need to be parallel run about $\tau = \lambda/(\log \frac{M}{M-1})$ times, in order to ensure the parties output a signature with overwhelming probability. In this case, the final communication and computation overheads of each party will expand about τ times, contrasted to the originally theoretical ones.

One might consider to reduce $1/M^n$ to $1/M$ again, by setting the standard deviation σ' of the discrete Gaussian distribution to be $n \cdot \sigma$, where σ denotes the original standard deviation for the non-distributed signature. However, just as pointed out in [18], this method will increase the size of each signature share, and affect the parameters of the underlying hard problem in the security reduction. Besides, this makes the choice of concrete parameters, especially the the standard deviation σ', inflexible with the number of involved parties scales. In summary, current schemes are not "scalable", when deploying with polynomial modulus.

1.1 Our Goals

Therefore, the motivation of this paper is to design two round n-out-of-n and multi-signatures for Dilithium, which benefit from QROM *security*, *round efficiency*, and *polynomial modulus* properties.

- QROM *security*: The security of these protocols are proved in the QROM.
- *Round-efficiency:* We focus on efficient protocols with *exact* two-round. In this case, the communication complexity of each party will remain to be unchanged, even when the participant number increases. As a result, our schemes will be much more scalable, compared with other related works.
- *Polynomial modulus*: The final protocols can be instantiated with polynomial modulus, which will have much better asymptotical efficiency.

This work aims to solve these challenges with the following particular goals.

(**Main Goal 1 (for Security)**) Design FSwA-style two-round n-out-of-n and multi-signatures from lattices in the QROM.

(**Main Goal 2 (for Efficiency)**) Design efficient FSwA-style two-round n-out-of-n signatures and multi-signatures, in which each party's communication overhead remains to be independent of the number of parties, in the case of broadcast channel.

1.2 Our Contributions

This work aims at the two main goals and makes the following three major contributions.

Contribution 1. We construct the first two-round n-out-of-n signature from lattices in the QROM, which can be viewed as the significant improvements of two-round protocol by Damgård et al. in [18]. For the security proof of our new construction, we leverage the online extractability technology by Unruh in [57] and our newly modified QPRF (which is introduced as the independent technique contribution in the following Contribution 3). Besides, our new construction has much lower security loss, as we use online extractability instead of usual rewinding method. Particularly, we improve the adversary's advantage for forging a signature from $\Theta(\varepsilon_{\mathsf{MSIS}}^{1/2})$ or $\Theta(\varepsilon_{\mathsf{MSIS}}^{1/4})$ to $\Theta(\varepsilon_{\mathsf{MSIS}})$, where $\varepsilon_{\mathsf{MSIS}}$ denotes the hardness of the underlying MSIS assumption.

Besides, our new n-out-of-n signature is the first lattice-based FSwA-style construction with both exact two-round and polynomial modulus. All previously known FSwA-style distributed protocols parameterized with polynomial modulus output a signature only with certain probability. The advantage of our exact two-round protocol is that each party's communication and computation overhead will be almost unchanged, even with the increasing of the participant number in the protocols. Thus, our constructions are more scalable than all other related constructions. From the perspective of technique, such exact two-round is obtained through embedding the protocol challenge in cut-and-choose way, rather than every participant directly choosing small ring elements as independent challenges.

Contribution 2. We construct the first two-round FSwA-style multi-signature from lattices in the QROM, which has the same properties (i.e., exact two-round, lower security loss, and polynomial modulus) as the above mentioned n-out-of-n signature. However, compared with n-out-of-n signature, there is an additional obstacle in the security proof. In order to conquer it, we prove the security in the key-verification model, rather than the plain public key model. In the key-verification model, we use a multi-proof straight-line extractable non-interactive zero knowledge proof of knowledge (NIZKPoK) as an additional building block. Although such NIZKPoK will induce many more overheads, it gives another nice property, key aggregation.

Contribution 3. As a technical contribution, we make essential modifications on the direct QPRF construction in [60], which was originally proposed by Banerjee et. al.'s in [7], such that it becomes to be an invertible variant. Besides, we show that the invertible QPRF is programable simultaneously against efficient quantum adversary conducting superposition queries.

With such QPRF, we can prove the security of our two-round protocols in the QROM. Particularly, we can efficiently simulate the adversary's view even without the real secret key, and then establish the efficient reduction from the underlying assumptions to the security of our constructions. We believe such a new QPRF should be of independent interest.

Overall, we list the detailed comparisons with the most related works in Table 1.

	QROM	**Round**	**Scalable**	**Reduc.-Appro.**	**Assumptions**
[30]	✓	3	×	Lossy	MLWE,rMLWE
DS_3 [18]+ [30]	✓	3	×	Lossy	MLWE
DS_2 [18]	×	2	×	Rewinding	MLWE,MSIS
MS_2 [18]	×	2	×	Rewinding	MLWE,MSIS
[13]	×	2	×	Rewinding	MLWE,MSIS
[15]	×	2	×	Rewinding	MLWE,MSIS
Our DS_2	✓	2	✓	Online-Extractability	MLWE,MSIS
Our MS_2	✓	2	✓	Online-Extractability	MLWE,MSIS

Table 1. Comparison with previous FSwA-based distributed and multi-signatures. The column "Reduc. Appro." indicates the security reduction approach.

Moreover, after an integrated evaluation, we conclude that the asymptotical communication overhead of our protocols is comparable with that of [18], especially when a large number of participants are involved. More details are deferred to Section 5.3. Besides, our constructions have the nice property of highly parallelization. For any P up to the security parameter λ, each participant can allocate $O(\lambda/P)$ of its computations to each of P processors.

2 Technical Overview

In this section, we present an overview of our techniques. In fact, our intuition is quite simple: analyze the essential obstacles to proving the existing two-round signatures to be secure in the QROM, then overcome them to achieve the desired security relying on the MLWE, MSIS and DSPR assumptions.

2.1 Recall the existing protocol in [18].

To present our techniques in a natural way, we first recall Damgård et al.'s elegant two-round distributed n-out-of-n protocol, called DS_2, in [18].[5] The protocols

[5] Here, we are just interested in two-round protocols, even our techniques can also apply to their three-round setting. Besides, such n-out-of-n construction can be easily extended to the setting of multi-signature.

in [18] are based on Dilithium signature scheme [27, 28], and thus work over cyclotomic rings $R = \mathbb{Z}[X]/(f(X))$ and $R_q = \mathbb{Z}_q[X]/(f(X))$, where N is a power of 2, and $f(X) = X^N + 1$ is the $2N$-th cyclotomic polynomial. For simplicity, here we just consider the case of 2 parties, which can be naturally generalized to the case of n-party setting. Particularly, we assume each party P_j has a secret vector $\boldsymbol{s}_j \in R^{\ell+k}$ and a matrix $\mathbf{A}_j \in R_q^{k\times\ell}$, where $j \in \{1,2\}$, $\boldsymbol{s}_j$ consists of small coefficients, and $\mathbf{A}_j$ is randomly sampled from $R_q^{k\times\ell}$. Then, P_1 and P_2 interactively generate the finally joint public verification key $\mathsf{pk} := (\hat{\mathbf{A}}, \boldsymbol{t}) = ([\mathbf{A}_1 + \mathbf{A}_2, \mathbf{I}], [\mathbf{A}_1 + \mathbf{A}_2, \mathbf{I}] \cdot (\boldsymbol{s}_1 + \boldsymbol{s}_2))$, through using the simple random oracle commitments as in left hand side of Figure 1.

KeyGen of DS_2 executed by $P_2(\mathsf{pp})$		**DS_2 executed by $P_2(\boldsymbol{s}_2, \mathsf{pk} := (\mathbf{A}, \boldsymbol{t}), \mu)$**	
Sample $\mathbf{A}_2 \xleftarrow{\$} R_q^{k\times\ell}$, compute $g_2 \leftarrow \mathsf{H}_1(\mathbf{A}_2, 2)$			
	$\xrightarrow{g_2}$	$\boldsymbol{y_2} \xleftarrow{\$} D^{\ell+k}$; $\boldsymbol{w_2} \leftarrow \hat{\mathbf{A}}\boldsymbol{y_2} \bmod q$	
	$\xleftarrow{g_1}$	$\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$	
	$\xrightarrow{A_2}$		
	$\xleftarrow{A_1}$	$\mathsf{com}_2 \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\boldsymbol{w_2}, r_2)$	
Check $g_1 \stackrel{?}{=} \mathsf{H}_1(\mathbf{A}_1, 1)$			$\xrightarrow{\mathsf{com}_2}$
If YES, compute $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$ and set $\hat{\mathbf{A}} = [\mathbf{A}, \mathbf{I}] \in R_q^{k\times(\ell+k)}$			$\xleftarrow{\mathsf{com}_1}$
Sample $\boldsymbol{s}_2 \xleftarrow{\$} S_\beta^{\ell+k}$, compute $\boldsymbol{t_2} := \hat{\mathbf{A}} \cdot \boldsymbol{s}_2$, $g_2' \leftarrow \mathsf{H}_2(\boldsymbol{t}_2, 2)$		$c \leftarrow \mathsf{H}_0(\mathsf{com}_1 + \mathsf{com}_2, \mu, \mathsf{pk})$ $\boldsymbol{z_2} \leftarrow c\boldsymbol{s_2} + \boldsymbol{y_2}$	
	$\xrightarrow{g_2'}$	If RejSamp $(c\boldsymbol{s_2} + \boldsymbol{z_2}) = 0$;	
	$\xleftarrow{g_1'}$	$(\boldsymbol{z_2}, \boldsymbol{r_2}) \leftarrow (\perp, \perp)$	
	$\xrightarrow{\boldsymbol{t}_2}$		$\xrightarrow{\boldsymbol{z_2}, \boldsymbol{r_2}}$
	$\xrightarrow{\boldsymbol{t}_1}$		$\xleftarrow{\boldsymbol{z_1}, \boldsymbol{r_1}}$
		If $\boldsymbol{z_1} = \perp \vee \boldsymbol{z_2} = \perp$: restart	
Check $g_1' \stackrel{?}{=} \mathsf{H}_2(\boldsymbol{t}_1, 1)$		Output $(\mathsf{com}_1 + \mathsf{com}_2, \boldsymbol{z_1} + \boldsymbol{z_1}, r_1 + r_2)$	
If YES, set $\boldsymbol{t} = \boldsymbol{t_1} + \boldsymbol{t}_2$		as a signature	

Fig. 1. Simple description of DS_2 Protocol in [18]. Here, we just describe the behaviors of P_2 for saving space, due to the fact that the computations and communications of P_1 are symmetrically equivalent to these of P_2.

The signature phase of DS_2 is described as in the right hand side of Figure 1. Particularly, DS_2 utilizes as a building block a homomorphic (equivocation)-trapdoor commitment scheme, which is the core tool to enable the full security proof from lattices. In the first round, given message μ, secret key $\boldsymbol{s}_j$ and the joint public key $\mathsf{pk} := (\hat{\mathbf{A}}, \boldsymbol{t})$, the party P_j first samples small vector $\boldsymbol{y}_j$, and computes $\mathsf{com}_j \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\boldsymbol{w}_j; r_2)$, where $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$, $\boldsymbol{w}_j = \hat{\mathbf{A}}\boldsymbol{y}_j \bmod q$ and r_2 is the randomness sampled from the corresponding set. Then, after exchanging their commitments com_1 and com_2, the party P_j computes the challenge $c \leftarrow \mathsf{H}(\mathsf{com}_1 + \mathsf{com}_2, \mu, \mathsf{pk})$, through using the linearly homomorphic property of trapdoor commitment. Moreover, P_j computes the response vector $\boldsymbol{z}_j = c \cdot \boldsymbol{s}_j + \boldsymbol{y}_j$ and conducts the rejection sampling algorithm to decide whether output $\boldsymbol{z}_j$ or not. If all parties output successfully, then they exchange the pairs $(\boldsymbol{z}_j, r_j)_{j\in\{1,2\}}$. The final signature is the addition of individual signature shares, i.e., $\mathsf{Sig} = (\mathsf{com}_1 + \mathsf{com}_2, \boldsymbol{z}_1 + \boldsymbol{z}_2, r_1 + r_2)$.

Security analysis of DS_2. In order to establish the security reduction for DS_2, there are the following important items need to be considered:

1. For the key generation, the simulator utilizes the invertible (or called pre-image extractable in certain previous papers) and programable properties of classical ROM (H_1 and H_2) to answer the adversary's key query and embed the MSIS challenge into the public key of DS_2.
2. For the signature queries, the simulator employs a homomorphic trapdoor-equivocation commitment scheme to enable the successful simulation of signatures, and thus achieves the full security proof in the ROM.
3. For the reduction from solving hard problems to forging signature, the simulator needs to simulate H_3 such that its output space is separable, i.e., the output space of H_3 can be divided into two parts: the normal commitment key ck and the trapdoor commitment key tck. Particularly, the indistinguishability of ck and tck allows the adversary to use ck for the challenge message μ^*, but use tck for all the signature generation queries, with a non-negligible probability. Then, assuming the binding property of the commitment scheme with normal key ck, the simulator can solve an underlying MSIS problem, through using rewinding technique.

2.2 Enhancing the security of DS_2 into the QROM

Based on the above analyses, we know that there are two main parts for the security proof of DS_2: the efficient simulations of random oracles and the efficient reduction through rewinding. Thus, a straight way to extend the security of DS_2 into the QROM is that: we need to ensure all the above techniques have the corresponding counterparts for the quantum adversary and the QROM setting. Below, we analyze the above security items one by one. Among this process, we will also insert our considerations on how to obtain much better efficiency.

Simulation of QROM. Clearly, we need to consider how to simulate QROM efficiently, such that not only it can be both invertible and programmable, but also its output space is separable, following the proof strategy of [18]. Up until now, there are two types of simulation approaches for QROM: stateless simulation (e.g. quantum-secure pseudorandom functions [10, 60] and $2Q$-wise independent functions [23,61]), and stateful simulation (e.g. the compressed oracle [25,42,62]).

Among them, the compressed oracle is the most powerful simulation technique, which can import almost all classical ROM security proof into the QROM setting. Particularly, the compressed oracle can be roughly viewed as the on-the-fly simulation of QROM, which supports both inversion and reprogramming [24, 25, 42], without previously bounding the adversary's queries times. However, it seems that we can not apply the compressed oracle to directly achieve the desired separable property, due to the inconsistency between the commitment key space S_{ck} of $\mathsf{ck}, \mathsf{tck}$ and the output of compressed oracle.

Particularly, the output of compressed oracle is inherently uniform over $\{0,1\}^n$.[6] But, in the known instantiations of trapdoor commitment scheme as in Section A.7, the commitment keys $\mathsf{ck}, \mathsf{tck}$ indeed have the particular format, such that they are just uniform over the key space S_{ck}, a subspace of $\{0,1\}^n$. To make matters even more complicated, when H_3 outputs a tck, the simulator also need the corresponding trapdoor td to generate the simulated signature. But, the adversary can only obtain tck from H_3. If using compressed oracle to simulate H_3, tck and td should be stored as superpositions over different registers, and thus it might need complex strategies to deal with the above different requirements between simulator and the adversary.

In contrast, a stateless simulation method, say QPRF, might be compatible with the separable property, as it is possible to use two different but related functions to deal with the consistency of tck and td. But, for QPRF, it is still not clear how to make it to be both invertible and programmable. Besides, we also need to consider how to limit the output of QPRF into the key space S_{ck}.

Notice also that, the technical details of the compressed oracle [62] and even its further simplified exposition in [16] are relatively complicated. One always needs additional investigations on the related backgrounds. So, as the main technic challenge in this paper, we want to ask

> *Is there much simpler and stateless simulation approach for* QROM*, which is not only both invertible and programmable, but also its output space is separable, just through using the simple and classical lattice-based concepts and techniques?*

In this paper, our answer is positive. Particularly, our choice is QPRF [60]. [60]. We illustrate the high-level idea for using QPRF in the security proof in Figure 2. Of course, we require the concrete instantiation of QPRF is both invertible *and* programmable, even against quantum adversaries, which are significantly non-trivial. As far as we know, it seems that this is the first time to consider how to program and invert QPRF simultaneously in the literature, even the invertible property for PRF has been defined previously in [12,34]. We believe such QPRF is of independent interest, and can be used on other applications, such as PIR [34].

More precisely, we choose to use the following direct construction of QPRF in [7,60]: for a key $\mathsf{k} := (\{a_i\}_{i\in[\bar{m}]}, \{s_i\}_{i\in[\bar{\ell}]}) \in \mathcal{K}$ and input $x := (x_1, \ldots, x_{\bar{\ell}}) \in \{0,1\}^{\bar{\ell}}$, let

$$\mathsf{QPRF}_{\mathsf{k}}(x) = \mathsf{QPRF}_{\{a_i\},\{s_i\}}(x_1, \ldots, x_{\bar{\ell}}) = \left\lfloor (a_1, \ldots, a_{\bar{m}}) \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \right\rceil_{\bar{p}}^{\bar{q}}, \tag{1}$$

[6] In the compressed oracle simulation of QROM $\mathsf{H} : \{0,1\}^m \to \{0,1\}^n$, if the adversary query $|x, y\rangle$ but a tuple $(x, \cdot)$ is not found in the compressed database D (D is initialized to be an empty database), then the simulator will create a uniform superposition $\frac{1}{\sqrt{2^n}} \sum_{\alpha_x \in \{0,1\}^n} |\alpha_x\rangle$ in new registers (this corresponds to choose α_x uniformly at random from $\{0,1\}^n$). Then the simulator adds (x, α_x) into D, and responds the query with $|x, y \oplus \alpha_x\rangle$.

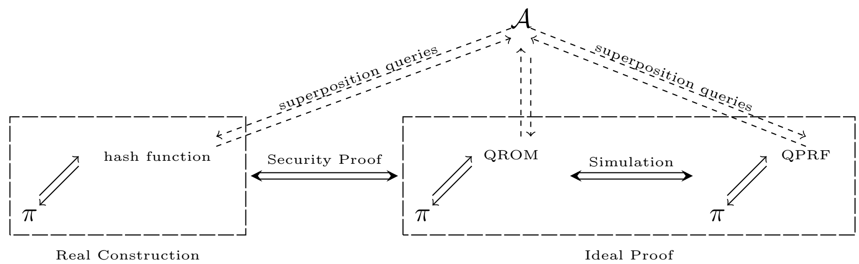

Fig. 2. Illustration of the high-level idea of using QPRF. Clearly, such QPRF is only used in the ideal security proof. In the real constructions, it will be replaced with a good hash function, such as SHA-256. And thus, the instantiation of QPRF will not affect the practical efficiency of our QDS_2 or QMS_2.

where $\bar{p}, \bar{q}, \bar{m}, \bar{\ell}$ are integers such that $\bar{q} > \bar{p}$, a_i is chosen from a ring $\bar{R}_{\bar{q}} = \mathbb{Z}_{\bar{q}}[X]/(X^{\bar{N}} + 1)$, and s_i is chosen from a small distribution χ over $\bar{R}$.[7] For certain parameter settings, such QPRF can be proven to be secure, just as in Theorem 4.3. Below, we sketch how to prove the properties of (1).

Inversion. For the invertible property of QPRF, it is essentially non-trivial. This is because the pseudorandomness of QPRF implicitly implies the one-way property, and it should be impossible to invert the input from certain QPRF output value. But, the situation might be quite different, when the simulator just uses it to simulate QROM. Here the simulator holds the secret key for QPRF, and the adversary can only get outputs through querying the simulator. Even with such new application scenario in our security proof, all existing known QPRFs, including the above (1), still do not satisfy our requirement of inversion.

As one of our significant technical contribution, we tweak the direct QPRF in [60], through (i) embedding a MP trapdoor [49] in the vector $\boldsymbol{a}^\top = (a_1, \ldots, a_{\bar{m}})$; (ii) choosing the specific ring structure such that the small ring element are invertible over $\bar{R}_{\bar{q}}$. Notice that, such a modification will not affect the security of the direct QPRF, as the ring-based learning with errors (RLWE) assumption still holds. Moreover, we can invert such QPRF, i.e., get $\boldsymbol{x}$ from $\boldsymbol{y}^\top = \mathsf{QPRF}(\boldsymbol{x}) \in \bar{R}_{\bar{p}}^{\bar{m}} = (\mathbb{Z}_{\bar{p}}[X]/(X^{\bar{N}}+1))^{\bar{m}}$, in the following way: (i) with the MP trapdoor in $\boldsymbol{a}^\top$, we can first get $\hat{s}_0 = \prod_{i=1}^{\ell} s_i^{x_i}$ from $\boldsymbol{y}^\top$, through using the inversion algorithm for the ring-based learning with round (RLWR) problem; (ii) in order to determine $x_1 = 0$ or 1, we directly compute $\hat{s}_1 = s_1^{-1} \cdot \hat{s}_0$. Notice that if $x_1 = 1$, then $\hat{s}_1 = \prod_{i=2}^{\ell} s_i^{x_i}$, whose norm should be upper bounded by certain value B with overwhelming probability. Otherwise, $\hat{s}_1 = s_1^{-1} \cdot \prod_{i=2}^{\ell} s_i^{x_i}$, whose norm will be larger than B with overwhelming probability, according to the decisional small polynomial rate (DSPR) assumption [43]. The detailed inversion algorithm and the related proof are given in Algorithm 1 and Theorem 4.7, respectively.

[7] Here, we use the bar notation to distinguish the notations of QPRF from those of the protocols DS, MS, QDS, QMS. Particularly, the parameter setting of QPRF is independent of those of protocols in this paper. And thus, the modulus of our protocols is still considered to be polynomial, regardless of the modulus of QPRF.

<u>Reprogramming.</u> In order to prove the reprogramable property for QPRF, we resort to the existing adaptive programming result and technique for random function in [32, 57]. The high level technique route can be described as follows:

$$\mathsf{QPRF}_k(\cdot) \overset{(i)}{\approx} \boxed{\mathsf{RF}(\cdot) \approx \mathsf{RF}'(\cdot)} \overset{(ii)}{\approx} \mathsf{QPRF}'_k(\cdot). \tag{2}$$

Here, we use the box to indicate the existing result about the adaptive programming for random functions in [32, 57]. $\mathsf{RF}(\cdot)$ denotes a random function. $\mathsf{RF}'(\cdot)$ and $\mathsf{QPRF}'_k(\cdot)$ denote the programmed functions at certain points. Below, we just need to consider how to prove the steps (i) and (ii) in the above (2).

At the first glance, it seems that the standard security of QPRF, i.e., *oracle indistinguishability* in [60], is sufficient. However, there is still a tiny mismatching! This is because for the box part in the above (2), the adversary not only accesses RF as an oracle, but also takes as input certain pairs $(x, \mathsf{RF}(x))$, where x has sufficient entropy. As a result, it is necessary to introduce a "seemingly" strong definition, oracle-and-input indistinguishability, for QPRF as in Definition 3.7. Furthermore, as a justification of such strong security notion, we show that it can be derived from standard oracle indistinguishability in Lemma 3.8.

<u>The separable property.</u> In order to simulate the signature successfully even without secret key, we need to rely on the separable property of H_3, i.e., its output separates the normal commitment key ck from the trapdoor commitment key tck. For an instantiation of QPRF, its output space is determinate. And thus, we can not directly match the output of QPRF with the spaces of valid ck and tck. In order to conquer this challenge, we first choose a specific key $\mathsf{k}_3 \overset{\$}{\leftarrow} \mathcal{K}$, and then divide the output of QPRF into two parts: $\{0,1\}^{\ell_{ra_1}}$ and $\{0,1\}^{\ell_{ra_2}}$. Particularly, we define

$$\mathsf{QPRF}_{\mathsf{k}_3}(\cdot) : \{0,1\}^{l_3^*} \to (\{0,1\}^{l_{ra_1}} \times \{0,1\}^{l_{ra_2}}),$$

and compute

$$(ra_1, ra_2) = \mathsf{QPRF}_{\mathsf{k}_3}(\mu, \mathsf{pp}, \mathsf{pk}),$$

where μ denotes the signing message, pp and pk are public parameter and the generated public key by KeyGen. Then, if the number of 1 in ra_1 is larger than certain value num, then we compute $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{Eqv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Eqv}}, ra_2)$, and return tck. Otherwise, compute $\mathsf{ck} \leftarrow \mathsf{Eqv\text{-}CGen}(\mathsf{cpp}_{\mathsf{Eqv}}, ra_2)$, return ck. Here, num is set to separate tck and ck with certain probability. $\mathsf{cpp}_{\mathsf{Eqv}}$ are public parameter of trapdoor commitment scheme. $\mathsf{Eqv\text{-}TCGen}$ and $\mathsf{Eqv\text{-}CGen}$ are two modes of trapdoor commitment scheme.

Moreover, as the above simulation of H_3 is stateless and a fixed function, we can easily define another fixed function H'_3, which will be consistent with H_3 and return the corresponding trapdoor td for simulation purpose. More details are presented in Figure 15.

Notice that the above partitioning argument inherently lead to a security loss linear in the number of signing queries. In [53, 54], Pan and Wagner have successfully remove such reduction loss through using pseudorandom matching/path technique. But, there are still other technical obstacles to instantiate their elegant approaches in the lattice-based settings. We leave it as future work.

Furthermore efficiency consideration. Up until now, we have successfully obtained the desired QPRF in theory, which satisfies inversion, reprogramming and separation, simultaneously. However, after analyzing it deeply, we find its drawback on the input length. Particularly, according to Theorem 4.3, the parameters need to satisfy $\bar{q} \geq O(\bar{\ell} \cdot (\sqrt{2(\bar{N}+\bar{\ell})})^{\bar{\ell}} \cdot \bar{N}^{\omega(1)})$, and the security of such QPRF is based on the $\mathsf{RLWE}_{\bar{q},1,\bar{m},\chi}$ assumption. Thus, in order to ensure its security, there should be an implicitly upper bound for the input length $\bar{\ell}$, say $\bar{\ell} \leq O(\bar{N}^{1/6})$. On the other hand, according to the KeyGen protocol in the left hand side of Figure 1, the input length of random oracles $\mathsf{H}_1, \mathsf{H}_2$ should be $(\ell k N \cdot \log q)$ and $(kN \cdot \log q)$, respectively. If directly using the same QPRF to simulate H_1 and H_2, we need to ensure $\ell k N \cdot \log q \leq O(\bar{N}^{1/6})$, through setting sufficiently large $\bar{N}$. But, such a large $\bar{N}$ will significantly affect the computation efficiency of the used QPRF, which further affect reduction loss. So, we want to ask if we can design more efficient KeyGen protocol, such that we can reduce the input length of the random oracle H_1. The answer is affirmative.

In order to compress the input length of H_1, our general idea is to introduce another random oracle H'_1. And each participant's random matrix $\mathbf{A}_u$ is generated as $\mathbf{A}_u \leftarrow \mathsf{H}'_1(s_u)$, where $s_u \xleftarrow{\$} \{0,1\}^{l^*_2}$ is a random seed with $u \in \{1,2\}$. In this case, the participants just need to interactively send the $\mathsf{H}_1(s_u)$ and s_u, rather than $\mathsf{H}_1(\mathbf{A}_u)$ and $\mathbf{A}_u$. Clearly, s_u is significantly shorter than $\mathbf{A}_u$. Thus, the most significant benefit of the above usage of H'_1 is that the new key generation protocol is more efficient in practice. Besides, in the security proof, we just need to invert H_1 and reprogram H'_1, rather than both inversion and reprogramming. Particularly, our modified protocol is described in Figure 3.

Modified KeyGen Protocol of QDS_2

Sample $s_2 \xleftarrow{\$} \{0,1\}^{\ell^*_2}$, compute $g_2 \leftarrow \mathsf{H}_1(s_2, 2)$

$\xrightarrow{g_2}$

$\xleftarrow{g_1}$

$\xrightarrow{s_2}$

$\xleftarrow{s_1}$

Check $g_1 \stackrel{?}{=} \mathsf{H}_1(s_1, 1)$

If YES, compute $\mathbf{A}_u = \mathsf{H}'_1(s_u)$, $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$, and set $\hat{\mathbf{A}} = [\mathbf{A}, \mathbf{I}] \in R_q^{k \times (\ell+k)}$

Sample $\boldsymbol{s}_2 \xleftarrow{\$} S_\beta^{\ell+k}$, compute $\boldsymbol{t}_2 := \hat{\mathbf{A}} \cdot \boldsymbol{s}_2$, $g'_2 \leftarrow \mathsf{H}_2(\boldsymbol{t}_2, 2)$

$\xrightarrow{g'_2}$

$\xleftarrow{g'_1}$

$\xrightarrow{\boldsymbol{t}_2}$

$\xrightarrow{\boldsymbol{t}_1}$

Check $g'_1 \stackrel{?}{=} \mathsf{H}_2(\boldsymbol{t}_1, 1)$, If YES, set $\boldsymbol{t} = \boldsymbol{t_1} + \boldsymbol{t}_2$

Fig. 3. Simple description of our modified KeyGen protocol for QDS_2 Protocol. Similar to Figure 1, we just describe the behaviors of P_2 for saving space.

With such KeyGen protocol, we can set $\bar{N}$, such that $kN \cdot \log q \leq O(\bar{N}^{1/6})$. Moreover, if with $k = 1$, then we just need to set $N \cdot \log q \leq O(\bar{N}^{1/6})$, which will significantly reduce the value of $\bar{N}$, and thus improve reduction efficiency.

For security reduction of $\mathsf{DS_2}$. One of essential obstacles for proving the QROM security of two round n-out-of n distributed signatures is the application of rewinding techniques, just as pointed out by [13, 18]. This is because the operation of measuring the state of the quantum adversary $\mathcal{A}$ before rewinding will essentially disturb the state of $\mathcal{A}$. And thus, the rewinding will return to an undefined earlier state [57].

Notice that in order to conquer this dilemma on quantum rewinding, Liu and Zhandry have proposed the collapsing technique [42], which can generally derive the QROM security of the existing lattice-based FSwA-style signatures, such as Dilithium-G signature scheme [27,28]. However, we can not apply this collapsing technique [42] to the settings of distributed signatures and multi-signatures. This is because we do not know how to define the compatible lossy/separable functions just as in [42]. Furthermore, the measure-and-reprogram technique has been proposed to prove the QROM security of Fiat-Shamir signature [23] and directly evaluate the QROM security of Dilithium [36]. But such elegant technique can not be applied to the QROM security of the distributed signature.

Another widely used approach of obtaining lattice-based distributed signatures in the QROM is the lossy ID technique [1, 18, 30, 38], which can obtain much tighter security proof. One implicit but crux point in this lossy ID technique is that there should be statistical security in the simulated experiment, i.e., the probability of forging a valid signature should be negligible, even for computationally unbounded adversary.

Recall that the homomorphic trapdoor-equivocation commitment scheme used in the DS_2 protocol just inherently satisfies computational binding, and do not satisfy the essential requirement of lossy technique. Thus, it seems that we need new security proof techniques for proving the security of two-round distributed signature protocol in the QROM.

2.3 New idea: Online extractability allowing security proof in the QROM.

Online extractability is another reasonable candidate direction to achieve security in the QROM. We notice an existing online extractability technique by Pino and Katsumata in [19, 37].[8] Particularly, they proposed a semi-generic transformation, which compiles lattice-based Σ-protocol into QROM-secure NIZKPoK. It seems that such an online extractability method can be adapted to the settings of distributed signature. However, their online extractability technique relies heavily on a primitive called extractable linear homomorphic commitment. And it seems that the extractable property of such commitment scheme is inherently not compatible with the equivocation property required for DS_2 in Figure 1. Thus, it is still not clear how to directly apply this online extractability technique to our desired settings.

[8] In their paper, such a property is named as straight-line extractability.

Let us recall the online extractability in [19, 37] again, whose intuitive is to efficiently find more than one valid response $\boldsymbol{z}$ with respect to different challenges such as c and c', from just one valid $\boldsymbol{z}$. So, inspired by Pino and Katsumata's technique, one crucial observation is that the party P_2 can directly generate more than one response $\boldsymbol{z}_{2,j}$ with respect to the same vector $\boldsymbol{w}_2 = \mathbf{A} \cdot \boldsymbol{y}_2$, where each different $\boldsymbol{z}_{2,j}$ is computed with different challenge $c_{2,j}$. I.e., $\boldsymbol{z}_{2,j} = \boldsymbol{y}_2 + c_{2,j} \cdot \boldsymbol{s}_2$ and $\boldsymbol{z}_{2,j'} = \boldsymbol{y}_2 + c_{2,j'} \cdot \boldsymbol{s}_2$. Based on this, given one forged signature, if we can find two valid responses for two different challenges, we can extract the witness through using the special soundness extractor of the underlying Σ-protocol.

Of course, in order to avoid the trivial extractability from normal valid signature, we need to first hide all different responses by certain hash function, i.e., just send out $h_{2,j} = \mathsf{H}(\boldsymbol{z}_{2,j})$ rather than sending all $\boldsymbol{z}_{2,j}$ in clear.[9] Then, we can use the idea of cut-and-choose to decide which $\boldsymbol{z}_{2,j}$ will be disclosed. Notice that if the value j is randomly chosen, we can easily prove its soundness. More importantly, such a new cut-and-choose proof idea provide a chance to allow each participant conducting rejection sampling independently, which is the essential idea to make our protocols to be exact two-round and scalable.

In fact, the above analysis is matched with the essential idea of Unruh in [57]. Notice that in [24], Don et al. further propose a much better technique to improve the framework of [57]. However, such an improvement can not apply to our distributed signatures.

One more subtlety. Even almost all security targets for two-round protocols have been achieved, there is still one subtlety: the hash function used to hide the responses $\boldsymbol{z}_j$. Here, as we consider for the case of all parties cooperating to sign message μ, we require it satisfy the linear homomorphic property. Besides, we also need such a hiding function to have binding and trapdoor-inversion properties, for the reason of security proof. So, we replace the hash function with a homomorphic trapdoor-inversion commitment scheme.

Putting all above ingredients together. We present our main two-round protocol QDS_2 in Figure 4. Below, we slightly analyze QDS_2. Compared with the sign protocol in the right hand side of Figure 1, there are several differences deriving some extra efficiency advantages. First, we notice that the real challenge for our QDS_2 is J output by the random oracle H_5. And the challenges $\{c_j\}_{j \in [m]}$ outputted by H_0 are just required to be different from each other, rather than ensuring enough soundness for the underlying Σ-protocol. [10] In this case, the parties in QDS_2 first run the rejection sampling algorithm, and then interactively send transcripts, in contrast to the reverse order in DS_2. With this particular

[9] The other one desired property of such a hash function H is collision resistance, which details are deferred to the security proof in Section 5.2.

[10] Here is another difference, that is c_j does not depend on any $\boldsymbol{w}_j^{(2)}$ or $\mathsf{com}^{(2)}$. But this will not affect our security, due to the following two reasons: (1) the output distribution of rejection sampling algorithms is still simulateable; (2) for the underlying Σ-protocol, the adversary can not forge the valid responses with respect to two different challenges c_{j_1}, c_{j_2}, with $j_1 \neq j_2$.

feature, the outcome of each party's rejection sampling will not affect other parties. And regardless of the number of parties in the system, the whole distributed signature protocol will determinedly output the correct signature, after exactly two round interactions. This makes our QDS_2 has the incomparable advantage on the round complexity over other related two-round FSwA-style distributed signature protocols.

QDS_2 **executed by** $P_2(\mathsf{sk} := \boldsymbol{s}_2, \mathsf{pk} := (\mathbf{A}, \boldsymbol{t}), \mu)$

i. $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk}), \mathsf{ck}' \leftarrow \mathsf{H}_4(\mu, \mathsf{pk})$

ii. $\boldsymbol{y}_2 \leftarrow D_\sigma^{\ell+k}, \boldsymbol{w}_2 = \hat{\mathbf{A}} \cdot \boldsymbol{y}_2 \mod q$

iii. Sample r_2 and compute $\mathsf{com}_2 \leftarrow \mathsf{Eqv.Commit}_{\mathsf{ck}}(\boldsymbol{w}_2; r_2)$

iv. For $j = 1$ to m, conduct as follows:

$$\begin{cases} A. & c_j \leftarrow \mathsf{H}_0(\mu, j, \mathsf{pk}, \mathsf{ck}, \mathsf{ck}') \\ B. & \boldsymbol{z}_{2,j} = c_j \cdot \boldsymbol{s}_2 + \boldsymbol{y}_2 \\ C. & \text{run} \quad \mathsf{Rej}(\boldsymbol{z}_{2,j}, c_j \cdot \boldsymbol{s}_2, \sigma) \to b_{2,j} \end{cases}$$

v. If $b_{2,j} = 0$ for certain $j \in [m]$, then go to Step(ii)

vi. Sample $r'_{2,j}$ and compute $\widetilde{\mathsf{com}}_{2,j} \leftarrow \mathsf{Inv.Commit}_{\mathsf{ck}'}(\boldsymbol{z}_{2,j}; r'_{2,j})$

$\xrightarrow{\mathsf{com}_2, \{\widetilde{\mathsf{com}}_{2,j}\}_{j \in [m]}}$

$\xleftarrow{\mathsf{com}_1, \{\widetilde{\mathsf{com}}_{2,j}\}_{j \in [m]}}$

vii. $\mathsf{com} = \mathsf{com}_1 + \mathsf{com}_2, \widetilde{\mathsf{com}}_j = \widetilde{\mathsf{com}}_{1,j} + \widetilde{\mathsf{com}}_{2,j}$
$J \leftarrow \mathsf{H}_5(\mathsf{pk}, \mu, \mathsf{com}, \{c_j\}, \{\widetilde{\mathsf{com}}_j\})$

$\xrightarrow{\boldsymbol{z}_{2,J}, r_2, r'_{2,J}}$

$\xleftarrow{\boldsymbol{z}_{1,J}, r_1, r'_{1,J}}$

viii. Output $\mathsf{Sig} := \big(\mathsf{com}, r = r_1 + r_2, \{\widetilde{\mathsf{com}}_j\}, \{\boldsymbol{z}_J = \boldsymbol{z}_{1,J} + \boldsymbol{z}_{2,J}\}, \{r'_J = r'_{1,J} + r'_{2,J}\}\big)$ as a signature

Fig. 4. Our Two-Round n-out-of-n Distributed Signature Protocol

Second, in order to ensure the domain of J is large enough, we might need to set the parameter m in Figure 4 to be at least equivalent to the security parameter λ. This will clearly cause the significantly size expansion, which seems to be unavoidable. Fortunately, we can first set a relative small value for m, and then conduct the parallelization to the current protocol for enough times. In this way, with the almost same size overhead, we can make our protocol to be highly parallelizable. This means for any P up to the security parameter λ, each participant can allocate $O(\lambda/P)$ of its computations to each of P processors. In this case, the overall computation time of our protocol will be reduced significantly.

Third, as we adopt the online extractability, instead of rewinding, to establish the reduction from the underlying MSIS problem to the unforgeability, our protocol should have much lower security loss than others with rewinding. This means that in theory, we can set much better parameters for the fixed security level.

2.4 Two-round multi-signature in the QROM.

Similar to [18], we can also convert the above QDS_2 into a two-round multi-signature protocol QMS_2 following [26], where our QMS_2 has the nice property of key aggregation. However, after applying all above mentioned techniques, there is still one reduction gap from the fully secure multi-signature. Particularly, through using the above online extractability technique, we can solve the MSIS problem with respect to $[\hat{\mathbf{A}}, \sum_{i\in|L|} \boldsymbol{t}_{j_i}]$ from the forged signature Sig^* output by the adversary, where L is the set of participants in the current running of QMS_2. Without loss of generality, for multi-signature protocol, we suppose the j_1-th participant is honest and all others are corrupted together with the adversary. In this case, we should use the reduction algorithm to solve the MSIS problem with respect to $[\hat{\mathbf{A}}, \boldsymbol{t}_{j_1}]$, rather than $[\hat{\mathbf{A}}, \sum_{i\in|L|} \boldsymbol{t}_{j_i}]$. And it seems to be an inherent obstacle for obtaining solutions of MSIS with respect to $[\hat{\mathbf{A}}, \boldsymbol{t}_{j_1}]$, from that of $[\hat{\mathbf{A}}, \sum_{i\in|L|} \boldsymbol{t}_{j_i}]$.

In order to conquer this dilemma, we try to enhance the multi-signature protocol into the key-verification model, where we require each participant to publish a multi-proof straight-line extractable NIZKPoK on his/her secret key sk_j with respect to the corresponding public key pk_j. Then, through using the extractability property of NIZKPoK, we can patch the above mentioned reduction gap, and thus obtain a provably secure multi-signature protocol in the key-verification model. In practice, one participant might want to ensure that the public keys of all his parters are well-formed, before jointing into one multi-party protocol. And thus, we believe such a key-verification model is reasonable, even it implicitly implies slightly many more overheads. The formal and detailed protocol of our two-round multi-signature is presented in Section D.

2.5 Other Related Work

Notice that MulSig-L in [13] and DualMS in [15] can be viewed as significant improvements over MS_2 in [18]. Thus, one might hope to apply our new techniques to [13, 15], which will derive more efficient two-round QMS_2. But, it seems that the double-forking technique used in [13, 15] is inherently not compatible with the online extractability in [57]. Of course, if we abandon the key-aggregation property in [13, 15], i.e., just use one-forking in the security proof, it is possible to leverage our techniques in this paper to obtain QROM security. But, this still need further analyses. We left it as future directions.

3 Preliminaries

Due to space limit, we defer the detailed descriptions on the notations, backgrounds on discrete gaussian distribution, definitions on underlying assumptions such as MSIS, MLWE, DSPR, and rejection sampling together with the signature scheme Dilithium in Sections A.1, A.2, A.3, and A.4, respectively.

3.1 Quantum Computation and Quantum Random Oracle Model

In this Section, we recall several basic results on Quantum Computation and Quantum Random Oracle Model.

Fact 3.1 (Fact 1 in [60]) *For any classical efficiently computable function f, we can efficiently implement it by a quantum computer. Moreover, f can be implemented as an oracle which can be queried on quantum superpositions.*

Definition 3.2 (Quantum Random Oracle, QROM) *Given sets X and Y, let $\mathsf{Fun}(X,Y)$ be the set of all functions $\mathsf{H}: X \to Y$. The quantum random oracle model (QROM) is a security model, in which any adversary $\mathcal{A}$ gets hash values from the random oracle by querying the oracle on quantum superpositions. Moreover, for a random hash function $\mathsf{H} \in \mathsf{Fun}(X,Y)$, we write $\mathcal{A}^{|\mathsf{H}\rangle}$ to denote that $\mathcal{A}$ can query the random oracle H in superpositions.*

There are several ways to simulate the QROM. Here, we recall techniques of replacing the random oracle with quantum-secure pseudorandom function (called QPRF, defined in Section 3.2)

Fact 3.3 ([10,60]) *For any sets X and Y, we can use quantum-secure pseudorandom function to efficiently simulate quantum random oracle from X to Y, when considering efficient quantum adversary.*

3.2 Quantum-Secure Pseudorandom Function

Definition 3.4 (PRF [60]) *A pesudorandom function is a function $\mathsf{PRF}: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{K}, \mathcal{X}, \mathcal{Y}$ are the key-space, domain and range, respectively. Implicitly, the settings of $\mathcal{K}, \mathcal{X}, \mathcal{Y}$ depend on the security parameter λ. Given any pair $(\mathsf{k}, x) \in \mathcal{K} \times \mathcal{X}$, there exists $y \in \mathcal{Y}$, which can be written as $y = \mathsf{PRF}_{\mathsf{k}}(x)$.*

Definition 3.5 (Classical Security) *A pseudorandom function PRF is classical security, if no efficient quantum adversary $\mathcal{A}$ making classical queries can distinguish between a truly random function and the function $\mathsf{PRF}_{\mathsf{k}}$ for a random $\mathsf{k} \in \mathcal{K}$. More formally, for any efficient quantum adversary $\mathcal{A}$, there exists a negligible function $\varepsilon = \varepsilon(\lambda)$ such that $\left| \Pr_{\mathsf{k} \xleftarrow{\$} \mathcal{K}}[\mathcal{A}^{\mathsf{PRF}_{\mathsf{k}}}(\cdot) = 1] - \Pr_{O \xleftarrow{\$} \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{O}(\cdot) = 1] \right| < \varepsilon$, where $\mathcal{Y}^{\mathcal{X}}$ denotes the class of all functions from $\mathcal{X}$ to $\mathcal{Y}$.*

Notice that in Definition 3.5, we only allow $\mathcal{A}$ to conduct classical queries, even $\mathcal{A}$ itself is a quantum algorithm. Below, we generalize the definition to allow $\mathcal{A}$ to conduct quantum queries, i.e., directly query one superposition of all $x \in \mathcal{X}$ each time.

Definition 3.6 (Quantum Security) *A pseudorandom function PRF is quantum security, if no efficient quantum adversary $\mathcal{A}$ making quantum queries can distinguish between a truly random function and the function $\mathsf{PRF}_{\mathsf{k}}$ for a random $\mathsf{k} \in \mathcal{K}$. More formally, for any efficient quantum adversary $\mathcal{A}$, there exists a negligible function $\varepsilon = \varepsilon(\lambda)$ such that $\left| \Pr_{\mathsf{k} \xleftarrow{\$} \mathcal{K}}[\mathcal{A}^{|\mathsf{PRF}_{\mathsf{k}}\rangle}(\cdot) = 1] - \Pr_{O \xleftarrow{\$} \mathcal{Y}^{\mathcal{X}}}[\mathcal{A}^{|O\rangle}(\cdot) = 1] \right| < \varepsilon$.*

Such quantum secure pseudorandom functions are called Quantum Pseudorandom Functions, or QPRF. In fact, the above security in Definitions 3.5 and 3.6 are called as Oracle-Indistinguishability, as in [60]. In this paper, we need to use the following "seemingly" strong quantum security: Oracle-and-input indistinguishability.

Definition 3.7 (Strong Quantum Security) *A* QPRF *is strong quantum security, if no efficient quantum adversary $\mathcal{A}$ making quantum queries and taking several random input-output pairs as input can distinguish between a truly random function and the function* $\mathsf{PRF_k}$ *for a random* $\mathsf{k} \in \mathcal{K}$*. More formally, for any efficient quantum adversary $\mathcal{A}$, there exists a negligible function $\varepsilon = \varepsilon(\lambda)$ such that*

$$\Big| \Pr_{(\mathsf{k},x_1,\ldots,x_n)\xleftarrow{\$}\mathcal{K}\times\mathcal{X}^n}\left[\mathcal{A}^{|\mathsf{PRF_k}\rangle}\left((x_i,\mathsf{PRF_k}(x_i))_{i\in[n]}\right)=1\right] - \Pr_{(O,x_1,\ldots,x_n)\xleftarrow{\$}\mathcal{Y}^{\mathcal{X}}\times\mathcal{X}^n}\left[\mathcal{A}^{|O\rangle}\left((x_i,O(x_i))_{i\in[n]}\right)=1\right]\Big| < \varepsilon.$$

Lemma 3.8 (Oracle-and-input Indistinguishability) *If one* PRF *satisfies the standard quantum security as in Definition 3.6, then such* PRF *also satisfies the strong quantum security as in Definition 3.7.*

Proof. In order to prove such lemma in a more clear way, we first notice the following facts: Definitions 3.5, 3.6, and 3.7 can be depicted equivalently as the corresponding interactive experiments between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. Taking Definition 3.7 as example, we denote its interactive experiment as $\mathbf{Exp}^{\mathsf{S\text{-}IND}}_{\mathsf{QPRF}}(\mathcal{A})$, and a secure QPRF implies that for any efficient $\mathcal{A}$, the probability $\mathbf{Adv}^{\mathsf{S\text{-}IND}}_{\mathsf{QPRF}}(\mathcal{A}) := \Pr\left[\mathbf{Exp}^{\mathsf{S\text{-}IND}}_{\mathsf{QPRF}}(\mathcal{A}) \to 1\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$.

Suppose $\mathbf{Exp}^{\mathsf{Ind}}_{\mathsf{QPRF}}(\mathcal{A})$ and $\mathbf{Exp}^{\mathsf{S\text{-}IND}}_{\mathsf{QPRF}}(\mathcal{A})$ are the corresponding interactive experiments for Definitions 3.6 and 3.7, respectively. It suffices to show that if there is an efficient adversary $\mathcal{A}$ such that $\mathbf{Exp}^{\mathsf{S\text{-}IND}}_{\mathsf{QPRF}}(\mathcal{A}) \to 1$, then there is another efficient adversary $\hat{\mathcal{A}}$ such that $\mathbf{Exp}^{\mathsf{Ind}}_{\mathsf{QPRF}}(\hat{\mathcal{A}}) \to 1$. I.e., we need to prove

$$\Pr\left[\mathbf{Exp}^{\mathsf{S\text{-}IND}}_{\mathsf{QPRF}}(\mathcal{A}) \to 1\right] \le \Pr\left[\mathbf{Exp}^{\mathsf{Ind}}_{\mathsf{QPRF}}(\hat{\mathcal{A}}) \to 1\right]. \tag{3}$$

For (3), we need to establish the following reduction: given an unknown oracle, the algorithm $\hat{\mathcal{A}}$ can invoke the algorithm $\mathcal{A}$ as a subroutine to distinguish QPRF and O. And thus, we need to show that $\hat{\mathcal{A}}$ could simulate the environment of $\mathcal{A}$ and answer the queries of $\mathcal{A}$.

Particularly, let $\mathcal{A}$ to be an algorithm making q times quantum queries and taking n additional inputs $(x_i^*, O(x_i^*))_{i\in[n]}$ as input, where each x_i^* is randomly chosen from the domain $\mathcal{X}$, and q, n are polynomial in λ. And let $\hat{\mathcal{A}}$ to be an algorithm directly making $(q+n)$ times quantum queries.

As q times quantum queries of $\mathcal{A}$ can be easily simulated through querying the unknown oracle directly, we just need to consider how to provide n additional inputs $(x_i^*, O(x_i^*))_{i\in[n]}$ to $\mathcal{A}$. During all these additional n times superposition queries, $\hat{\mathcal{A}}$ can make and query the particular superpositions $\sum_{x_j \in \mathcal{X}} |x_j\rangle$, such that the function values of $(O(x_i^*))_{i\in[n]}$ or $(\mathsf{QPRF}_\mathsf{k}(x_i^*))_{i\in[n]}$ can be measured from the returned superpositions $\sum_{x_j \in \mathcal{X}} |O(x_j)\rangle$ or $\sum_{x_j \in \mathcal{X}} |\mathsf{QPRF}_\mathsf{k}(x_j)\rangle$ with overwhelming probability. For example, given a randomly chosen $x_i^* \in \mathcal{X}$, $\hat{\mathcal{A}}$ can directly generate the pure state of x_i^* or a superposition with most of wight over x_i^*, rather than an uniform position, such that the value of x_i^* can be successfully measured at least with overwhelming probability. And thus, for any $\{x_i^*\}_{i\in[n]}$, $\hat{\mathcal{A}}$ can provide n additional inputs $(x_i^*, O(x_i^*))_{i\in[n]}$ or $(x_i^*, \mathsf{QPRF}_\mathsf{k}(x_i^*))_{i\in[n]}$ to $\mathcal{A}$. □

3.3 Trapdoor Homomorphic Commitment Scheme

In this section, we recall the notion of trapdoor commitment scheme. According to the functionality of the trapdoor td, we can divide it into two different paradigms: Eqv-Trapdoor Commitment Scheme (Eqv-TCOM) and Inv-Trapdoor Commitment Scheme (Inv-TCOM). Particularly, for the case of Eqv-trapdoor, td is used to equivocate a commitment to an arbitrary message. But, for the case of Inv-trapdoor, td is used to invert a commitment to the underlying committed message. Of course, regardless of Eqv-case or Inv-case, the commitment scheme always satisfies the hiding and binding properties. Below, we present the syntaxes for Inv/Eqv-trapdoor commitment scheme.

Definition 3.9 (Eqv/Inv-Trapdoor Commitment Scheme [17]) *A trapdoor commitment scheme* Eqv/Inv-TCOM *consists of seven algorithms* (CSetup, CGen, Commit, Open, TCGen, Eqv-TCommit, Eqv, Inv) *as follows.*

- $\mathsf{CSetup}(1^\lambda) \to \mathsf{cpp}$*: The setup algorithm takes the security parameter λ as input, and outputs a public parameter* cpp *defining sets $S_\mathsf{ck}, S_\mathsf{msg}, S_r, S_\mathsf{com}$, and S_td and the distribution $\mathcal{D}(S_r)$ from which the randomness is sampled.*
- $\mathsf{CGen}(\mathsf{cpp}) \to \mathsf{ck}$*: The key generation algorithm takes* cpp *as input, and outputs a commitment key from S_ck.*
- $\mathsf{Commit}_\mathsf{ck}(\mathsf{msg}; \mathsf{Rand}) \to \mathsf{com}$*: The commit algorithm takes as input a message* $\mathsf{msg} \in S_\mathsf{msg}$ *and randomness* $\mathsf{Rand} \in S_r$*, and outputs commitment* $\mathsf{com} \in S_\mathsf{com}$.
- $\mathsf{Open}_\mathsf{ck}(\mathsf{com}, \mathsf{Rand}, \mathsf{msg}) \to b$*: The opening algorithm outputs $b = 1$ if the input tuple is valid, and $b = 0$ otherwise.*
- $\mathsf{TCGen}(\mathsf{cpp}) \to (\mathsf{tck}, \mathsf{td})$*: The trapdoor key generation algorithm takes* cpp *as input, and outputs* $\mathsf{tck} \in S_\mathsf{ck}$ *and the trapdoor* $\mathsf{td} \in S_\mathsf{td}$.
- $\mathsf{Eqv\text{-}TCommit}_\mathsf{tck}(\mathsf{td}) \to \mathsf{com}$*: The trapdoor committing algorithm takes* tck, td *as input, and outputs a commitment* $\mathsf{com} \in S_\mathsf{com}$.
- $\mathsf{Eqv}_\mathsf{tck}(\mathsf{td}, \mathsf{com}, \mathsf{msg}) \to \mathsf{Rand}$*: The equivocation algorithm takes as input* (td, com, msg)*, outputs randomness* $\mathsf{Rand} \in S_r$*, such that* $\mathsf{Open}_\mathsf{tck}(\mathsf{com}, \mathsf{Rand}, \mathsf{msg}) \to 1$.
- $\mathsf{Inv}_\mathsf{tck}(\mathsf{td}, \mathsf{com}) \to \mathsf{msg}$*: The invert algorithm takes* (td, com) *as input, and outputs the underlying message* $\mathsf{msg} \in S_\mathsf{msg}$ *of* com.

A usual commitment scheme COM is a special case of $\mathsf{Eqv/Inv\text{-}TCOM}$: it only consists of $\mathsf{CSetup}, \mathsf{CGen}, \mathsf{Commit}$, and Open. Of course, a concrete $\mathsf{Eqv\text{-}TCOM}$ scheme consists of seven algorithms: (CSetup, CGen, Commit, Open, TCGen, $\mathsf{Eqv\text{-}TCommit}$, Eqv). And, a concrete $\mathsf{Inv\text{-}TCOM}$ scheme consists of six algorithms: (CSetup, CGen, Commit, Open, TCGen, Inv).

Due to space limitation, we defer the formal presentations of the correctness, hiding, binding, key uniformness, and additive homomorphism to Section A.5, and present the detailed instantiations in Section A.7.

3.4 n-out-of-n Signature and Multi-Signature

Definition 3.10 (Distributed Signature Protocol) *A distributed signature protocol* QDS *consists of the following algorithms.*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: The algorithm takes a security parameter* λ *as input, and outputs public parameters* pp*.*
- $\mathsf{Gen}_j(\mathsf{pp}) \to (\mathsf{sk}_j, \mathsf{pk})$ *for every* $j \in [n]$*: The interactive key generation algorithm that is run by party* P_j*. Each* P_j *runs the protocol on public parameters* pp *as input. At the end of the protocol* P_j *obtains a secret key share* sk_j *and public key* pk*.*
- $\mathsf{Sign}_j(sid, \mathsf{sk}_j, \mathsf{pk}, \mu) \to \mathsf{Sig}$ *for every* $j \in [n]$*:The interactive signing algorithm that is run by party* P_j*. Each* P_j *runs the protocol on session ID sid, its signing key share* sk_j*, public key* pk*, and message to be signed* μ *as input. We also assume that the algorithm can use any state information obtained during the key generation phase. At the end of the protocol* P_j *obtains a signature* Sig *as output.*
- $\mathsf{Ver}(\mathsf{Sig}, \mu, \mathsf{pk}) \to b$ *: The verification algorithm that takes a signature, message, and a single public key* pk *and outputs* $b = 1$ *if the signature is valid and otherwise* $b = 0$*.*

Definition 3.11 (Multi-signature Protocol) *A multisignature protocol* QMS *consists of the following algorithms.*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *: The set up algorithm that outputs a public parameter* pp *on a security parameter* λ *as input.*
- $\mathsf{Gen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk}, \pi)$*: Given a public parameter* pp *as input, the non-interactive key generation algorithm outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$*, together with an* $\mathsf{NIZKPoK}$ *proof of validity of the public key* pk*, denoted as* π*.*
- $\mathsf{Sign}(\mathsf{pp}, sid, \mathsf{sk}, \mathsf{pk}, \mu, L) \to \mathsf{Sig}$*: The interactive signing algorithm that is run by a party* P *holding a key pair* $(\mathsf{sk}, \mathsf{pk})$*. Each* P *runs the protocol on session ID sid, its signing key* sk*, public key* pk*, message to be signed* μ*, and a set of co-signers public keys* L *as input. At the end of the protocol* P *obtains a signature* Sig *as output.*
- $\mathsf{Ver}(\mathsf{pp}, \mathsf{Sig}, \mu, L) \to b$ *:The verification algorithm that takes* pp*, a signature, message, and a set of public keys and outputs* $b = 1$ *if the signature is valid. Otherwise b=0.*

– $\mathsf{KVer}(\mathsf{pp}, \mathsf{pk}, \pi) \to b$: *The key verification algorithm that takes as input* pp, pk*, and a proof* π*, and outputs* $b = 1$ *if* pk *is a valid public key. Otherwise b=0.*

Notice that in order to prove the security of our QMS against quantum access adversary, we redefine the multi-signature protocol in a more stronger model, i.e., the key-verification model as in [6, 26]. Compared with the plain public key model, this model additionally ask every participant to prove the knowledge of secret key, i.e., publish a NIZKPoK of the used secret key. In this paper, we just focus on how to design the multi-signature protocol itself, since there are existing efficient multi-proof straight-line extractable NIZKPoK protocols for MSIS in the QROM [19], which can be used in a black-box way.

Due to space limitation, we defer the formal security notions for n-out-of-n signature and multi-signature to Section A.6.

4 Simulation of Quantum Random Oracle

In this section, we consider how to simulate QROM through using Quantum secure PRF (QPRF), such that it can be programable and invertible. Particularly, we notice that the direct QPRF construction in [60], which was first proposed by Banerjee et. al. in [7], can be used to simulate QROM, according to [10, 60]. Thus, the core target of this section is to show that for any efficient quantum adversary conducting superposition queries, the above mentioned direct QPRF construction can be reprogramable and invertible.

Below, we first recall a ring-based variant of concrete construction of QPRF in [7, 60]. Then we define a new "injective mode" for such a QPRF, which is computationally close to the original "normal mode", following from the RLWE assumption. Moreover, for such "injective mode" QPRF, we present an efficient algorithm, which could invert successfully with certain parameter setting. Finally, with the same parameter settings, we show that such QPRF is reprogramable, i.e., any efficient adversary can not distinguish whether the value $\mathsf{QPRF}_{\mathsf{k}}(x)$ has been redefined or not, when x has sufficient entropy. Besides, we add bar symbol for the variables in this section, in order to indicate that the parameters are locally defined and independent of other parts in this paper.

Construction 4.1 (Direct QPRF in [7, 60]) *Let* $\bar{p}, \bar{q}, \bar{d}, \bar{m}, \bar{N}, \bar{\ell}$ *be integers with* $\bar{q} > \bar{p}$*,* $\bar{d} = \lceil \log \bar{q} \rceil$*, and* $\bar{m} = \bar{d}+2$*. Let* $\bar{R} = \mathbb{Z}[X]/(X^{\bar{N}}+1)$ *be a* $2\bar{N}$*-th cyclotomic ring with* $\bar{N}$ *being power of 2 and* $\bar{R}_{\bar{q}} = \bar{R}/\bar{q}\bar{R}$*. Let* χ *be a small distribution over* $\bar{R}$*. We define* $\mathsf{QPRF} : \mathcal{K} \times \{0,1\}^{\bar{\ell}} \to \bar{R}_{\bar{p}}^{1\times\bar{m}}$ *as follows:*

For a key $\mathsf{k} := (\{a_i\}_{i\in[\bar{m}]}, \{s_i\}_{i\in[\bar{\ell}]}) \in \mathcal{K}$ *and input* $x := (x_1, \ldots, x_{\bar{\ell}}) \in \{0,1\}^{\bar{\ell}}$*, let* $\mathsf{QPRF}_{\mathsf{k}}(x) = \mathsf{QPRF}_{\{a_i\},\{s_i\}}(x_1, \ldots, x_{\bar{\ell}}) = \left\lfloor (a_1, \ldots, a_{\bar{m}}) \cdot \prod_{i=1}^{\ell} s_i^{x_i} \right\rceil_p$*, where* $a_i \leftarrow \bar{R}_q$*,* $s_i \leftarrow \chi$*.*

Remark 4.2 *Notice that if* $\bar{q}$ *is chosen such that* $X^{\bar{N}} + 1$ *splits into very few irreducible factors modulus* $\bar{q}$*, and* χ *is concentrated on 'small' elements, then each independent* $s_i \leftarrow \chi$ *is invertible over* $\bar{R}_{\bar{q}}$*, according to Corollary 1.2 in [47].*

For the security of the above Construction 3.1, we have the following theorem.

Theorem 4.3 (Generalization of Theorem 6.1 in [60]) *Let $\chi = D_{\bar{R},\bar{r}}$ be a small distribution over $\bar{R}$, where all coefficients of each polynomial are chosen independently from $D_{\mathbb{Z},\bar{r}}$. Let $\bar{q} \geq \bar{p} \cdot \bar{\ell} \cdot (\bar{r} \cdot \sqrt{2(\bar{N}+\bar{\ell})} \cdot \omega(\sqrt{\log(\bar{N}+\bar{\ell})}))^{\bar{\ell}} \cdot \bar{N}^{\omega(1)}$. Let* QPRF *be as in Construction 4.1. If the* $\mathsf{RLWE}_{\bar{q},1,\bar{m},\chi}$ *holds, then Construction 4.1 is a secure* QPRF.

Generally, the proof idea of this theorem is quite similar to that of Theorem 6.1 in [60], except with the replacement of matrices from $D^{n\times n}_{\mathbb{Z},\bar{r}}$ with ring elements from $\chi = D_{R,\bar{r}}$. In this case, we can still show the security of QPRF through using RLWE. Here, due to space limitation, we defer the detailed proof to Section B.1.

4.1 Inversion of Construction 4.1

In this section, we show that if the vector $\boldsymbol{a} \in \bar{R}^{\bar{m}}_{\bar{q}}$ is generated together with the trapdoor T as in [49] and each $s_i \leftarrow \chi$ is invertible over $\bar{R}_{\bar{q}}$, then QPRF in Construction 4.1 can be inverted efficiently. Basically, this is due to the fact that Construction 4.1 is corresponding to the ring learning with rounding (RLWR) problem, which can be inverted efficiently with the related trapdoor.

Particularly, we have the following formal theorems on the RLWR.

Lemma 4.4 (Trapdoors for RLWR [4,49]) *For any $\bar{N} \geq 1, \bar{q} \geq 2$, $\bar{d} = \lceil \log \bar{q} \rceil$, $\bar{m} = \bar{d} + 2$, $\bar{p} \geq 3 \cdot \sqrt{\bar{m}\bar{N}} \cdot (\sqrt{2\bar{N}} + \sqrt{\bar{d}\bar{N}})$, there exist the following two efficient algorithms* (TrapGen*,* RLWRInvert).
$\mathsf{TrapGen}(1^{\bar{N}}, \bar{q}, \bar{m}, \bar{d})$*: A* PPT *algorithm which on input positive integers $\bar{N}, \bar{q}, \bar{m}, \bar{d}$, first samples a vector $(a_1, a_2) \in \bar{R}^2_{\bar{q}}$ and trapdoor $\mathbf{T} \in S^{2\times\bar{d}}_1$, where $\bar{R}_{\bar{q}} = \mathbb{Z}_{\bar{q}}[X]/(X^{\bar{N}}+1)$. Furthermore, the algorithm computes $(a_3, \ldots, a_{\bar{m}}) = (a_1, a_2)\mathbf{T} + \boldsymbol{g}^\top$, where $\boldsymbol{g}^\top = (1, 2, \ldots, 2^{\bar{d}-1})$. In this case, $\boldsymbol{a}^\top = (a_1, \ldots, a_{\bar{m}})^\top$ is computationally close to uniform over $R^{\bar{m}}_q$, according to the* RLWE *assumption. Clearly, it holds $\boldsymbol{a}^\top \cdot \begin{bmatrix} -\mathbf{T} \\ \mathbf{I}_{\bar{d}\times\bar{d}} \end{bmatrix} = \boldsymbol{g}^\top$, where $\mathbf{I}_{\bar{d}\times\bar{d}} \in \bar{R}^{\bar{d}\times\bar{d}}_{\bar{q}}$ is an identity matrix.*
$\mathsf{RLWRInvert}(\mathbf{T}, \boldsymbol{a}, \boldsymbol{b})$*: An algorithm taking as input $(\boldsymbol{a}, \mathbf{T})$ output by $\mathsf{TrapGen}(1^{\bar{n}}, q)$, and some value $\boldsymbol{b} \in R^{\bar{m}}_{\bar{p}}$ such that $\boldsymbol{b}^\top = \lfloor \boldsymbol{a}^\top \cdot s \rceil_{\bar{p}}$ for some $s \in \bar{R}_{\bar{q}}$, outputs s.*

Due to space limitation, we defer the detailed proof to Section B.1.

Based on the above result in Lemma 4.4, we can define the following *injective mode* for Construction 4.1, which is almost identical to Construction 4.1 except that $\mathbf{A}$ is generated from the algorithm TrapGen.

Construction 4.5 (Injective mode of Construction 4.1) *Let $\bar{p}, \bar{q}, \bar{d}, \bar{m}, \bar{N}, \bar{\ell}$ be integers with $\bar{q} > \bar{p}$, $\bar{d} = \lceil \log \bar{q} \rceil$, and $\bar{m} = \bar{d} + 2$. Let $\bar{R} = \mathbb{Z}[X]/(X^{\bar{N}}+1)$ be a $2\bar{N}$-th cyclotomic ring with $\bar{N}$ being power of 2 and $\bar{R}_{\bar{q}} = \bar{R}/\bar{q}\bar{R}$. Let $\chi = D_{\bar{R},\bar{r}}$ be a small distribution over $\bar{R}$. We define* $\mathsf{QPRF} : \mathcal{K} \times \{0,1\}^{\bar{\ell}} \rightarrow \bar{R}^{1\times\bar{m}}_{\bar{p}}$ *as follows:*

For a key $\mathsf{k} := (\{a_i\}_{i\in[\bar{m}]}, \{s_i\}_{i\in[\bar{\ell}]}) \in \mathcal{K}$ *and input* $x := (x_1, \ldots, x_{\bar{\ell}}) \in \{0,1\}^{\bar{\ell}}$, *let* $\mathsf{QPRF}_\mathsf{k}(x) = \mathsf{QPRF}_{\{a_i\},\{s_i\}}(x_1, \ldots, x_{\bar{\ell}}) = \left\lfloor (a_1, \ldots, a_{\bar{m}}) \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \right\rceil_{\bar{p}}$, *where the vector* $\boldsymbol{a} \in R_{\bar{q}}^{\bar{m}}$ *is generated through running the algorithm* $\mathsf{TrapGen}(1^{\bar{N}}, \bar{q})$, *i.e.,* $(\boldsymbol{a}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^{\bar{N}}, \bar{q})$, *and* $s_i \leftarrow \chi$.

Clearly, for the adversary without the trapdoor matrix $\mathbf{T}$, this injective mode is computationally close to the original *normal mode* in Construction 4.1. Besides, Theorem 4.3 should be still set up in the injective mode, for the adversary without the trapdoor $\mathbf{T}$.

Lemma 4.6 (Indistinguishability of Normal/Injective modes) *For the adversary $\mathcal{A}$ without the trapdoor $\mathbf{T}$ of the vector $\boldsymbol{a}$, if the* $\mathsf{RLWE}_{\bar{q},1,1,S_1}$ *assumption holds, then Constructions 4.1 and 4.5 are computational indistinguishability, even $\mathcal{A}$ queries the functions in a superposition for any polynomial times.*

Due to space limit, we defer the detailed proof to Section B.1.

Below, we describe the concrete invert algorithm for Inj-QPRF in the injective mode.

Algorithm 1: Efficient algorithm $\mathsf{Invert}^{O_{\mathsf{RLWRInvert}}}(\mathbf{T}, \{a_i\}, \{s_i\}, \{b_i\})$ for inverting the function $\mathsf{Inj\text{-}QPRF}_{\{a_i\},\{s_i\}}(x_1, ..., x_{\bar{\ell}})$

Input: An oracle $O_{\mathsf{RLWRInvert}}$ for inverting $\lfloor (a_1, \ldots, a_{\bar{m}}) \cdot s \rceil_{\bar{p}}$, when $\bar{p}$ is large enough.

- PRFKey : vector $\boldsymbol{a} = (a_1, \ldots, a_{\bar{m}})^\top \in \bar{R}_{\bar{q}}^{1\times\bar{m}}$ and $\{s_i\}_{i\in[\bar{\ell}]}$;
- Trapdoor $\mathbf{T} \in \bar{R}^{2\times\bar{d}}$ for $(a_1, \ldots, a_{\bar{m}})$;
- Vector $\boldsymbol{b} = \left\lfloor \boldsymbol{a}^\top \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \right\rceil_{\bar{p}}$ for any $x_i \leftarrow \{0,1\}$.

Output: The vector $x = (x_1, ..., x_{\bar{\ell}}) \in \{0,1\}^{\bar{\ell}}$.

1. Get $s \leftarrow O_{\mathsf{RLWRInvert}}(\mathbf{T}, \boldsymbol{a}, \boldsymbol{b})$, s.t. $\boldsymbol{b} = \lfloor \boldsymbol{a}^\top \cdot s \rceil_{\bar{p}}$, where $s \in \bar{R}_{\bar{q}}$;
2. Set $\hat{s} = s$, if $\|\hat{s}\| \geq r^{\bar{\ell}} \cdot (2\bar{N})^{\bar{\ell}/2}$, return $\perp$;
3. Set $s'_0 = \hat{s}$, for $i = 1, ..., (\bar{\ell} - 1)$, conduct the following steps:
   (i) Compute s_i^{-1}, set $s'_i = s_i^{-1} \cdot s'_{i-1}$, where the computation is conducted over $\bar{R}_{\bar{q}}$.
   (ii) If $\|s'_i\| \leq (\bar{r}\sqrt{2\bar{N}})^{\bar{\ell}-i}$, set $x_i = 1$; Otherwise set $x_i = 0$;
4. Check if $s'_{\bar{\ell}-1} = s_{\bar{\ell}}$, set $x_{\bar{\ell}} = 1$; Otherwise set $x_{\bar{\ell}} = 0$;

return $\boldsymbol{x} = (x_1, ..., x_{\bar{\ell}})$.

Theorem 4.7 *For some $\boldsymbol{a} \in R_{\bar{q}}^{\bar{m}}$ and integers $\bar{p}, \bar{q}, \bar{d}, \bar{N}, \bar{m}$ such that $\bar{q} \geq \bar{p} \cdot \bar{\ell} \cdot (\bar{r} \cdot \sqrt{2(\bar{N}+\bar{\ell})} \cdot \omega(\sqrt{\log(\bar{N}+\bar{\ell})}))^{\bar{\ell}} \cdot \bar{N}^{\omega(1)} \geq \left(\bar{r} \cdot \sqrt{2\bar{N}}\right)^{\bar{\ell}}$, $\bar{d} = \lceil \log \bar{q} \rceil$, and $\bar{m} = \bar{d} + 2$ and $\bar{p} \geq 3 \cdot \sqrt{\bar{m}\bar{N}} \cdot (\sqrt{2\bar{N}} + \sqrt{\bar{d}\bar{N}})$, suppose the oracle* $O_{\mathsf{RLWRInvert}}$ *in Algorithm 1 correctly invert $\lfloor \boldsymbol{a}^\top \cdot s \rceil_{\bar{p}}$ for any $s \in \bar{R}_{\bar{q}}$. Then, for any invertible $s_i \in \bar{R}_{\bar{q}}$, Algorithm 1 correctly inverts* $\mathsf{Inj\text{-}QPRF}_{\boldsymbol{a},\{s_i\}} = \left\lfloor \boldsymbol{a}^\top \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \right\rceil_{\bar{p}}$, *assuming the* $\mathsf{DSPR}_{\bar{q},\bar{R},\chi}$ *assumption.*

Due to space limitation, we defer the detailed proof to Section B.1.

4.2 Adaptive Programming for QPRF in Construction 4.1

In this section, we need to prove that when using the QPRF to simulate QROM, we can conduct adaptive programming similar to the results in [32,56,57], which is needed for the security proof of our two-round threshold signature in the QROM. Particularly, we show that even when conducting quantum queries, an efficient quantum adversary can not distinguish whether the value $\mathsf{QPRF}_{\mathsf{k}}(x)$ in Construction 4.1 has been redefined or not, where x has sufficient entropy.

Overall, the result of this section can be viewed as a generalization of the existing results in [32, 57]. Particularly, in this section, we consider the oracle algorithms $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_c, \mathcal{A}_2)$ essentially access $\mathsf{QPRF}_{\mathsf{k}}(\cdot)$, rather than the random function as in [32, 57]. Moreover, we just consider $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_c, \mathcal{A}_2)$ to be computationally bound adversaries, as QPRF itself is a computational notion.

Theorem 4.8 (QPRF programming, adaptive) *Let* $\mathsf{QPRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *be a quantum secure pseudorandom function for certain sets* $\mathcal{K}, \mathcal{X}, \mathcal{Y}$. *For a random key* $\mathsf{k} \xleftarrow{\$} \mathcal{K}$, *consider the following algorithms:*

- *The oracle algorithm* $\mathcal{A}_1$ *making at most* q *queries to* $\mathsf{QPRF}_{\mathsf{k}}$.
- *The classical algorithm* $\mathcal{A}_c$ *may access the classical part of the final state of* $\mathcal{A}_1$. *Assume that for all initial states, the output of* $\mathcal{A}_c$ *has the collision entropy at least* κ.
- *The oracle algorithm* $\mathcal{A}_2$ *may access the final states of* $\mathcal{A}_1$, *and perform any polynomial times queries to* $\mathsf{QPRF}_{\mathsf{k}}$.

Let

$$P^1_{\mathcal{A}} := \Pr[b' = 1 : \mathcal{A}_1^{|\mathsf{QPRF}_{\mathsf{k}}\rangle}(), x \leftarrow \mathcal{A}_C(), b' = \mathcal{A}_2^{|\mathsf{QPRF}_{\mathsf{k}}\rangle}(x, \mathsf{QPRF}_{\mathsf{k}}(x))]$$
$$P^2_{\mathcal{A}} := \Pr[b' = 1 : \mathcal{A}_1^{|\mathsf{QPRF}_{\mathsf{k}}\rangle}(), x \leftarrow \mathcal{A}_C(), B^* \xleftarrow{\$} \mathcal{Y}, \mathsf{QPRF}_{\mathsf{k}}(x) = B^*, b' = \mathcal{A}_2^{|\mathsf{QPRF}_{\mathsf{k}}\rangle}(x, B^*)]$$

Then

$$\left|P^1_{\mathcal{A}} - P^2_{\mathcal{A}}\right| \le \frac{3}{2}\sqrt{q}2^{\frac{-\kappa}{2}} + 2\varepsilon_{\mathsf{QPRF}}, \tag{4}$$

where $\varepsilon_{\mathsf{QPRF}}$ *is the probability for the efficient quantum adversary to distinguish* QPRF *and random function.*

Proof. For a random function $\mathsf{H} \xleftarrow{\$} (\mathcal{X} \to \mathcal{Y})$, we first define two probabilities $\hat{P}^1_{\mathcal{A}}$ and $\hat{P}^2_{\mathcal{A}}$ as follows:

$$\hat{P}^1_{\mathcal{A}} := \Pr[b' = 1 : \mathsf{H} \xleftarrow{\$} (\mathcal{X} \to \mathcal{Y}), \mathcal{A}_0^{|\mathsf{H}\rangle}(), x \leftarrow \mathcal{A}_C(), b' = \mathcal{A}_1^{|\mathsf{H}\rangle}(x, \mathsf{H}(x))].$$

and

$$\hat{P}^2_{\mathcal{A}} := \Pr\left[b' = 1 : \mathsf{H} \xleftarrow{\$} (\mathcal{X} \to \mathcal{Y}), \mathcal{A}_0^{|\mathsf{H}\rangle}(), x \leftarrow \mathcal{A}_C(), B^* \xleftarrow{\$} \mathcal{Y}, \mathsf{H}(x) = B^*, b' = \mathcal{A}_1^{|\mathsf{H}\rangle}(x, B^*)\right].$$

According to Theorem 6 in [32], it holds $\left|\hat{P}^1_{\mathcal{A}} - \hat{P}^2_{\mathcal{A}}\right| \le \frac{3}{2}\sqrt{q}2^{\frac{-\kappa}{2}}$.

Thus, in order to prove (4), it suffices to prove

$$\left|P^1_{\mathcal{A}} - \hat{P}^1_{\mathcal{A}}\right| \leq \varepsilon_{\mathsf{QPRF}} \text{ and } \left|P^2_{\mathcal{A}} - \hat{P}^2_{\mathcal{A}}\right| \leq \varepsilon_{\mathsf{QPRF}}. \tag{5}$$

Furthermore, we just need to focus on the left-hand side of (5), as the right-hand side of (5) will be set up for the similar argument. Particularly, we could establish the following reduction: suppose there is an efficient quantum adversary $\mathcal{D}$ distinguishing $P^1_{\mathcal{A}}$ and $\hat{P}^1_{\mathcal{A}}$ with probability ε, then we can construct another quantum adversary $\mathcal{B}$ breaking the stronger security of QPRF with probability ε. More precisely, according to Definition 3.7, suppose there is an oracle H^*, the goal of $\mathcal{B}$ is to distinguish $\mathsf{H}^* = \mathsf{QPRF}_{\mathsf{k}}(\cdot)$ or $\mathsf{H}^* \xleftarrow{\$} (\mathcal{X} \to \mathcal{Y})$. Now, $\mathcal{D}$ just needs to answer all $\mathcal{B}$'s queries through further querying H^*, and return the answer of H^* as his answer. Clearly, if $\mathsf{H}^* = \mathsf{QPRF}_{\mathsf{k}}(\cdot)$, then $\mathcal{D}$ is interacting with the case of $P^1_{\mathcal{A}}$; Otherwise, $\mathcal{D}$ is interacting with the case of $\hat{P}^1_{\mathcal{A}}$.

Furthermore, combining with the stronger security of QPRF in Definition 3.7 and Lemma 3.8, we know that $\varepsilon \leq \varepsilon_{\mathsf{QPRF}}$ for all efficient quantum algorithm $\mathcal{B}$, and thus the left-hand side of (5) is set up. So, the right-hand side of (5) is set up too. Summing up all above analysis, for any efficient adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_C, \mathcal{A}_2)$, (4) holds. □

5 Two Round n-out-of-n Threshold Signature from lattices in the QROM

In this section, we present our main construction: two-round n-out-of-n threshold signature, which is provably secure based on MSIS and MLWE in the QROM. Below, we first describe our protocol in Section 5.1, and then prove the correctness and the security of strong unforgeability in Section 5.2. Finally, in Section 5.3, we analyze the efficiency and compare it with other related work.

5.1 Construction

Generally, our protocol can be viewed as enhancing the security of the existing protocol by Damgård et al. in [18] from classical ROM into the QROM, through leveraging the online extractability technology by Unruh in [57] and our modified QPRF in Construction 4.5. Similar to [18], we need to use as a building block an additively homomorphic trapdoor-equivocation commitment scheme Eqv-TCOM with uniform keys, where the trapdoor can be used to equivocate a random commitment to an arbitrary message, according to Definition 3.9. Besides, we also need to use as a building block another type of additively homomorphic trapdoor-inversion commitment scheme Inv-TCOM, where the trapdoor can be used to invert the committed message from the commitment, according to Definition 3.9. Notice that both of above mentioned commitment schemes can be efficiently instantiated by BDLOP commitment in [8] or its variants, just as presented in Section A.7.

Particularly, our construction of two-round threshold n-out-of-n signature $\mathsf{QDS}_2 = (\mathsf{Setup}, (\mathsf{Gen}_u)_{u\in[n]}, (\mathsf{Sign}_u)_{u\in[n]}, \mathsf{Ver})$ is formally specified in Figures 5-7. Here, as in Definition 3.10, all players have the same role, and hence we just

Parameter	Description
n	Number of parties
N	A power of two defining the degree of $f(X)$
$f(X) = X^N + 1$	The $2N$-th cyclotomic polynomial
q	Prime modulus
$R = \mathbb{Z}[X]/(f(X))$	Cyclotomic ring
$R_q = \mathbb{Z}_q[X]/(f(X))$	Ring
k	The height of random matrices $\mathbf{A}$
ℓ	The width of random matrices $\mathbf{A}$
$B = \sigma\sqrt{2N(\ell+k)}$	The upper bound of $\\|\boldsymbol{z}_{i,J_i}^{(u)}\\|$
$B_n = \sqrt{n}B$	The upper bound of $\\|\boldsymbol{z}_{i,J_i}\\|$, with $\boldsymbol{z}_{i,J_i} = \sum_{u=1}^{n} \boldsymbol{z}_{i,J_i}^{(u)}$
$C = \{c \in R : \\|c\\|_\infty = 1 \wedge \\|c\\|_1 = \kappa\}$	Challenge space where $\|C\| = \binom{N}{n}2^\kappa$
$\mathcal{M}$	Message space
κ	The ℓ_1-norm of challenge $c \in C$
$S_\eta = \{x \in R : \\|x\\|_\infty \leq \eta\}$	Set of small secrets
m, t	Iteration parameters for Sign protocol
$T = \kappa\eta\sqrt{m \cdot N(\ell+k)}$	The upper bound of $\\|(c_{i,j}\boldsymbol{s}_n)_{j\in[m]}\\|$
α	Parameter defining σ and M, according to Lemma A.8
$\sigma = \alpha T$	Standard deviation of the Gaussian distribution of $\boldsymbol{y}_i^{(n)}$
$M = \exp\left(\sqrt{\frac{2(\lambda+1)}{\log e}} \cdot \frac{1}{\alpha} + \frac{1}{2\alpha^2}\right)$	The expected number of restarts until Rej output 1.
$\mathsf{cpp}_{\mathsf{Eqv}}, \mathsf{cpp}_{\mathsf{Inv}}$	Public parameters for commitment schemes, honestly generated by Eqv-CSetup and Inv-CSetup
$l_0, l_1, l_1' = k \cdot \ell \cdot N \cdot \log q, l_2, l_5 = t \log m$	Output bit lengths of random oracles $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_1', \mathsf{H}_2, \mathsf{H}_5$
$l_0^* = \log(m \cdot t \cdot \|\mathcal{M}\|) + k \cdot N \cdot \log q \cdot (\ell+1) + \log\|\text{Eqv-}S_{\mathsf{ck}}\| + \log\|\text{Inv-}S_{\mathsf{ck}}\|$	Input bit lengths of random oracles H_0, where Eqv-S_{ck} and Inv-S_{ck} are specified by $\mathsf{cpp}_{\mathsf{Eqv}}$ and $\mathsf{cpp}_{\mathsf{Inv}}$, respectively
l_1^*	Input bit lengths of random oracles $\mathsf{H}_1, \mathsf{H}_1'$
$l_2^* = k \cdot N \cdot \log q + \log n$	Input bit lengths of random oracles H_2
$l_3^* = l_4^* = \log\|\mathcal{M}\| + k \cdot N \cdot \log q \cdot (\ell+1)$	Input bit lengths of random oracles $\mathsf{H}_3, \mathsf{H}_4$
$l_5^* = k \cdot N \cdot \log q \cdot (\ell+1) + \log\|\mathcal{M}\| + t\log\|\text{Eqv-}S_{\mathsf{com}}\| + mt\log(2N\kappa\|\text{Inv-}S_{\mathsf{com}}\|)$	Input bit lengths of random oracles H_5, where Eqv-S_{com} and Inv-S_{com} are specified by $\mathsf{cpp}_{\mathsf{Eqv}}$ and $\mathsf{cpp}_{\mathsf{Inv}}$ respectively.

Table 2. Parameters of Our Two Round n-out-of-n Threshold Signature

Protocol $\mathsf{QDS}_2.\mathsf{Gen}_n(\mathsf{pp})$:
The protocol is parameterized by public parameters described in Table 2 and relies on the random oracles: $\mathsf{H}_1 : \{0,1\}^{l_1^*} \to \{0,1\}^{l_1}$, $\mathsf{H}_1' : \{0,1\}^{l_1^*} \to \{0,1\}^{l_1'}$, $\mathsf{H}_2 : \{0,1\}^{l_2^*} \to \{0,1\}^{l_2}$.

Matrix Generation

1. Sample a random seed $s_n \in \{0,1\}^{l_1^* - \log n}$, and generate a random oracle commitment $g_n \leftarrow \mathsf{H}_1(s_n, n)$. Send out g_n.
2. Upon receiving g_u for all $u \in [n-1]$, send out the seed s_n.
3. Upon receiving s_u for all $u \in [n-1]$:
   (a) If $\mathsf{H}_1(s_u, u) \neq g_u$ for some u, then send out $\perp$.
   (b) Otherwise compute $\mathbf{A}_u = \mathsf{H}_1'(s_u, u)$ for all $u \in [n]$. And set public random matrix $\overline{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$, where $\mathbf{A} := \sum_{u\in[n]} \mathbf{A}_u$.

Key Pair Generation

1. Sample a secret key shares $\boldsymbol{s}_n \xleftarrow{\$} S_\eta^{\ell+k}$ and compute a public key share $\boldsymbol{t}_n := \overline{\mathbf{A}}\boldsymbol{s}_n$, respectively, and generate a random oracle commitment $g_n' \leftarrow \mathsf{H}_2(\boldsymbol{t}_n, n)$. Send out g_n'.
2. Upon receiving g_u' for all $u \in [n-1]$, send out $\boldsymbol{t}_n$.
3. Upon receiving $\boldsymbol{t}_u$ for all $u \in [n-1]$:
   (a) If $\mathsf{H}_2(\boldsymbol{t}_u, u) \neq g_u'$ for some u then send out $\perp$.
   (b) Otherwise set a combined public key $\boldsymbol{t} := \sum_{u\in[n]} \boldsymbol{t}_u$.

If the protocol does not abort, P_n obtain $(\mathsf{sk}_n, \mathsf{pk}) = (\boldsymbol{s}_n, (\mathbf{A}, \boldsymbol{t}))$ as local output.

Fig. 5. Gen Protocol of Our Two-Round n-out-of-n Threshold Signature Scheme

describe the n-th player's behavior. In order to help the readers to understand Figures 5-7 more easily, we go over the high-level ideas for each step as follows.

Parameter setup. According to Definition 3.10, the algorithm $\mathsf{QDS}_2.\mathsf{Setup}$ should be invoked by a trusted party, and outputs a set of public parameters as in Table 2. Notice that most of our parameters follow from those of [18], except with the following case:

- As we want to generalize the framework of Unruh in [57] into the threshold setting, it is necessary to replace the random oracle for hashing the signatures of Dilithium-G as an additively homomorphic trapdoor-inversion commitment scheme. Thus, we need to run the algorithm $\mathsf{Inv\text{-}TCOM.CSetup}(1^\lambda)$ to generate an additional public parameter $\mathsf{cpp_{Inv}}$ for Inv-TCOM. Besides, for the reason of security proof in Lemma C.2, we require Inv-TCOM satisfies the binding property too. And, we can set suitable parameters such that the binding of Inv-TCOM is statistical, which is necessary for security proof in Lemma C.3.

Key generation. The key generation algorithm $\mathsf{QDS_2.Gen}$ almost follows that of [18], except that we introduce another random oracle H_1' as the randomness generator. Particulary, in order to interactively generate a random matrix $\mathbf{A} \in R_q^{k\times\ell}$ in a secure way, the n-th participant employs the following random oracle commitments: first choose his random seed $s_n \xleftarrow{\$} \{0,1\}^{l_1^*}$, then compute and send out $g_n \leftarrow \mathsf{H}_1(s_n, n)$. Then with s_u for all $u \in [n]$, any one can generate the random matrix $\mathbf{A}_u \xleftarrow{\$} H_1'(s_u, u)$. Due to the uniform and random distribution of s_n, the input of H_1' has sufficient entropy, thus we can reprogram H_1' in the security proof. Notice that in this case, the participants just need to send out the seed $s_u \in \{0,1\}^{l_2^*}$, rather than $\mathbf{A}_u \in R_q^{k\times\ell}$, in the public channel. Clearly, this will significantly reduce the communication overhead of our construction.

Similarly, the n-th participant directly utilize H_2 to generate random oracle commitment g_n'.

Signature generation. One important point for the $\mathsf{QDS_2.Sign}_n$ algorithm in Figure 6 is the iterations at (1.a) of **Signature generation**. With these steps, we can realize online extractability, according to [57]. And thus, we can circumvent the essential obstacle, rewinding, for the security proof of signature in the QROM.

The other one crucial point is the computation of $c_{i,j}$, i.e., $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{pk}, \mathsf{ck}, \mathsf{ck}')$. In fact, this step has at least two significance:

- For fixed i and different j and j', $c_{i,j} \neq c_{i,j'}$. This is necessary for successful extractability through using the extractor Ext presented in Figure 12, according to [57].
- The computation of $c_{i,j}$ does not rely on $\mathsf{com}_i^{(u)}$ or $\boldsymbol{w}_i^{(u)}$. And thus, for each $(i, j) \in [t] \times [m]$, all participants will use the same challenge $c_{i,j}$ for the related individual running of underlying underlying Dilithium-G signature scheme. Clearly, only with such condition, $\{\boldsymbol{z}_{i,J_i}\}_{i\in[t]}$ in the final signature can be verified successfully with respect to public key $(\mathbf{A}, \boldsymbol{t})$, according to the step (2.c) of the algorithm $\mathsf{QDS_2.Ver}$ in Figure 7.

Verification. Thanks to the linearity of the underlying Dilithium-G signature scheme, and additive homomorphism of Eqv-TCOM and Inv-TCOM with respect to both message and randomness, the verifier just need to verify the sum

Protocol $\mathsf{QDS}_2.\mathsf{Sign}_n(sid, \mathsf{sk}_n, \mathsf{pk}, \mu)$

The protocol is parameterized by public parameters described in Table 2 and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^{l_0^*} \to C$, $\mathsf{H}_3 : \{0,1\}^{l_3^*} \to \mathsf{Eqv}\text{-}S_{\mathsf{ck}}, \mathsf{H}_4 : \{0,1\}^{l_4^*} \to \mathsf{Inv}\text{-}S_{\mathsf{ck}}$ and $\mathsf{H}_5 : \{0,1\}^{l_5^*} \to \{0,1\}^{l_5}$. The protocol assumes that $\mathsf{QDS}_2.\mathsf{Gen}_n(\mathsf{pp})$ has been previously invoked.

Inputs

1. P_n receives a unique sessions ID $sid, \mathsf{sk}_n = \boldsymbol{s}_n, \mathsf{pk} = (\mathbf{A}, \boldsymbol{t})$ and message $\mu \in M$ as input.
2. P_n verifies that sid has not been used before (if it has been, the protocol is not executed).
3. P_n locally computes per-message commitment keys $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk}), \mathsf{ck}' \leftarrow \mathsf{H}_4(\mu, \mathsf{pk})$.

Signature Generation P_n works as follows:

1. Compute the first group messages as follows:
   (a) for $i = 1$ to t; conduct as follows:
      i. Sample $\boldsymbol{y}_i^{(n)} \leftarrow D_\sigma^{\ell+k}$ and compute $\boldsymbol{w}_i^{(n)} := \overline{\mathbf{A}} y_i^{(n)}$.
      ii. Compute $\mathsf{com}_i^{(n)} \leftarrow \mathsf{Eqv}\text{-}\mathsf{Commit}_{\mathsf{ck}}(\boldsymbol{w}_i^{(n)}, r_i^{(n)})$ with $r_i^{(n)} \xleftarrow{\$} \mathsf{Eqv}\text{-}S_r$.
      iii. for $j = 1$ to m; conduct as follows:
         A. Derive challenges $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{pk}, \mathsf{ck}, \mathsf{ck}')$.
         B. Compute signature shares $\boldsymbol{z}_{i,j}^{(n)} = c_{i,j}\boldsymbol{s}_n + \boldsymbol{y}_i^{(n)}$.
         C. Run the rejection sampling $\mathsf{Rej}(\boldsymbol{z}_{i,j}^{(n)}, c_{i,j}\boldsymbol{s}_n, \sigma) \to \{0,1\}$.
      iv. If the above rejection sampling algorithm outputs 0 for certain $j \in [m]$, then go to the Step i.
   (b) Compute $\widetilde{\mathsf{com}}_{i,j}^{(n)} \leftarrow \mathsf{Inv}\text{-}\mathsf{Commit}_{\mathsf{ck}'}(\boldsymbol{z}_{i,j}^{(n)}, r_{i,j}'^{(n)})$ where $r_{i,j}'^{(n)} \xleftarrow{\$} \mathsf{Inv}\text{-}S_r$ for all $i \in [t], j \in [m]$.
   (c) Send out $(\{\mathsf{com}_i^{(n)}\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^{(n)}\}_{i\in[t],j\in[m]})$.
2. Upon receiving $(\{\mathsf{com}_i^{(u)}\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^{(u)}\}_{i\in[t],j\in[m]})$ for all $u \in [n-1]$, compute the signature shares as follows:
   (a) Set $\mathsf{com}_i := \sum_{u\in[n]} \mathsf{com}_i^{(u)}$ and $\widetilde{\mathsf{com}}_{i,j} := \sum_{u\in[n]} \widetilde{\mathsf{com}}_{i,j}^{(u)}$ for all $i \in [t], j \in [m]$.
   (b) Get challenges $J_1||...||J_t \leftarrow \mathsf{H}_5(\mathsf{pk}, \mu, \{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]})$.
   (c) Send out $(\{\boldsymbol{z}_{i,J_i}^{(n)}\}_{i\in[t]}, \{r_i^{(n)}\}_{i\in[t]}, \{r_{i,J_i}'^{(n)}\}_{i\in[t]})$.
3. Upon receiving $(\{\boldsymbol{z}_{i,J_i}^{(u)}\}_{i\in[t]}, \{r_i^{(u)}\}_{i\in[t]}, \{r_{i,J_i}'^{(u)}\}_{i\in[t]})$ for all $u \in [n]$ compute the combined signature as follows:
   (a) For each $u \in [n-1]$, compute J_i and c_{i,J_i} as before, and reconstruct $\boldsymbol{w}_i^{(u)} := \overline{\mathbf{A}}\boldsymbol{z}_{i,J_i}^{(u)} - c_{i,J_i}\boldsymbol{t}_u$, then validate the signature shares

$$||\boldsymbol{z}_{i,J_i}^{(u)}|| \leq B, \mathsf{Eqv}\text{-}\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}_i^{(u)}, r_i^{(u)}, \boldsymbol{w}_i^{(u)}) = 1$$

   and

$$\mathsf{Inv}\text{-}\mathsf{Open}_{\mathsf{ck}'}(\widetilde{\mathsf{com}}_{i,J_i}^{(u)}, r_{i,J_i}'^{(u)}, \boldsymbol{z}_{i,J_i}^{(u)}) = 1.$$

   for all $i \in [t]$. If the check fails for some u then send out $\perp$.
   (b) Compute $\boldsymbol{z}_{i,J_i} := \sum_{u\in[n]} \boldsymbol{z}_{i,J_i}^{(u)}$, $r_i := \sum_{u\in[n]} r_i^{(u)}$ and $r_{i,J_i}' := \sum_{u\in[n]} r_{i,J_i}'^{(u)}$ for all $i \in [t]$.

If the protocol does not abort, P_n obtains a signature:
$\mathsf{Sig} := (\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_{i,J_i}\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r_{i,J_i}'\}_{i\in[t]})$ as local output.

Fig. 6. Sign Protocol of Our Two-Round n-out-of-n Threshold Signature Scheme

of signature shares, i.e., $\sum_{u=1}^n \mathsf{Sig}^{(u)}$, where each signature share $\mathsf{Sig}^{(u)}$ consists of commitments $\left(\{\mathsf{com}_i^{(u)}\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^{(u)}\}_{i\in[t],j\in[m]}\right)$, underlying responses $\{\boldsymbol{z}_{i,J_i}^{(u)}\}_{i\in[t]}$ and randomness $\left(\{r_i^{(u)}\}_{i\in[t]}, \{r_{i,J_i}'^{(u)}\}_{i\in[t]}\right)$.

5.2 Correctness and Security

Theorem 5.1 (Correctness) *For public parameters as in Table 2, two-round threshold n-out-of-n signature* $\mathsf{QDS}_2 = (\mathsf{Setup}, (\mathsf{Gen}_u)_{u\in[n]}, (\mathsf{Sign}_u)_{u\in[n]}, \mathsf{Ver})$ *in Figures 5, 6, 7 satisfies the correctness. In other word, suppose the underlying Dilithium scheme is correct, and the trapdoor commitment schemes* Inv-TCOM *and* Eqv-TCOM *are correct and additively homomorphic, then a valid generated* *signatures must be accepted by the verification algorithm, except with a negligible probability.*

Proof. Notice that the algorithm Ver in Figure 7 needs to conduct four checks. Thus, below we will discuss them one by one.

1. Due to the collision resistance of random oracle, it will output different values for different inputs, except with a negligible probability. Thus, for $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{pk}, \mathsf{ck}, \mathsf{ck}')$, all $c_{i,1}, ..., c_{i,m}$ are pairwise distinct except with the probability $m \cdot \mathsf{negl}(\lambda)$. Clearly, this is still negligible in λ, when m is a polynomial.
2. For $\boldsymbol{y}_i^{(n)} \leftarrow D_\sigma^{\ell+k}$, $\boldsymbol{z}_{i,j}^{(n)} = c_{i,j}\boldsymbol{s}_n + \boldsymbol{y}_i^{(n)}$ and $\mathsf{Rej}(\boldsymbol{z}_{i,j}^{(n)}, c_{i,j}\boldsymbol{s}_n, \sigma) \rightarrow 1$, we know that the distribution of $\boldsymbol{z}_{i,J_i}^{(n)}$ is statistically close to $D_\sigma^{(\ell+k)}$, according to Lemma A.8. Thus, we have $\|\boldsymbol{z}_{i,j}^{(n)}\| \leq \sigma\sqrt{2 \cdot (\ell + k) \cdot N} = B$ with overwhelming probability, according to Lemma A.3. Furthermore, according to Lemma A.4, we know that the distribution of $\boldsymbol{z}_{i,J_i} := \sum_{u\in[n]} \boldsymbol{z}_{i,J_i}^{(u)}$ is statistically close to $D_{\sigma\sqrt{n}}^{(\ell+k)}$. And thus, it holds $\|\boldsymbol{z}_{i,J_i}\| \leq \sigma\sqrt{2 \cdot n \cdot (\ell + k) \cdot N} = B_n$, except with a negligible probability.
3. Due to the correctness of the underlying Dilithium-G scheme, we know that $\overline{\mathbf{A}}\boldsymbol{z}_{i,J_i}^{(u)} = \boldsymbol{w}_i^{(u)} + c_{i,J_i}\boldsymbol{t}_u$ for each honest participant P_u with $u \in [n]$. Then, it holds $\overline{\mathbf{A}}\boldsymbol{z}_{i,J_i} = \boldsymbol{w}_i + c_{i,J_i}\boldsymbol{t}$, according to $\boldsymbol{w}_i = \sum \boldsymbol{w}_i^{(u)}$, $\boldsymbol{z}_i = \sum \boldsymbol{z}_i^{(u)}$, and $\boldsymbol{t} = \sum \boldsymbol{t}_u$ from the Sign and Gen protocols in Figures 6 and 5. Then, Verifier can reconstruct $\boldsymbol{w_i} := \overline{\mathbf{A}}\boldsymbol{z}_{i,J_i} - c_{i,J_i}\boldsymbol{t}$. And according to the homomorphic property and correctness of the used $\mathsf{Eqv\text{-}TCOM}$, it holds $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}}(\mathsf{com}_i, r_i, \overline{\mathbf{A}}\boldsymbol{z}_i - c_{i,J_i}\boldsymbol{t}) = 1$, except with a negligible probability.
4. According to the homomorphic property and correctness of the used $\mathsf{Inv\text{-}TCOM}$, for all honestly generated signatures, it holds $\mathsf{Inv\text{-}Open}_{\mathsf{ck}'}(\widetilde{\mathsf{com}}_{i,J_i}, r'_{i,J_i}, \boldsymbol{z}_i) = 1$, except with a negligible probability.

Summing up all above analysis, the honestly generated signatures should be accepted, except with at most a negligible probability. ◻

Below, we focus on the security of our QDS_2 construction. Just as analysis in Remark A.14, we know that for QDS_2, the $\mathsf{SUF\text{-}CMA}$ security implies the $\mathsf{UF\text{-}CMA}$ security. Thus, in the following theorem, we just focus on the much stronger one, $\mathsf{SUF\text{-}CMA}$ security.

Theorem 5.2 *Suppose the trapdoor commitment schemes* $\mathsf{Inv\text{-}TCOM}$ *and* $\mathsf{Eqv\text{-}TCOM}$ *are secure, additively homomorphic, have uniform keys and uniform commitment. Particularly, the output of* $\mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ *has sufficient min-entropy* ϑ*. And suppose there exists* QPRF *that can be programable and invertible simultaneously. For any quantum polynomial-time adversary* $\mathcal{A}$ *that initiates a single key generation protocol by querying* $\mathcal{O}_n^{\mathsf{QDS}_2}$ *with sid* $= 0$*, initiates* Q_s *signature generation protocols by querying* $\mathcal{O}_n^{\mathsf{QDS}_2}$ *with sid* $\neq 0$*, and makes* Q_h *quantum superpositions queries to random oracle* $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}'_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5$*, the protocol* QDS_2 *of Figures 5, 6, 7 is* $\mathsf{QDS\text{-}SUF\text{-}CMA}$ *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions in the* QROM*, where* $\beta = 2\sqrt{B_n^2 + \kappa}$*. Concretely,*

Algorithm $\mathsf{QDS}_2.\mathsf{Ver}((\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r'_{i,J_i}\}_{i\in[t]}), \mu, \mathsf{pk})$
Upon receiving a message μ, signature $(\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r'_{i,J_i}\}_{i\in[t]})$, and combined public key $\mathsf{pk} := (\mathbf{A}, \boldsymbol{t})$ works as follows:

1. Generate commitment keys $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$, $\mathsf{ck}' \leftarrow \mathsf{H}_4(\mu, \mathsf{pk})$, derive $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{pk}, \mathsf{ck}, \mathsf{ck}')$ for all $i \in [t], j \in [m]$ and compute $J_1||...||J_t \leftarrow \mathsf{H}_5(\mathsf{pk}, \mu, \{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]})$.
2. Perform the checks as follows:
   (a) for $i = 1$ to t do:
   Check that $c_{i,1}, ..., c_{i,m}$ pairwise distinct.
   (b) for $i = 1$ to t do:
   Check that $||\boldsymbol{z}_i|| \leq B_n$.
   (c) for $i = 1$ to t do:
   Reconstruct $\boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_i - c_{i,J_i}\boldsymbol{t}$, and check $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}}(\mathsf{com}_i, r_i, \boldsymbol{w}_i) = 1$.
   (d) for $i = 1$ to t do:
   Check $\mathsf{Inv\text{-}Open}_{\mathsf{ck}'}(\widetilde{\mathsf{com}}_{i,J_i}, r'_{i,J_i}, \boldsymbol{z}_i) = 1$.

If all checks succeed then return 1, otherwise, return 0.

Fig. 7. Ver Algorithm of Our Two-Round n-out-of-n Threshold Signature Scheme

using other parameters specified in Table 2, the advantage of $\mathcal{A}$ is bounded as follows.

$$\begin{aligned}\mathbf{Adv}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}_{\mathsf{QDS}_2}(\mathcal{A}) \leq\ & 2\varepsilon_{\mathsf{Inj\text{-}QPRF}} + 5\varepsilon_{\mathsf{QPRF}} + e(Q_h + Q_s + 1)\Big[(Q_h + Q_s)(\varepsilon_{\mathsf{td}} + \varepsilon_{\mathsf{td}'}) \\ & + 2(Q_h + Q_s)\cdot\varepsilon_{\mathsf{QPRF}} + \frac{3}{2}\sqrt{Q_h}2^{\frac{-t\cdot\vartheta}{2}} + 2\varepsilon_{\mathsf{QPRF}} + t\cdot Q_s\cdot(m-1)\cdot\mathsf{negl}(\lambda) \\ & + t\cdot Q_s\cdot\varepsilon_{\mathsf{Rej}} + \frac{3}{2}\sqrt{Q_h}(2^{\frac{-q^{klN}}{2}} + 2^{\frac{-q^{kN}}{2}}) + 4(\varepsilon_{\mathsf{QPRF}} + \varepsilon_{\mathsf{Inj\text{-}QPRF}}) \\ & + \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}} + 2(Q_h+1)2^{-(t\log m)/2} + Q_s\cdot t\cdot\varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2}\cdot 2^{-t\cdot\vartheta} \\ & + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}\Big]\end{aligned}$$

Here, $\varepsilon_{\mathsf{QPRF}}$ denotes the advantage for an efficient quantum adversary distinguishing QROM *and* QPRF *in Construction 4.1. $\varepsilon_{\mathsf{Inj\text{-}QPRF}}$ denotes the advantage distinguishing injective* QPRF *in Construction 4.5 from the direct Construction 4.1. $\varepsilon_{\mathsf{td}}$ (or $\varepsilon_{\mathsf{td}'}$) is the statistical distances of true commitment key (or trapdoor commitment key) for* Eqv-TCOM *(or* Inv-TCOM*) and the uniform. $\varepsilon_{\mathsf{Rej}}$ is the statistical distances of the output distribution of rejection sampling algorithm and the ideal distribution. ε'_{bind} is the advantages of breaking* Inv-TCOM *for any efficient quantum adversary. Moreover, all these values are negligible according to the related instantiations in this paper.*

Below, we first sketch the proof idea, before presenting the formal proof. According to Definition A.12, we need to prove that for any efficient adversary $\mathcal{A}$ against QDS_2, its advantage $\mathbf{Adv}^{\mathsf{QDS\text{-}UF\text{-}CMA}}_{\mathsf{QDS}_2}(\mathcal{A})$ is negligible. In order to do this, we conduct the following two steps:

- We first show that the party P_n in the experiment $\mathbf{Adv}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}_{\mathsf{QDS}_2}(\mathcal{A})$ can be simulated by a simulator $\mathcal{B}$ defined in Figure 11, together with its subroutines Figures 13 to 16. And $\mathcal{B}$ do not have any secret key, through using a sequence of hybrid experiments. Particularly, in the key generation and signature query phases, we use the QPRF to simulate the quantum random oracle, which satisfy the requirements of extraction and reprogrammability. In the signature query phase, we use the trapdoor-equivocation commitment scheme and the adaptive programming of H_5 to simulate the signature.

- Then, we show that in such a simulated experiment, the signature is strong unforgeability, through establishing a reduction from MSIS and the binding properties of Inv-TCOM, following from the similar proof idea of [57]. Particularly, we first show that there is an efficient extractor Ext in Figure 12, such that given a valid forged message-signature pair $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$, Ext can output a solution for MSIS problem, if the used Inv-TCOM scheme satisfies the binding property. And then, we bound the probability of generating a valid forged message-signature pair $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$ by the union bound of two events happen: Ext succeeds and Ext fails.

Due to space limitation, we defer the detailed proof of this theorem to Section C.

5.3 Asymptotical Efficiency and Comparison with [18]

In this paper, we focus on constructing the first asymptotically efficient n-out-of-n and multi-signature protocols in the QROM, rather than striving for concrete efficiency. Thus, in this section, we just analyze the asymptotical efficiency of our protocol in Section 5.1, and then compare it with [18].

In order to take advantage of our parallelizable property, we would like to set $m = 2$ and $t = \lambda$, which will ensure the domain of $(J_1, \ldots, J_t)$ is large enough. Similar to the optimization in [18], we can replace $(\mathsf{com}_i, r_i, \boldsymbol{z}_{i,J_i})$ with $(c_{i,J_i}, r_i, \boldsymbol{z}_{i,J})$. Even in our case, c_{i,J_i} can be omitted, due to its computation process. Thus, the final signature size for each party is about $\lambda \cdot (|r_i| + (\ell + k) \cdot N \log(12\sigma) + |r_i'| + 2|\widetilde{\mathsf{com}_{i,j}}|)$.

In order to ensure a relatively fair comparison, we should enhance the protocol in [18] as follows: (i) enlarge the standard deviation σ about n times, when dealing with all n parties. In this case, we can ensure the whole expected abort time is about $1/M$, rather than $1/M^n$. (ii) run $\tau = \lambda/(\log \frac{M}{M-1})$ parallel executions simultaneously. In this case, we can ensure that the parties output a signature with overwhelming probability, after two round interactions. Thus, the final signature size for each party is about $\lambda(|c_{i,j}| + |r_i| + (\ell + k) \cdot N \cdot \log(12n\sigma))$.

Clearly, the main additional overheads of our construction are the size of $|r_i'| + 2|\widetilde{\mathsf{com}_{i,j}}|$. However, further considering the reduction loss for the underlying MSIS problem, the protocol in [18] need to use much larger parameters to compensate such security loss. Overall, conditioned on our QROM security, we believe that such slightly more overheads on signature size are completely acceptable.

6 Two Round Multi-Signature from lattices in the QROM

We can construct a multi-signature scheme QMS_2 in the QROM through using the similar processes for QDS_2 in Section 5, besides with an additional multi-proof straight-line extractable NIZKPoK system in the QROM in the key generation algorithm. Such QMS_2 supports key aggregation and can be proven secure relying on essentially the same idea as QDS_2. The main difference from QDS_2 is that, the protocol requires no interactive key generation at all, and instead for each signing execution a party receives a set of public keys Ł together with a message to be signed. Particularly, our construction of two-round multi-signature $\mathsf{QMS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{KVer})$ is formally specified in Figures 17, 18, 19. Due to space limit, we defer to Section D the detailed presentations of our multi-signature construction together with the related security proof.

References

1. M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. Tightly secure signatures from lossy identification schemes. *Journal of Cryptology*, 29(3):597–631, July 2016.
2. S. Agrawal, D. Stehlé, and A. Yadav. Round-optimal lattice-based threshold signatures, revisited. In M. Bojanczyk, E. Merelli, and D. P. Woodruff, editors, *ICALP 2022*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl, July 2022.
3. H. K. Alper and J. Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 157–188, Virtual Event, Aug. 2021. Springer, Cham.
4. J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs. Learning with rounding, revisited - new reduction, properties and applications. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 57–74. Springer, Berlin, Heidelberg, Aug. 2013.
5. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 595–618. Springer, Berlin, Heidelberg, Aug. 2009.
6. A. Bagherzandi, J. H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, Oct. 2008.
7. A. Banerjee, C. Peikert, and A. Rosen. Pseudorandom functions and lattices. In Pointcheval and Johansson [55], pages 719–737.
8. C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert. More efficient commitments from structured lattice assumptions. In D. Catalano and R. De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385. Springer, Cham, Sept. 2018.
9. A. Boldyreva and D. Micciancio, editors. *CRYPTO 2019, Part II*, volume 11693 of *LNCS*. Springer, Cham, Aug. 2019.
10. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Berlin, Heidelberg, Dec. 2011.
11. D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, Aug. 2018.

12. D. Boneh, S. Kim, and D. J. Wu. Constrained keys for invertible pseudorandom functions. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 237–263. Springer, Cham, Nov. 2017.
13. C. Boschini, A. Takahashi, and M. Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Dodis and Shrimpton [21], pages 276–305.
14. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 266–296. Springer, Cham, May 2020.
15. Y. Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In H. Handschuh and A. Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 716–747. Springer, Cham, Aug. 2023.
16. K.-M. Chung, S. Fehr, Y.-H. Huang, and T.-N. Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work. In A. Canteaut and F.-X. Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 598–629. Springer, Cham, Oct. 2021.
17. I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In J. Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Cham, May 2021.
18. I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, Apr. 2022.
19. R. del Pino and S. Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Dodis and Shrimpton [21], pages 306–336.
20. Y. Desmedt. Threshold cryptosystems. In *International Workshop on the Theory and Application of Cryptographic Techniques*, pages 1–14. Springer, 1992.
21. Y. Dodis and T. Shrimpton, editors. *CRYPTO 2022, Part II*, volume 13508 of *LNCS*. Springer, Cham, Aug. 2022.
22. J. Doerner, Y. Kondi, E. Lee, and a. shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy*, pages 1051–1066. IEEE Computer Society Press, May 2019.
23. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Boldyreva and Micciancio [9], pages 356–383.
24. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Efficient NIZKs and signatures from commit-and-open protocols in the QROM. In Dodis and Shrimpton [21], pages 729–757.
25. J. Don, S. Fehr, C. Majenz, and C. Schaffner. Online-extractability in the quantum random-oracle model. In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 677–706. Springer, Cham, May / June 2022.
26. M. Drijvers, K. Edalatnejad, B. Ford, E. Kiltz, J. Loss, G. Neven, and I. Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1084–1101. IEEE, 2019.
27. L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

28. L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS – Dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017.
29. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 335–352. Springer, Berlin, Heidelberg, Aug. 2014.
30. M. Fukumitsu and S. Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model. In K. Nguyen, W. Wu, K.-Y. Lam, and H. Wang, editors, *ProvSec 2020*, volume 12505 of *LNCS*, pages 45–64. Springer, Cham, Nov. / Dec. 2020.
31. N. Genise and D. Micciancio. Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In Nielsen and Rijmen [51], pages 174–203.
32. A. B. Grilo, K. Hövelmanns, A. Hülsing, and C. Majenz. Tight adaptive reprogramming in the QROM. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 637–667. Springer, Cham, Dec. 2021.
33. K. D. Gur, J. Katz, and T. Silde. Two-round threshold lattice signatures from threshold homomorphic encryption. *Cryptology ePrint Archive*, 2023.
34. A. Hoover, S. Patel, G. Persiano, and K. Yeo. Plinko: Single-server PIR with efficient updates via invertible PRFs. Cryptology ePrint Archive, Report 2024/318, 2024.
35. K. Itakura. A public-key cryptosystem suitable for digital multisignature. *NEC research and development*, 71:1–8, 1983.
36. K. A. Jackson, C. A. Miller, and D. Wang. Evaluating the security of CRYSTALS-dilithium in the quantum random oracle model. In M. Joye and G. Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 418–446. Springer, Cham, May 2024.
37. S. Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 580–610, Virtual Event, Aug. 2021. Springer, Cham.
38. E. Kiltz, V. Lyubashevsky, and C. Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Cham, Apr. / May 2018.
39. C. Komlo and I. Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In O. Dunkelman, M. J. J. Jr., and C. O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 34–65. Springer, Cham, Oct. 2020.
40. A. Langlois and D. Stehle. Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography, 2015.
41. Y. Lindell. Simple three-round multiparty schnorr signing with full simulatability. *Cryptology ePrint Archive*, 2022.
42. Q. Liu and M. Zhandry. Revisiting post-quantum Fiat-Shamir. In Boldyreva and Micciancio [9], pages 326–355.
43. A. López-Alt, E. Tromer, and V. Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In H. J. Karloff and T. Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.
44. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, Dec. 2009.
45. V. Lyubashevsky. Lattice signatures without trapdoors. In Pointcheval and Johansson [55], pages 738–755.

46. V. Lyubashevsky, N. K. Nguyen, M. Plançon, and G. Seiler. Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In M. Tibouchi and H. Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 218–248. Springer, Cham, Dec. 2021.
47. V. Lyubashevsky and G. Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In Nielsen and Rijmen [51], pages 204–224.
48. G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Report 2018/068, 2018.
49. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In Pointcheval and Johansson [55], pages 700–718.
50. J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*, pages 1717–1731. ACM Press, Nov. 2020.
51. J. B. Nielsen and V. Rijmen, editors. *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*. Springer, Cham, Apr. / May 2018.
52. NIST. Post-quantum cryptography project.
53. J. Pan and B. Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In C. Hazay and M. Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 597–627. Springer, Cham, Apr. 2023.
54. J. Pan and B. Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. In M. Joye and G. Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 460–489. Springer, Cham, May 2024.
55. D. Pointcheval and T. Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*. Springer, Berlin, Heidelberg, Apr. 2012.
56. D. Unruh. Quantum position verification in the random oracle model. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, Aug. 2014.
57. D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Berlin, Heidelberg, Apr. 2015.
58. H. Xue, M. H. Au, M. Liu, K. Y. Chan, H. Cui, X. Xie, T. H. Yuen, and C. Zhang. Efficient multiplicative-to-additive function from joye-libert cryptosystem and its application to threshold ecdsa. In *CCS 2023*, pages 2974–2988, 2023.
59. H. Xue, M. H. Au, X. Xie, T. H. Yuen, and H. Cui. Efficient online-friendly two-party ECDSA signature. In G. Vigna and E. Shi, editors, *ACM CCS 2021*, pages 558–573. ACM Press, Nov. 2021.
60. M. Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, Oct. 2012.
61. M. Zhandry. Secure identity-based encryption in the quantum random oracle model. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 758–775. Springer, Berlin, Heidelberg, Aug. 2012.
62. M. Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Boldyreva and Micciancio [9], pages 239–268.

A Supplementary for Preliminaries

Due to the space limitation in the main body, we present many more supplementary materials for Preliminaries in Section 3

A.1 Notations

In this paper, $\mathbb{Z}$ and $\mathbb{R}$ denote the sets of integers and real numbers. For positive integers n, q, let $[n]$ denotes the set $\{1, ..., n\}$ and $\mathbb{Z}_q$ denotes the ring of integers modulo q. We use λ to denote the security parameter, which is the implicit input for all algorithms presented in this paper. A function $f(\lambda) > 0$ is negligible and denoted by $\mathsf{negl}(\lambda)$ if for any $c > 0$ and sufficiently large λ, $f(\lambda) < 1/\lambda^c$. A probability is called to be overwhelming if it is $1 - \mathsf{negl}(\lambda)$. A column vector is denoted by a bold lower case letter (e.g., $\boldsymbol{x}$). A matrix is denoted by a bold upper case letter (e.g., $\mathbf{A}$), and its transposition is denoted by $\mathbf{A}^\top$. Let $R = \mathbb{Z}[x]/(x^N + 1)$ be a cyclotomic ring, with N be a power of 2. The norm of an element in $R_q = \mathbb{Z}_q[x]/(x^N+1)$ will be the norm of its unique representative with coefficients in $[-(q-1)/2, (q-1)/2]$. For positive $\beta \in \mathbb{R}$, we use S_β to denote the set of all polynomials of infinity norm less than β, i.e., $S_\beta = \{a \in R \mid \|a\|_\infty \leq \beta\}$.

We define a rounding function $\lfloor\cdot\rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ for $q \geq p \geq 2$ as $\lfloor x\rceil_{q\to p} = \lfloor (p/q)\bar{x}\rceil_{q\to p}$, where $\bar{x} \in \mathbb{Z}$ is any integer congruent to x mod q. Furthermore, $\lfloor\cdot\rceil_{q\to p}$ can be extended component-wise to vectors and matrices over $\mathbb{Z}_q$. Especially, for a ring element $a \in R$ represented as coefficient embedding, we first view it as the vector consisting of all its coefficients, and then conduct rounding function to such vector. In places where the context is clear about the modulus q, we would omit q in the notation as $\lfloor\cdot\rceil_p$ for simplicity of presentation.

For a distribution or a set D, we write $x \xleftarrow{\$} D$ to denote the operation of sampling an uniformly random x according to D. We denote as $\mathsf{Supp}(D)$ the support of a distribution D. For two distributions D_1, D_2, we let $\mathsf{SD}(D_1, D_2)$ denote their statistical distance. We write $D_1 \overset{s}{\approx} D_2$ to mean that they are statistically close, and $D_1 \overset{c}{\approx} D_2$ to say that they are computationally indistinguishable. The collision entropy of a random variable X is $-\log \Pr[X = X']$ where X' is independent of X and has the same distribution. The min-entropy of X is $\min_x(-\log \Pr[X = x])$.

Matrix norms. For a vector $\boldsymbol{x}$, its Euclidean norm (also known as the ℓ_2 norm) is defined as $\|\boldsymbol{x}\| = (\sum_i x_i^2)^{1/2}$. For a matrix $\mathbf{R}$, we denote its i-th column vector as $\boldsymbol{r}_i$, and use $\widetilde{\mathbf{R}}$ to denote its Gram-Schmidt orthogonalization. In addition,

- $\|\mathbf{R}\|$ denotes the Euclidean norm of $\mathbf{R}$, i.e., $\|\mathbf{R}\| = \max_i \|\boldsymbol{r}_i\|$.
- $s_1(\mathbf{R})$ denotes the spectral norm of $\mathbf{R}$, i.e., $s_1(\mathbf{R}) = \sup_{\|\boldsymbol{x}\|=1} \|\mathbf{R}\boldsymbol{x}\|$, with $\boldsymbol{x} \in \mathbb{Z}^m$.

Besides, we have the following lemma for the bounding spectral norm.

Lemma A.1 ([29]) *Let $\mathbf{X} \in \mathbb{R}^{n\times m}$ be a subgaussian random matrix with parameter s. There exists a universal constant $c \approx 1/\sqrt{2\pi}$ such that for any $t > 0$, we have $s_1(\mathbf{X}) \leq c \cdot s \cdot (\sqrt{m} + \sqrt{n} + t)$ except with probability at most $\frac{2}{e^{\pi t^2}}$.*

A.2 Discrete Gaussian Distribution

For a ring R of degree N, we can define the discrete Gaussian distribution over it in the following way.

Definition A.2 (Definition 4.2 in [45]) *For any positive integer ℓ, the discrete Gaussian distribution over R^ℓ centered around $\boldsymbol{v} \in R^\ell$ with standard deviation $\sigma > 0$ is given by $D^{\ell \cdot N}_{\boldsymbol{v},\sigma}(\boldsymbol{z}) = \frac{e^{-\|\boldsymbol{z}-\boldsymbol{v}\|^2/2\sigma^2}}{\sum_{\boldsymbol{z}' \in \mathcal{R}^\ell} e^{-\|\boldsymbol{z}'\|^2/2\sigma^2}}$. When $\boldsymbol{v} = 0$, we just write $D^{\ell \cdot N}_{\sigma}$ for simplicity. Particularly, we write $D_{\mathbb{Z},\sigma}$ to denote the discrete Gaussian distribution over $\mathbb{Z}$ with standard deviation σ.*

We also need to use the following facts about the discrete Gaussian distribution.

Lemma A.3 (Lemma 4.4 in [45]) *For any positive integer ℓ and any real $\sigma > 0$, and a sample sampled from $D^{\ell \cdot N}_{\sigma}$ defined as above. Then for $\boldsymbol{x} \leftarrow D^{\ell \cdot N}_{\sigma}$, it holds $\Pr\left[\|\boldsymbol{x}\| > t \cdot \sigma\sqrt{\ell N}\right] \le \left(te^{\frac{1-t^2}{2}}\right)^{\ell N}$, where t is any constant value.*

Lemma A.4 (Sum of Discrete Gaussian Samples) *Let $\boldsymbol{x}_i$ for $i \in [n]$ be vectors sampled independently from D^m_σ. Suppose $\sigma \cdot \sqrt{2\pi} \ge \sqrt{2} \cdot \omega(\log m)$, then the distribution of $\sum_i \boldsymbol{x}_i$ is statistically close to $D^m_{\sigma\sqrt{n}}$.*

A.3 Lattices Problems and Underlying Assumptions

Definition A.5 (MSIS [40]) *The $\mathsf{MSIS}_{q,\ell,m,\beta}$ problem (over an implicit ring R) is defined as follows. Given an uniformly random matrix $\mathbf{A} \in R_q^{\ell \times m}$, output vector $\boldsymbol{z} \in R^m$ such that $\mathbf{A}\boldsymbol{z} = 0$ and $0 < \|\boldsymbol{z}\| \le \beta$.*

Definition A.6 (MLWE [40]) *For an error distribution χ over R, the decision $\mathsf{MLWE}_{q,\ell,m,\chi}$ problem (over an implicit ring R) is defined as follows. For $\boldsymbol{s} \xleftarrow{\$} \chi^\ell$, use $A_{q,\boldsymbol{s}}$ to denote the distribution of $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e) \in R_q^\ell \times R_q$, where $\boldsymbol{a} \xleftarrow{\$} R_q^\ell$ and $e \xleftarrow{\$} \chi$. The goal is to distinguish m samples from either $A_{q,\boldsymbol{s}}$ or $\mathcal{U}(R_q^\ell, R_q)$, i.e., distinguish $(\mathbf{A}, \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e})$ from $(\mathbf{A}, \boldsymbol{u})$, where $\mathbf{A} \xleftarrow{\$} R_q^{m \times \ell}$, $\boldsymbol{u} \xleftarrow{\$} R_q^m$, $\boldsymbol{s} \leftarrow \chi^\ell$, and $\boldsymbol{e} \leftarrow \chi^m$.*

Moreover, the $\mathsf{MLWE}_{q,\ell,m,\chi}$ problem defined above are the so-called "Hermite Normal Form" version, as its secrete key and error are chosen from the identical "small" distribution χ. And such an "Hermite Normal Form" can be easily reduced to the standard MLWE via the approach in [5]. For standard MLWE and the above defined MSIS, it is known to be at least as hard as certain standard lattice problems over ideal lattice in the worst case [40]. It should be pointed out that the ring learning with errors problem (RLWE) is the special case of MLWE for $\ell = 1$. Particularly, we denote the corresponding problem as $\mathsf{RLWE}_{q,1,m,\chi}$. More generally, for a small set S_β, we use $\mathsf{RLWE}_{q,1,m,S_\beta}$ to denote that both secret key and error are sampled uniformly at random from S_β.

Definition A.7 (DSPR [43]) *For an error distribution χ over R, the decisional small polynomial ratio (DSPR) assumption $\mathsf{DSPR}_{q,R,\chi}$ says that the following two distributions are indistinguishable:*

- *a polynomial $h = g \cdot f^{-1} \in R_q$, where $g, f \leftarrow \chi$.*
- *a polynomial $u \xleftarrow{\$} R_q$.*

A.4 Rejection Sampling and Dilithium-G

In this paper, we use the well-known Dilithium-G signature scheme the basis for our distributed signature protocols. Thus, for completeness, we present the non-optimized version of Dilithium-G signature scheme in Algorithms 2 to 4.

Algorithm 2: Key generation

Input: $\mathsf{pp} = (R_q, k, \ell, \eta, B, s, M)$
Output: $(\mathsf{sk}, \mathsf{pk})$

1. $\mathbf{A} \xleftarrow{\$} R^{k \times \ell}$
2. $\overline{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R^{k \times (\ell + k)}$
3. $(\boldsymbol{s}_1, \boldsymbol{s}_2) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k; \boldsymbol{s} := \binom{\boldsymbol{s}_1}{\boldsymbol{s}_2}$.
4. $\boldsymbol{t} := \overline{\mathbf{A}}\boldsymbol{s}$
5. $\mathsf{sk} := \boldsymbol{s}$
6. $\mathsf{pk} := (\overline{\mathbf{A}}, \boldsymbol{t})$

return $(\mathsf{sk}, \mathsf{pk})$

Algorithm 3: Signature generation

Input: $\mathsf{sk}, \mathsf{pk}, \mu, \mathsf{pp} = (R_q, k, \ell, \eta, B, s, M)$
Output: valid signature pair $(\boldsymbol{z}, c)$

1. $(\boldsymbol{y}_1, \boldsymbol{y}_2) \xleftarrow{\$} D_s^\ell \times D_s^k; \boldsymbol{y} := \binom{\boldsymbol{y}_1}{\boldsymbol{y}_2}$
2. $\boldsymbol{w} = \overline{\mathbf{A}}\boldsymbol{y}$
3. $c \leftarrow \mathsf{H}_0(\boldsymbol{w}, \mu, \mathsf{pk})$
4. $\boldsymbol{z} := c\boldsymbol{s} + \boldsymbol{y}$
5. With prob. min $(1, D_s^{\ell+k}(\boldsymbol{z}))/(M \cdot D_{c\boldsymbol{s},s}^{\ell+k}(\boldsymbol{z}))$:
6. return $(\boldsymbol{z}, c)$
7. Restart otherwise

Besides, we recall the rejection sampling algorithm as in Lemma A.8, which is important for the security of the FSwA-style signature such as Dilithium-G.

Algorithm 4: Signature verification

Input: $\mathsf{pk}, (\boldsymbol{z}, c), \mu, \mathsf{pp} = (R_q, k, \ell, \eta, B, s, M)$
1. If $||\boldsymbol{z}|| \leq B$ and $c = \mathsf{H}_0(\overline{\mathbf{A}}\boldsymbol{z} - c\boldsymbol{t}, \mu, \mathsf{pk})$:
2. **return** 1
3. Otherwise:
4. **return** 0

Lemma A.8 (Rejection Sampling [45]) *Let V be a subset of $\mathbb{R}^m$ in which all elements have norms less than T, and $\rho : V \to [0,1]$ be a probability distribution. Let $\sigma = \alpha T$ for $\alpha = O(\sqrt{\lambda})$ and*

$$M = \exp\left(\sqrt{\frac{2(\lambda+1)}{\log e}} \cdot \frac{1}{\alpha} + \frac{1}{2\alpha^2}\right) = O(1).$$

Now, sample $\boldsymbol{v} \xleftarrow{\$} \rho$ and $\boldsymbol{y} \xleftarrow{\$} D_\sigma^m$, set $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{v}$, and run $b \leftarrow \mathsf{Rej}(\boldsymbol{z}, \boldsymbol{v}, \sigma)$ defined in Table 3. Then, the probability that $b = 1$ is at least $\frac{1-2^{-\lambda}}{M}$. And conditioned on $b = 1$, the distribution of $(\boldsymbol{v}, \boldsymbol{z})$ is within statistical distance of $\varepsilon_{\mathsf{Rej}} = \frac{2^{-\lambda}}{M}$ of the product distribution $\rho \times D_\sigma^m$.

$\mathsf{Rej}(\boldsymbol{z}, \boldsymbol{v}, \sigma)$
01 $u \xleftarrow{\$} [0,1)$
02 If $u > \frac{1}{M} \cdot \exp(\frac{-2\langle \boldsymbol{z}, \boldsymbol{v}\rangle + \|\boldsymbol{v}\|^2}{2\sigma^2})$
03 return 0 (i.e. abort)
04 Else
05 return 1 (i.e. non-abort)

Table 3. Standard rejection sampling algorithm in [45].

A.5 Supplementary for Trapdoor Homomorphic Commitment Scheme in Section 3.3

In this section, we present the properties of trapdoor homomorphic commitment scheme as follows.

Correctness. Eqv/Inv-TCOM (resp. COM) is correct if for any $\mathsf{msg} \in S_{\mathsf{msg}}$

$$\Pr\left[\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, \mathsf{Rand}, \mathsf{msg}) \to 1 : \begin{array}{l} \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ \mathsf{Rand} \xleftarrow{\$} D(S_r); \\ \mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{msg}; \mathsf{Rand}) \end{array}\right] = 1.$$

Hiding. $\mathsf{Eqv/Inv\text{-}TCOM}$ (resp. COM) is unconditionally (resp. computationally) hiding if the following probability is negligible in λ for any probabilistic adversary (resp. probabilistic polynomial-time adversary) $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

$$\epsilon_{hide} := \left| \Pr\left[b = b' : \begin{array}{c} \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\lambda); \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ (\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}_1(\mathsf{ck}, \mathsf{cpp}) \\ b \xleftarrow{\$} \{0,1\}; \mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{msg}_b) \\ b' \leftarrow \mathcal{A}_2(\mathsf{com}) \end{array} \right] - \frac{1}{2} \right|$$

Binding. $\mathsf{Eqv/Inv\text{-}TCOM}$ (resp. COM) is unconditionally (resp. computationally) binding if the following probability is negligible in λ for any probabilistic adversary (resp. probabilistic polynomial-time adversary) $\mathcal{A}$.

$$\epsilon_{bind} := \Pr\left[\begin{array}{c} \mathsf{msg} \neq \mathsf{msg}' \\ \wedge \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, \mathsf{Rand}, \mathsf{msg}) \rightarrow 1 \\ \wedge \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, \mathsf{Rand}', \mathsf{msg}') \rightarrow 1 \end{array} : \begin{array}{c} \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\lambda) \\ \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ (\mathsf{com}, \mathsf{msg}, \mathsf{Rand}, \mathsf{msg}', \mathsf{Rand}') \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} \right]$$

In particular, unconditionally binding implies that the following probability is also negligible in λ, since otherwise unbounded adversaries can simply check all possible values in S_{com}, S_{msg} and S_r to find a tuple that breaks binding.

$$\epsilon_{ubind} := \Pr\left[\begin{array}{c} \exists(\mathsf{com}, \mathsf{Rand}, \mathsf{msg}, \mathsf{Rand}', \mathsf{msg}') : \\ \mathsf{msg} \neq \mathsf{msg}' \\ \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, \mathsf{Rand}, \mathsf{msg}) \rightarrow 1 \\ \wedge \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}, \mathsf{Rand}', \mathsf{msg}') \rightarrow 1 \end{array} \begin{array}{c} : \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^\lambda) \\ \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \end{array} \right]$$

Secure Trapdoor. $\mathsf{Eqv/Inv\text{-}TCOM}$ has the secure trapdoors if $\mathsf{Eqv\text{-}TCOM}$ and $\mathsf{Inv\text{-}TCOM}$ each has a secure trapdoor.
$\mathsf{Eqv\text{-}TCOM}$ has a secure trapdoor if for any $\mathsf{msg} \in S_{\mathsf{msg}}$, the statistical distance ϵ_{td} between $(\mathsf{ck}, \mathsf{msg}, \mathsf{com}, \mathsf{Rand})$ and $(\mathsf{tck}, \mathsf{msg}, \mathsf{com}^*, \mathsf{Rand}^*)$ is negligible in λ, where $\mathsf{cpp}_{\mathsf{Eqv}} \leftarrow \mathsf{CSetup}(1^\lambda)$; $\mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}_{\mathsf{Eqv}}); \mathsf{Rand} \xleftarrow{\$} D(S_r)$; $\mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{msg}; \mathsf{Rand})$ and $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{TCGen}(\mathsf{cpp}_{\mathsf{Eqv}}); \mathsf{com}^* \leftarrow \mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$; $\mathsf{Rand}^* \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}', \mathsf{msg})$, $\mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{msg}; \mathsf{Rand})$.
$\mathsf{Inv\text{-}TCOM}$ has a secure trapdoor if for any $\mathsf{msg} \in S_{\mathsf{msg}}$, the statistical distance $\epsilon_{\mathsf{td}'}$ between $(\mathsf{ck}', \mathsf{msg}, \mathsf{com}', \mathsf{Rand}')$ and $(\mathsf{tck}', \mathsf{msg}, \mathsf{com}'^*, \mathsf{Rand}'^*)$ is negligible in λ, where $\mathsf{cpp}_{\mathsf{Inv}} \leftarrow \mathsf{CSetup}(1^\lambda)$, $\mathsf{ck}' \leftarrow \mathsf{CGen}(\mathsf{cpp}_{\mathsf{Inv}}), \mathsf{Rand}' \xleftarrow{\$} D(S_r)$; $\mathsf{com}' \leftarrow \mathsf{Commit}_{\mathsf{ck}'}(\mathsf{msg}; \mathsf{Rand}')$. And $\left(\mathsf{tck}', \mathsf{td}'\right) \leftarrow \mathsf{TCGen}(\mathsf{cpp}_{\mathsf{Inv}}), \mathsf{com}'^* \leftarrow \mathsf{Commit}_{\mathsf{tck}'}(\mathsf{msg}; \mathsf{Rand}'^*)$ and $\mathsf{Rand}'^* \xleftarrow{\$} D(S_r)$.

Definition A.9 (Uniform Key) *A commitment key is said to be uniform if the output of* $\mathsf{CGen}(\mathsf{cpp})$ *follows the uniform distribution over the key space* S_{ck}.

Definition A.10 (Additive Homomorphism) *A commitment scheme is said to be additively homomorphic if for any* $\mathsf{msg}, \mathsf{msg}' \in S_{\mathsf{msg}}$

$$\Pr\left[\begin{array}{c|c} \mathsf{Open}_{\mathsf{ck}}(\mathsf{com}+\mathsf{com}', \mathsf{Rand}+\mathsf{Rand}', & \mathsf{cpp} \leftarrow \mathsf{CSetup}(1^{\lambda}) \\ \mathsf{msg}+\mathsf{msg}') \rightarrow 1 & \mathsf{ck} \leftarrow \mathsf{CGen}(\mathsf{cpp}) \\ & \mathsf{Rand} \xleftarrow{\$} D(S_r); \mathsf{Rand}' \xleftarrow{\$} D(S_r) \\ & \mathsf{com} \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{msg}; \mathsf{Rand}) \\ & \mathsf{com}' \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\mathsf{msg}'; \mathsf{Rand}' \end{array}\right] = 1$$

Moreover, in the following detailed security proof for our constructions, we additionally need the commitment of Eqv/Inv-TCOM satisfies statistic/computational uniform property, which is much more stronger than the previously defined hiding property. Particularly, for Inv-TCOM, we require that the distribution of $\mathsf{com}'^{*} \leftarrow \mathsf{Commit}_{\mathsf{tck}'}(\mathsf{msg}; \mathsf{Rand}'^{*})$ is computationally indistinguishable from the uniform one. On the other hand, for Eqv-TCOM, we require that $\mathsf{com}^{*} \leftarrow \mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ has sufficient min-entropy, say, follows the uniform distribution.

Definition A.11 (Uniform Commitment) *For a* Eqv-TCOM *scheme, it is said to be uniform commitment if* $\mathsf{com}^{*} \leftarrow \mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ *follows the uniform distribution over the commitment space* S_{com}*.*

For a Inv-TCOM *scheme, it is said to be computationally uniform commitment if the distribution of* $\mathsf{com}'^{*} \leftarrow \mathsf{Commit}_{\mathsf{tck}'}(\mathsf{msg}; \mathsf{Rand}'^{*})$ *is computationally indistinguishable from the uniform distribution over the commitment space* S_{com}*.*

A.6 Supplementary for n-out-of-n Signature and Multi-Signature in Section 3.4

Algorithm 5: $\mathbf{Exp}_{\mathsf{QDS}}^{\mathsf{(S)UF\text{-}CMA}}(\mathcal{A})$

1 : $\mathsf{Mset} \leftarrow \emptyset$ (or $\underline{\mathsf{MSset} \leftarrow \emptyset}$)
2 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
3 : $(\mu^*, \mathsf{Sig}^*) \leftarrow \mathcal{A}^{\mathcal{O}_n^{\mathsf{QDS}}(\cdot,\cdot)}(\mathsf{pp})$
4 : $b \leftarrow \mathsf{Ver}(\mu^*, \mathsf{Sig}^*, \mathsf{pk})$
5:return $(b = 1) \wedge \mu^* \notin \mathsf{Mset}$ (or$\underline{(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}}$)

Algorithm 6: $\mathbf{Exp}_{\mathsf{QMS}}^{\mathsf{(S)UF\text{-}CMA}}(\mathcal{A})$

1 : $\mathsf{Mset} \leftarrow \emptyset$ (or $\underline{\mathsf{MSset} \leftarrow \emptyset}$), $\mathsf{Kset} \leftarrow \emptyset$
2 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
3 : $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i\in[t]} \leftarrow \mathsf{Gen}(\mathsf{pp})$
4 : $(\mu^*, \mathsf{Sig}^*, L^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{QMS}}(\cdot,\cdot)}(\{\mathsf{pk}_i\}, \mathsf{pp})$
5 : $b \leftarrow \mathsf{Ver}(\mu^*, \mathsf{Sig}^*, L^*)$
6:return $(b = 1) \wedge \mathsf{pk} \in L^* \wedge (\mu^*, L^*) \notin \mathsf{Mset}$ (or $\underline{(\mu^*, \mathsf{Sig}^*, L^*) \notin \mathsf{MSset}}$)

Fig. 8. QDS-(S)UF-CMA and QMS-(S)UF-CMA experiments. Here, *we use UF and SUF to distinguish the settings of unforgeability and strong unforgeability, respectively.* Particularly, for the case of UF, in the left (resp. right) experiment, Mset is the set of all inputs μ such that (sid, μ) was queried by $\mathcal{A}$ to its oracle as the first query with identifier $sid \neq 0$ (resp.with any identifier sid). Note that pk in the left experiment is the public verification key output by P_n when it completes $\mathsf{Gen}_n(\mathsf{pp})$. Besides, the oracles $\mathcal{O}_n^{\mathsf{QDS}}$ and $\mathcal{O}^{\mathsf{QMS}}$ are described in Figure 9 and Figure 10. Furthermore, the case of SUF can be described similarly, except that MSset is composed of not only the queried messages but also the corresponding signatures.

Definition A.12 (QDS-(S)UF-CMA Security) *A distributed signature protocol* QDS *is said to be* QDS-(S)UF-CMA*, i.e., distributed signature (strong) unforgeability against chosen message attacks, secure, if for any quantum polynomial time adversary $\mathcal{A}$, its advantage*

$$\mathbf{Adv}_{\mathsf{QDS}}^{\mathsf{QDS\text{-}(S)UF\text{-}CMA}}(\mathcal{A}) := \Pr\left[\mathbf{Exp}_{\mathsf{QDS}}^{\mathsf{QDS\text{-}(S)UF\text{-}CMA}}(\mathcal{A}) \to 1\right]$$

is negligible in λ, where $\mathbf{Exp}_{\mathsf{QDS}}^{\mathsf{QDS\text{-}(S)UF\text{-}CMA}}(\mathcal{A})$ is described in Figure 8.

Definition A.13 (QMS-(S)UF-CMA Security) *A multisignature protocol* QMS *is said to be* QMS-(S)UF-CMA*, i.e., multisignature (strong) unforgeability against chosen message attacks, secure, if for any quantum polynomial time adversary $\mathcal{A}$, its advantage*

$$\mathbf{Adv}_{\mathsf{QMS}}^{\mathsf{QMS\text{-}(S)UF\text{-}CMA}}(\mathcal{A}) := \Pr\left[\mathbf{Exp}_{\mathsf{QMS}}^{\mathsf{QMS\text{-}(S)UF\text{-}CMA}}(\mathcal{A}) \to 1\right]$$

is negligible in λ, where $\mathbf{Exp}_{\mathsf{QMS}}^{\mathsf{QMS\text{-}(S)UF\text{-}CMA}}(\mathcal{A}))$ is described in Figure 8.

Oracle $\mathcal{O}_n^{\mathsf{QDS}}(sid, m)$
The oracle is initialized with public parameters pp generated by Setup algorithm. The variable $flag$ is initially set to false.
Key Generation. Upon receiving $(0, m)$, if $flag$ = true then return $\perp$. Otherwise do the following:

- If the oracle is queried with $sid = 0$ for the first time then it initializes a machine $\mathcal{M}_0$ running the instructions of party P_n in the distributed key generation protocol $\mathsf{Gen}_n(\mathsf{pp})$. If P_n sends the first message in the key generation protocol, then this message is the oracle reply.
- If $\mathcal{M}_0$ has been already initialized then the oracle hands the machine $\mathcal{M}_0$ the next incoming message m and returns $\mathcal{M}_0$'s reply. If $\mathcal{M}_0$ concludes with local output $(\mathsf{sk}_n, \mathsf{pk})$, then set $flag$ =true.

Signature Generation. Upon receiving (sid, m) with $sid \neq 0$, if $flag$ =false then return $\perp$. Otherwise do the following:

- Initializes a machine $\mathcal{M}_{sid}$ running the instructions of party P_n in the distributed signing protocol $\mathsf{Sign}_n(sid, \mathsf{sk}_n, \mathsf{pk}, \mu)$, which will finally output a signature Sig_μ. The machine $\mathcal{M}_{sid}$ is initialized with the key share and any state information stored by $\mathcal{M}_0$ at the end of the key generation phase. The message μ to be signed is included in Mset (or MSset). If P_n sends the first message in the signing protocol, then this message is the oracle reply.
- If $\mathcal{M}_{sid}$ has been already initialized then the oracle hands the machine $\mathcal{M}_{sid}$ the next incoming message m and returns the next message sent by $\mathcal{M}_{sid}$. If $\mathcal{M}_{sid}$ concludes with local output Sig, then the output obtained by $\mathcal{M}_{sid}$ is returned, (and append such Sig_μ as the signature of μ in MSset).

Fig. 9. Honest party oracle for the distributed signing protocol

Oracle $\mathcal{O}^{\mathsf{QMS}}(sid, m)$
The oracle is initialized with public parameters pp generated by Setup algorithm.
Signature Generation Upon receiving (sid, m) do the following:

- If the oracle is queried with sid for the first time, then parse the incoming message m as (μ, L). If $\mathsf{pk} \notin L$, then it returns $\perp$. Otherwise it initializes a machine $\mathcal{M}_{sid}$ running the instructions of party P in the multi-signature protocol $\mathsf{Sign}(sid, \mathsf{sk}, \mathsf{pk}, \mu, L)$, which will finally output a signature $\mathsf{Sig}_{\mu,L}$. The machine $\mathcal{M}_{sid}$ is initialized with the key pair $(\mathsf{sk}, \mathsf{pk})$ and any state information obtained during $\mathsf{Gen}(\mathsf{pp})$. The pair (μ, L) is included in Mset (or MSset). If P sends the first message in the signing protocol, then this message is the oracle reply.
- If $\mathcal{M}_{sid}$ has been already initialized, then the oracle hands the machine $\mathcal{M}_{sid}$ the next incoming message m and returns the next message sent by $\mathcal{M}_{sid}$. If $\mathcal{M}_{sid}$ concludes, then the output obtained by $\mathcal{M}_{sid}$ is returned, (and append such $\mathsf{Sig}_{\mu,L}$ as the signature of (μ, L) in MSset).

Fig. 10. Honest party oracle for the multi-signature protocol

Remark A.14 *For* QDS *and* QMS*, the* SUF-CMA *security implies the* UF-CMA *security. This is because for any* (μ^*, Sig^*)*, it always holds that* $\mu^* \notin \mathsf{Mset}$ *implies* $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$*. In this paper, we will focus directly on the* SUF-CMA *for our constructions.*

A.7 Concreted Instantiations of Trapdoor Commitment Schemes

Two types of trapdoor commitment schemes can be instantiated using the commitment schemes of [17] and [46], respectively. Below, we provide brief descriptions of these two trapdoor commitment schemes.

Eqv-Commitment Scheme

The used Eqv-COM scheme can be instantiated using the commitment scheme in Section 5.2 of [17]. Particularly, the commitment scheme includes the following algorithms.

- $\mathsf{CSetup}(1^\lambda)$ takes a security parameter as input, and outputs $\mathsf{cpp} = (N, q, \overline{s}, s, B, \ell, w)$.
- $\mathsf{CGen}(\mathsf{cpp})$ takes a commitment parameter as input, and samples $\hat{a}_{1,1} \xleftarrow{\$} R_q^\times$ (a uniform invertible element of R_q) and $\hat{a}_{1,j} \xleftarrow{\$} R_q$ for $j = 2, ..., \ell + 2w$, $\hat{\boldsymbol{a}}_{2,j} \xleftarrow{\$} R_q$ for $j = 3, ...\ell + 2w$. It then outputs:
$$\hat{\mathbf{A}} = \begin{bmatrix} \hat{a}_{1,1} & \hat{a}_{1,2} & \hat{a}_{1,3} & ... & \hat{\boldsymbol{a}}_{1,\ell+2w} \\ 0 & 1 & \hat{a}_{2,3} & ... & \hat{a}_{2,\ell+2w} \end{bmatrix}$$
as ck.
- $\mathsf{Commit}_{\mathsf{ck}}(x; \boldsymbol{r})$ takes $x \in R_q$ and $\boldsymbol{r} \xleftarrow{\$} D_s^{\ell+2w}$ as input, and outputs
$$\boldsymbol{f} = \hat{\mathbf{A}} \cdot \boldsymbol{r} + \begin{bmatrix} 0 \\ x \end{bmatrix} \in R_q^2.$$
To ensure perfect correctness, retry unless $||\boldsymbol{r}|| \leq B$.
- $\mathsf{Open}_{\mathsf{ck}}(\boldsymbol{f}, \boldsymbol{r}, x)$ takes commitments, randomness and message as input, and checks that
$$\boldsymbol{f} = \hat{\mathbf{A}} \cdot \boldsymbol{r} + \begin{bmatrix} 0 \\ x \end{bmatrix} \text{ and } ||\boldsymbol{r}|| \leq B.$$
- $\mathsf{TCGen}(\mathsf{cpp})$ takes a commitment parameter as input, and samples $\overline{\mathbf{A}}$ of the form:
$$\overline{\mathbf{A}} = \begin{bmatrix} \overline{a}_{1,1} & \overline{a}_{1,2} & \overline{a}_{1,3} & ... & \overline{a}_{1,\ell} \\ 0 & 1 & \overline{a}_{2,3} & ... & \overline{a}_{2,\ell} \end{bmatrix}$$
where all the $\overline{a}_{i,j}$ are uniform in R_q, except $\overline{a}_{1,1}$ which is uniform in $R_q^\times$. It also samples $\mathbf{R} \xleftarrow{\$} D_{\overline{s}}^{\ell \times 2w}$ with discrete Gaussian entries. It then outputs $\mathbf{A}$ as the trapdoor td and $\hat{\mathbf{A}} = [\overline{\mathbf{A}} | \mathbf{G} - \overline{\mathbf{A}}\mathbf{R}]$ as the commitment key tck, where $\mathbf{G}$ is given by:
$$\mathbf{G} = \begin{bmatrix} 1 & 2 & ... & 2^{w-1} & 0 & 0 & ... & 0 \\ 0 & 0 & ... & 0 & 1 & 2 & ... & 2^{w-1} \end{bmatrix} \in R^{2 \times 2w}.$$

– $\mathsf{Eqv\text{-}TCommit_{tck}}(\mathsf{td})$ simply returns a uniformly random commitment $\boldsymbol{f} \xleftarrow{\$} R_q^{2\times 1}$. There is no need to keep a state.
– $\mathsf{Eqv_{tck}}(\mathbf{R}, \boldsymbol{f}, x)$ uses the trapdoor discrete Gaussian sampling algorithm of Micciancio-Peikert [[49], Algorithm 3] (or faster variants such as the one described in [31]) to sample $\boldsymbol{r} \xleftarrow{\$} D_{\Lambda_{\boldsymbol{u}}^{\perp}(\hat{\mathbf{A}}, s)}$ according to the discrete Gaussian of parameter s supported on the lattice coset:
$\Lambda_{\boldsymbol{u}}^{\perp}(\hat{\mathbf{A}}) = \{\boldsymbol{z} \in R^{\ell+2w} : \hat{\mathbf{A}} \cdot \boldsymbol{z} \equiv \boldsymbol{u}(\mod q)\}$ where $\boldsymbol{u} = \boldsymbol{f} - \begin{bmatrix} 0 & x \end{bmatrix}$

Theorem A.15 (Theorem 3 of [17]) *The trapdoor commitment scheme of above, with the following choice of parameters:*

$$\begin{array}{ccc} \overline{s} = \Theta(N) & s = \Theta(N^{3/2}\log^2 N) & B = \Theta(N^2 \log^3 N) \\ \ell = w = \lceil \log_2 q \rceil & q = N^{2+\varepsilon} & (\varepsilon > 0, q \textit{ prime}) \end{array}$$

is a secure trapdoor commitment scheme assuming that the $\mathsf{MSIS}_{q,1,\ell+2w-1,2B}$ *problem is hard.*

Inv-Commitment Scheme

The used $\mathsf{Inv\text{-}COM}$ scheme can be instantiated using the commitment scheme in Section 5.2 of [46]. Particularly, the commitment scheme includes the following algorithms.

Construction A.16 ($\mathsf{Inv\text{-}COM}$ Scheme) *The scheme consists of six algorithm as follows.*

– $\mathsf{CSetup}(1^\lambda)$*: Taking a security parameter* λ *as input, the algorithm conducts the following steps:*
  1. *Choose two integers* N, q*, where* N *is a power of 2, and* q *is a prime with* $q = 5 \mod 8$*;*
  2. *Set* $n, k, \hat{\lambda}$ *be integers satisfying* $k = n + 2 + \hat{\lambda}$*;*
  3. *Output* $\mathsf{cpp} = (N, q, n, k, \hat{\lambda})$.
– $\mathsf{CGen}(\mathsf{cpp})$*: Given the public parameter* cpp*, the algorithm conducts the following steps:*
  1. *For the ring* $R = \mathbb{Z}[X]/(X^N + 1)$*, and let* $R_q = \mathbb{Z}_q[X]/(X^N + 1)$.
  2. *Sample* $\mathbf{A} \xleftarrow{\$} R_q^{n\times k}$*, and sample* $\mathbf{B} \xleftarrow{\$} R_q^{2\times k}$.
  3. *Output* $\mathsf{ck} := (\mathbf{A}, \mathbf{B})$.
– $\mathsf{Commit_{ck}}(x; \boldsymbol{r})$*: Given the message vector* $x \in R_q$ *and randomness* $\boldsymbol{r} \xleftarrow{\$} R_q^k$*, the algorithm conducts the following steps:*
  1. *Compute*
$$\mathsf{com} = \begin{bmatrix} \boldsymbol{t}_0 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \cdot \boldsymbol{r} + \begin{bmatrix} \mathbf{0} \\ x \\ x \cdot \lfloor \sqrt{q} \rceil \end{bmatrix}$$
  2. *Output* com.

  To ensure perfect correctness, retry unless $||\boldsymbol{r}|| \leq B'$.

– $\mathsf{Open}_{\mathsf{pk}}(\mathsf{com}, x, \boldsymbol{r})$: *Given the commitment* com*, message* x*, and randomness* $\boldsymbol{r}$*, the algorithm checks if*

$$\mathsf{com} = \begin{bmatrix} \boldsymbol{t}_0 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \cdot \boldsymbol{r} + \begin{bmatrix} \mathbf{0} \\ x \\ x \cdot \lfloor \sqrt{q} \rceil \end{bmatrix}, \text{ and } \|\boldsymbol{r}\| \leq B'.$$

– $\mathsf{TCGen}(\mathsf{cpp})$: *Given the public parameter* cpp*, the algorithm conducts the following steps:*
  1. *For the ring* $R = \mathbb{Z}[X]/(X^N + 1)$*, and let* $R_q = \mathbb{Z}_q[X]/(X^N + 1)$.
  2. *Sample* $\mathbf{A} \xleftarrow{\$} R_q^{n \times k}$, $\boldsymbol{s}_i \leftarrow S_1^n, \boldsymbol{e}_i \leftarrow S_1^k$ *for* $i \in [2]$*, where* $\boldsymbol{s}_i, \boldsymbol{e}_i$ *are vectors over* R_q.
  3. *Compute* $\boldsymbol{b}_i = \mathbf{A}^\top \cdot \boldsymbol{s}_i + \boldsymbol{e}_i (\bmod q)$*. And set* $\mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2]^\top$.
  4. *Output* $\mathsf{tck} := (\mathbf{A}, \mathbf{B})$, $\mathsf{td} := (\boldsymbol{s}_1, \boldsymbol{s}_2)$.

– $\mathsf{Inv}_{\mathsf{tck}}(\mathsf{com}, \mathsf{td})$: *On input the key* tck*,* $\mathsf{com} = (\boldsymbol{t}_0, t_1, t_2)$ *and* td*, the algorithm conducts the following steps:*
  1. *Compute* $u_1 = t_1 - \langle \boldsymbol{t}_0, \boldsymbol{s}_1 \rangle$ *and* $u_2 = t_2 - \langle \boldsymbol{t}_0, \boldsymbol{s}_2 \rangle$.
  2. *Compute* $\Delta_2 = u_2 - u_1 \cdot \lfloor \sqrt{q} \rceil (\bmod \lfloor \sqrt{q} \rceil)$.
  3. *Compute and output* $m' = \frac{u_2 - \Delta_{2,}}{\lfloor \sqrt{q} \rceil}$.

Below, we present the security and correctness of Construction A.16.

Correctness. The correctness consists of two respects: a valid commitment can be opened correctly, and a valid commitment generated with tck can be inverted successfully through using $\mathsf{sk} := (\boldsymbol{s}_1, \boldsymbol{s}_2)$. As the former one is trivial, below we just focus on the latter one. Suppose $\mathsf{com} = (\boldsymbol{t}_0, t_1, t_2)$ is a valid commitment, then for the valid commitment key and secret key $\mathsf{tck} := (\mathbf{A}, \mathbf{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2]^\top)$, $\mathsf{td} := (\boldsymbol{s}_1, \boldsymbol{s}_2)$, it holds

$$\begin{cases} u_1 & = t_1 - \langle \boldsymbol{t}_0, \boldsymbol{s}_1 \rangle = \langle \boldsymbol{e}_1, \boldsymbol{r} \rangle + x \ (\bmod q) \\ u_2 & = t_2 - \langle \boldsymbol{t}_0, \boldsymbol{s}_2 \rangle \\ & = \langle \boldsymbol{e}_2, \boldsymbol{r} \rangle + x \cdot \lfloor \sqrt{q} \rceil \ (\bmod q) \end{cases} \tag{6}$$

In this case, we denote $\langle \boldsymbol{e}_i, \boldsymbol{r} \rangle$ and $\langle \boldsymbol{e}_2, \boldsymbol{r} \rangle$ as Δ_1 and Δ_2, respectively. Thus, we have

$$\begin{cases} u_1 = \Delta_1 + x \ (\bmod q) \\ u_2 = \Delta_2 + x \cdot \lfloor \sqrt{q} \rceil \ (\bmod q) \end{cases} \tag{7}$$

Then after multiplying $\lfloor \sqrt{q} \rceil$ into both sides of the first equation, we can get

$$\begin{cases} u_1 \cdot \lfloor \sqrt{q} \rceil = \Delta_1 \cdot \lfloor \sqrt{q} \rceil + x \cdot \lfloor \sqrt{q} \rceil \ (\bmod q) \\ u_2 = \Delta_2 + x \cdot \lfloor \sqrt{q} \rceil \ (\bmod q) \end{cases} \tag{8}$$

Furthermore, we can get

$$k = u_2 - u_1 \cdot \lfloor \sqrt{q} \rceil = \Delta_2 - \Delta_1 \cdot \lfloor \sqrt{q} \rceil (\bmod q). \tag{9}$$

Notice that each coefficient of $\langle \boldsymbol{e}_1, \boldsymbol{r}\rangle = \sum_{j\in[k]}(e_{1,j} \cdot r_j)$ is upper bounded by $k \cdot N$. Notice that if Δ_1, Δ_2 are small enough such that $\|\Delta_i\|_\infty \leq \lfloor\sqrt{q}\rceil/4$, then no reduction modulo q takes place in the Equation (9).

In this case, Δ_2 can be easily recovered by further modulo $\lfloor\sqrt{q}\rceil$ for Equation (9), i.e., $\Delta_2 = k(\bmod \lfloor\sqrt{q}\rceil)$. Finally, we can obtain that

$$x = \frac{u_2 - \Delta_2}{\lfloor\sqrt{q}\rceil} \bmod q.$$

Security of Construction A.16. Notice that according to the $\mathsf{MLWE}_{q,n,k,1}$ assumption, $\boldsymbol{b}_i$ is computational indistinguishability from uniform. Conditioned on this case, the above encryption scheme can be viewed as a BDLOP commitment scheme with parameter n, k, ℓ, $R_q = \mathbb{Z}_q[X]/(X^N + 1)$, and thus we have the following theorem.

Theorem A.17 (Theorem 3 of [17]) *The trapdoor commitment scheme of above is a secure trapdoor commitment scheme satisfies binding and hiding properties, following from* $\mathsf{MSIS}_{q,n,k,8\sqrt{2}\cdot\alpha\cdot\kappa\cdot k\cdot N}$ *and* $\mathsf{MLWE}_{q,\hat{\lambda},k,1}$*, respectively. Here, α is the parameter for rejection sampling as in Lemma A.8, κ is the parameter for the challenge set of* NIZKPoK *system as in Table 2, assuming that the* MSIS *problem is hard.*

B Supplementary for QPRF in Section 4

Due to the space limitation in the main body, we present many more supplementary materials for QPRF in Section 4.

B.1 Detailed Proof for theorems in Section 4.1

Theorem B.1 (Restatement of Theorem 4.3) *Let $\chi = D_{\bar{R},\bar{r}}$ be a small distribution over $\bar{R}$, where all coefficients of each polynomial are chosen independently from $D_{\mathbb{Z},\bar{r}}$. Let $\bar{q} \geq \bar{p} \cdot \bar{\ell} \cdot (\bar{r} \cdot \sqrt{2(\bar{N}+\bar{\ell})} \cdot \omega(\sqrt{\log(\bar{N}+\bar{\ell})}))^{\bar{\ell}} \cdot \bar{N}^{\omega(1)}$. Let* QPRF *be as in Construction 4.1. If the* $\mathsf{RLWE}_{\bar{q},1,\bar{m},\chi}$ *holds, then Construction 4.1 is a secure* QPRF.

Proof. Similar to the proof of Theorem 6.1 by Zhandry in [60], we first define a class of functions $G : \mathcal{K} \times [2]^{\bar{\ell}} \to \bar{R}_{\bar{q}}^{1\times\bar{d}}$ as

$$G_{\mathsf{k}}(x) = (a_1, \ldots, a_{\bar{m}}) \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \mod \bar{q},$$

where $x := (x_1, \ldots, x_{\bar{\ell}}) \in \{0,1\}^{\bar{\ell}}$. Then, we define a related class of functions $\tilde{G}^{(\bar{\ell})}$ in the following recursive way.

- $\tilde{G}^{(0)}$ is a function from from $[2]^0$ to $\bar{R}_{\bar{q}}^{1\times\bar{m}}$ defined as follows: sample $\boldsymbol{a}^\top = (a_1, \ldots, a_{\bar{m}}) \leftarrow \bar{R}_{\bar{q}}^{1\times\bar{m}}$, and set $\tilde{G}^{(0)}(\epsilon) = \boldsymbol{a}^\top$.

– $\tilde{G}^{(i)}$ is a function from from $[2]^i$ to $\bar{R}_{\bar{q}}^{1\times\bar{m}}$ defined as follows: choose a random $\tilde{G}^{(i-1)}$, sample $s_i \leftarrow \chi$, and for each $x' := (x_1, \ldots, x_{\bar{\ell}-1}) \in [2]^{i-1}$, sample $\boldsymbol{e}_{x'} \leftarrow \chi^{1\times\bar{m}}$. Then,

$$\tilde{G}^{(i)}(x = (x'|x_i)) = \tilde{G}^{(i-1)}(x') \cdot s_i^{x_i} + x_i \cdot \boldsymbol{e}_{x'} \mod \bar{q}.$$

Furthermore, we define two truly random function $U : [2]^{\bar{\ell}} \to \bar{R}_{\bar{p}}^{1\times\bar{m}}$ and $U' : [2]^{\bar{\ell}} \to \bar{R}_{\bar{q}}^{1\times\bar{m}}$.

With above definitions, the high-level proof route is that for any adversary choosing query $x \in [2]^{\bar{\ell}}$, it holds

$$\mathsf{QPRF}_{\mathsf{k}}(x) := \lfloor G_{\mathsf{k}}(x) \rceil_{\bar{p}} \overset{\text{(i)}}{\approx_c} \lfloor \tilde{G}^{(\bar{\ell})}(x) \rceil_{\bar{p}} \overset{\text{(ii)}}{\approx_c} \lfloor U'(x) \rceil_{\bar{p}} \overset{\text{(iii)}}{\approx_c} U(x), \tag{10}$$

with overwhelming probability.

According to the above definition on $G^{(\bar{\ell})}(x)$, we know that

$$\begin{aligned}
\tilde{G}(x_1 \cdots x_{\bar{\ell}}) &= (\cdots((\boldsymbol{a}^\top \cdot s_1^{x_1} + x_1 \cdot \boldsymbol{e}_\epsilon) \cdot s_2^{x_2} + x_2 \cdot \boldsymbol{e}_1) \cdots) \cdot s_{\bar{\ell}}^{x_{\bar{\ell}}} + x_{\bar{\ell}} \cdot \boldsymbol{e}_{x_1 \cdots x_{\bar{\ell}-1}} \\
&= \boldsymbol{a}^\top \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} + x_1 \cdot \boldsymbol{e}_\epsilon \cdot \prod_{i=2}^{\bar{\ell}} s_i^{x_i} + x_2 \cdot \boldsymbol{e}_{x_1} \cdot \prod_{i=3}^{\bar{\ell}} s_i^{x_i} + \cdots x_{\bar{\ell}} \cdot \boldsymbol{e}_{x_1 \cdots x_{\bar{\ell}-1}} \\
&= G_{\mathsf{k}}(x) + x_1 \cdot \boldsymbol{e}_\epsilon \cdot \prod_{i=2}^{\bar{\ell}} s_i^{x_i} + x_2 \cdot \boldsymbol{e}_{x_1} \cdot \prod_{i=3}^{\bar{\ell}} s_i^{x_i} + \cdots x_{\bar{\ell}} \cdot \boldsymbol{e}_{x_1 \cdots x_{\bar{\ell}-1}},
\end{aligned}$$

where the above computations are conducted over $\bar{R}_{\bar{q}}$. Notice that according to Lemma 2.3 in [7], for $s_i \leftarrow \chi$, and each error vector $\boldsymbol{e}_{x_1 \cdots x_{i-1}} \leftarrow D_{\bar{R},\bar{r}}^{1\times\bar{m}}$, it holds the difference between the coefficient of each entry of $G_{\mathsf{k}}(x)$ and the corresponding coefficient of $\tilde{G}(x)$ is bounded by $\bar{B} = \bar{\ell} \cdot (\bar{r} \cdot \sqrt{2\bar{N}} \cdot \omega(\sqrt{\log \bar{N}}))^{\bar{\ell}} / \sqrt{\bar{N}}$.

Then, in order to ensure the indistinguishability even with all QPRF queries in $[2]^{\bar{\ell}}$ by the quantum adversary, just as Zhandry's argument in [60], we reset $\bar{B} = \bar{\ell} \cdot (\bar{r} \cdot \sqrt{2(\bar{N} + \bar{\ell})} \cdot \omega(\sqrt{\log(\bar{N} + \bar{\ell})}))^{\bar{\ell}} / \sqrt{\bar{N}}$. With this value $\bar{B}$, for each $y \in \mathbb{Z}_{\bar{q}}$, we can define $\mathsf{BAD}(y)$ to be the event that $\lfloor y + [-\bar{B}, \bar{B}] \rceil_{\bar{p}} \neq \lfloor y \rceil_{\bar{p}}$. Suppose, we can set $\bar{q} \geq \bar{p} \cdot \bar{\ell} \cdot (\bar{r} \cdot \sqrt{2(\bar{N} + \bar{\ell})} \cdot \omega(\sqrt{\log(\bar{N} + \bar{\ell})}))^{\bar{\ell}} \cdot \bar{N}^{\omega(1)}$ such that $\frac{(2\bar{B}+1)\bar{p}}{\bar{q}} \cdot \bar{m} \cdot \bar{N} = \mathsf{negl}(\lambda)$. Then, for all coefficients in the output of $\tilde{G}^{(\bar{\ell})}(x)$, the BAD happens with negligible probability. And thus, the step (i) in (10) will hold.

And the computational indistinguishability of $\tilde{G}^{(\bar{\ell})}(x)$ follows from the oracle-LWE indistinguishability defined by Zhandry in [60], which further follows from the underlying $\mathsf{RLWE}_{\bar{q},1,\bar{m},\chi}$ assumption, defined in Definition A.7. This also implies that the step (ii) in (10) holds.

Finally, for the step (iii) in (10), it holds due to the fact that the event BAD happens with negligible probability. Overall, (10) is set up, and thus the statement of this theorem holds. □

Lemma B.2 (Restatement of Lemma 4.4) *For any* $\bar{N} \geq 1, \bar{q} \geq 2$, $\bar{d} = \lceil \log \bar{q} \rceil$, $\bar{m} = \bar{d} + 2$, $\bar{p} \geq 3 \cdot \sqrt{\bar{m}\bar{N}} \cdot (\sqrt{2\bar{N}} + \sqrt{\bar{d}\bar{N}})$*, there exist the following* *two efficient algorithms* (TrapGen, RLWRInvert).

TrapGen$(1^{\bar{N}}, \bar{q}, \bar{m}, \bar{d})$*: A* PPT *algorithm which on input positive integers* $\bar{N}, \bar{q}, \bar{m}, \bar{d}$*, first samples a vector* $(a_1, a_2) \in \bar{R}_{\bar{q}}^2$ *and trapdoor* $\mathbf{T} \in S_1^{2\times\bar{d}}$*, where* $R_{\bar{q}} = \mathbb{Z}_{\bar{q}}[X]/(X^{\bar{N}}+1)$*. Furthermore, the algorithm computes* $(a_3, \ldots, a_{\bar{m}}) = (a_1, a_2)\mathbf{T} + \boldsymbol{g}^\top$*, where* $\boldsymbol{g}^\top = (1, 2, \ldots, 2^{\bar{d}-1})$*. In this case,* $\boldsymbol{a}^\top = (a_1, \ldots, a_{\bar{m}})^\top$ *is computationally close to uniform over* $\bar{R}_{\bar{q}}^{\bar{m}}$*, according to the* RLWE *assumption. Clearly, it holds* $\boldsymbol{a}^\top \cdot \begin{bmatrix} -\mathbf{T} \\ \mathbf{I}_{\bar{d}\times\bar{d}} \end{bmatrix} = \boldsymbol{g}^\top$*, where* $\mathbf{I}_{\bar{d}\times\bar{d}} \in \bar{R}_{\bar{q}}^{\bar{d}\times\bar{d}}$ *is an identity matrix.*

RLWRInvert$(\mathbf{T}, \boldsymbol{a}, \boldsymbol{b})$*: An algorithm taking as input* $(\boldsymbol{a}, \mathbf{T})$ *output by* TrapGen$(1^{\bar{n}}, \bar{q})$*, and some value* $\boldsymbol{b} \in R_{\bar{p}}^{\bar{m}}$ *such that* $\boldsymbol{b}^\top = \lfloor \boldsymbol{a}^\top \cdot s \rceil_{\bar{p}}$ *for some* $s \in \bar{R}_{\bar{q}}$*, outputs* s*.*

Proof. Given RLWR samples $(\boldsymbol{a}^\top, \boldsymbol{b}^\top = \lfloor \boldsymbol{a}^\top \cdot s \rceil_{\bar{p}})$, we first transform it into RLWE samples $(\boldsymbol{a}^\top, \boldsymbol{a}^\top \cdot s + \boldsymbol{e}^\top)$, and then invert such RLWE problem through using the trapdoor for $\boldsymbol{a}$. Thus, we will get the secret s for the original RLWR samples.

Particularly, given $\boldsymbol{b} \in R_{\bar{p}}^{\bar{m}}$, we compute $\lfloor \frac{\bar{q}}{\bar{p}} \cdot \boldsymbol{b} \rceil \in \bar{R}_{\bar{q}}^{\bar{m}}$. More precisely, it holds

$$\boldsymbol{c} = \lfloor \frac{\bar{q}}{\bar{p}} \cdot \boldsymbol{b} \rceil = \lfloor \frac{\bar{q}}{\bar{p}} \cdot \lfloor \frac{\bar{p}}{\bar{q}} \cdot \boldsymbol{a} \cdot s \rceil \rceil = \lfloor \frac{\bar{q}}{\bar{p}} \cdot (\frac{\bar{p}}{\bar{q}} \cdot \boldsymbol{a} \cdot s + \boldsymbol{e}') \rceil = \boldsymbol{a} \cdot s + \boldsymbol{e},$$

where $\boldsymbol{e}' \in (-\frac{1}{2}, \frac{1}{2}]^{\bar{N}\cdot\bar{m}}$ and $\boldsymbol{e} \in (-\frac{\bar{q}}{2\bar{p}}, \frac{\bar{q}}{2\bar{p}}]^{\bar{N}\cdot\bar{m}}$. Then, we compute

$$\hat{\boldsymbol{c}}^\top = \boldsymbol{c} \cdot \begin{bmatrix} -\mathbf{T} \\ \mathbf{I}_{\bar{d}\times\bar{d}} \end{bmatrix} = s \cdot \boldsymbol{g}^\top + \hat{\boldsymbol{e}}^\top = s \cdot \boldsymbol{g}^\top + \boldsymbol{e}^\top \cdot \begin{bmatrix} -\mathbf{T} \\ \mathbf{I}_{\bar{d}\times\bar{d}} \end{bmatrix}. \tag{11}$$

For simplicity, we denote $\mathbf{T}' = \begin{bmatrix} -\mathbf{T} \\ \mathbf{I}_{\bar{d}\times\bar{d}} \end{bmatrix}$. And it holds $s_1(\mathbf{T}') = \sqrt{s_1(\mathbf{T})^2 + 1}$. Thus we have $\|\hat{\boldsymbol{e}}\| \leq s_1(\mathbf{T}') \cdot \frac{\bar{q}}{2\bar{p}} \cdot \sqrt{\bar{N} \cdot \bar{m}}$. According to the property of primitive vector $\boldsymbol{g}^\top$, we know that (11) will be successfully inverted if $\hat{\boldsymbol{e}} \in \mathcal{P}_{1/2}(\bar{q} \cdot \mathbf{B}^{-\top})$, where $\mathbf{B}$ is the basis for the lattice $\Lambda_{\bar{q}}^{\perp}(\boldsymbol{g}^\top)$, satisfying $\|\mathbf{B}\| \leq \sqrt{5}$. This equivalently implies that $\|\hat{\boldsymbol{e}}\| \leq \frac{\bar{q}}{2\sqrt{5}}$. Thus, it suffices to set $s_1(\mathbf{T}') \cdot \frac{\bar{q}}{2\bar{p}} \cdot \sqrt{\bar{N} \cdot \bar{m}} \leq \frac{\bar{q}}{2\sqrt{5}}$. Combining $s_1(\mathbf{T}) \leq (\sqrt{2\bar{N}} + \sqrt{\bar{m} \cdot \bar{N}})$ by Lemma A.1, it is sufficient to set $\bar{p} \geq 3 \cdot \sqrt{\bar{m}\bar{N}} \cdot (\sqrt{2\bar{N}} + \sqrt{\bar{d}\bar{N}})$. □

Lemma B.3 (Restatement of Lemma 4.6) *For the adversary* $\mathcal{A}$ *without the trapdoor* $\mathbf{T}$ *of the vector* $\boldsymbol{a}$*, if the* $\mathsf{RLWE}_{\bar{q},1,1,S_1}$ *assumption holds, then Constructions 4.1 and 4.5 are computational indistinguishability, even* $\mathcal{A}$ *queries the functions in a superposition for any polynomial times.*

Proof. Notice that for any $i \in [\bar{d}]$, we know $a_{i+2} = (a_1, a_2) \cdot \begin{pmatrix} t_{1,i} \\ t_{2,i} \end{pmatrix} + 2^i \mod \bar{q}$. Furthermore, due to the $\mathsf{RLWE}_{\bar{q},1,1,S_1}$ assumption, we know that for uniform and public ring elements a_1, a_2, $(a_1, a_2) \cdot \begin{pmatrix} t_{1,i} \\ t_{2,i} \end{pmatrix}$ is computationally indistinguishable from uniform over $\bar{R}_{\bar{q}}$. As a result, Constructions 4.1 and 4.5 are computational indistinguishability. □

Theorem B.4 (Restatement of Theorem 4.7) *For some $\boldsymbol{a} \in R_{\bar{q}}^{\bar{m}}$ and integers $\bar{p}, \bar{q}, \bar{d}, \bar{N}, \bar{m}$ such that $\bar{q} \geq \bar{p} \cdot \bar{\ell} \cdot (\bar{r} \cdot \sqrt{2(\bar{N}+\bar{\ell})} \cdot \omega(\sqrt{\log(\bar{N}+\bar{\ell})}))^{\bar{\ell}} \cdot \bar{N}^{\omega(1)} \geq \left(\bar{r} \cdot \sqrt{2\bar{N}}\right)^{\bar{\ell}}$, $\bar{d} = \lceil \log \bar{q} \rceil$, and $\bar{m} = \bar{d}+2$ and $\bar{p} \geq 3 \cdot \sqrt{\bar{m}\bar{N}} \cdot (\sqrt{2\bar{N}} + \sqrt{\bar{d}\bar{N}})$, suppose the oracle $O_{\mathsf{RLWRInvert}}$ in Algorithm 1 correctly invert $\left\lfloor \boldsymbol{a}^\top \cdot s \right\rceil_{\bar{p}}$ for any $s \in \bar{R}_{\bar{q}}$. Then, for any invertible ring element $s_i \in \bar{R}_{\bar{q}}$, Algorithm 1 correctly inverts $\mathsf{Inj\text{-}QPRF}_{\boldsymbol{a},\{s_i\}} = \left\lfloor \boldsymbol{a}^\top \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \right\rceil_{\bar{p}}$, assuming the $\mathsf{DSPR}_{\bar{q},\bar{R},\chi}$ assumption.*

Proof. From the oracle $O_{\mathsf{RLWRInvert}}$, we can get the correct ring element $\hat{s} = \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \mod \bar{q}$ from the above Step 1, due to our parameter settings on $\bar{m}, \bar{N}, \bar{d}, \bar{\ell}, \bar{q}$ and $\bar{p}$.

Then, in order to show the correctness of the following Steps 3 and 4, Particularly, as each $s_i \leftarrow D_{\bar{R},\bar{r}}$ is invertible over $\bar{R}_{\bar{q}}$ with overwhelming probability, if the matrix $s'_{i-1} = s_i \cdot s_{i+1}^{x_{i+1}} \cdot \ldots \cdot s_{\bar{\ell}}^{x_{\bar{\ell}}}$, then it is clearly that the norm of $s'_i = s_i^{-1} \cdot s'_{i-1} = s_{i+1}^{x_{i+1}} \cdot \ldots \cdot s_{\bar{\ell}}^{x_{\bar{\ell}}}$ will be smaller than $(\bar{r}\sqrt{2\bar{N}})^{\bar{\ell}-i}$ with overwhelming probability, according to Lemma A.3.

On the other hand, if the matrix s'_{i-1} does not consist of the i-th small ring element s_i, i.e., $s'_{i-1} = s_{i+1}^{x_{i+1}} \cdot \ldots \cdot s_{\bar{\ell}}^{x_{\bar{\ell}}}$, then $s'_i = s_i^{-1} \cdot s'_{i-1} = s_i^{-1} \cdot s_{i+1}^{x_{i+1}} \cdot \ldots \cdot s_{\bar{\ell}}^{x_{\bar{\ell}}}$. Without loss of generality, we assume $x_{i+1} = 1$. In this case, we know

$$s'_i = \frac{s_{i+1}}{s_i} \cdot \ldots \cdot s_{\bar{\ell}}^{x_{\bar{\ell}}}.$$

According to the DSPR assumption, we know that $\frac{s_{i+1}}{s_i}$ is computationally indistinguishable from uniform over R_q. And thus, s'_i is computationally indistinguishable from uniform, which implies that $\|s'_i\| > (\bar{r}\sqrt{2\bar{N}})^{\bar{\ell}-i}$ with overwhelming probability, according to our parameter setting.

Summing up all above analyzes, we conclude that Algorithm 1 correctly inverts $\mathsf{Inj\text{-}QPRF}_{\boldsymbol{a},\{s_i\}} = \left\lfloor \boldsymbol{a}^\top \cdot \prod_{i=1}^{\bar{\ell}} s_i^{x_i} \right\rceil_{\bar{p}}$. □

C Supplementary for QDS_2 in Section 5

Due to the space limitation in the main body, we present many more supplementary for QDS_2 in Section 5

Theorem C.1 (Restatement of Theorem 5.2) *Suppose the trapdoor commitment schemes $\mathsf{Inv\text{-}TCOM}$ and $\mathsf{Eqv\text{-}TCOM}$ are secure, additively homomorphic, have uniform keys and uniform commitment. Particularly, the output of $\mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ has sufficient min-entropy ϑ. And suppose there exists QPRF that can be programable and invertible simultaneously. For any quantum polynomial-time adversary $\mathcal{A}$ that initiates a single key generation protocol by querying $\mathcal{O}_n^{\mathsf{QDS}_2}$ with sid $= 0$, initiates Q_s signature generation protocols by querying $\mathcal{O}_n^{\mathsf{QDS}_2}$ with sid $\neq 0$, and makes Q_h quantum superpositions queries to random oracle $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}'_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5$, the protocol QDS_2 of Figures 5, 6, 7 is*

QDS-SUF-CMA *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions in the* QROM, *where* $\beta = 2\sqrt{B_n^2+\kappa}$. *Concretely, using other parameters specified in Table 2, the advantage of* $\mathcal{A}$ *is bounded as follows.*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}_{\mathsf{QDS}_2}(\mathcal{A}) \leq{}& 2\varepsilon_{\mathsf{Inj\text{-}QPRF}} + 5\varepsilon_{\mathsf{QPRF}} + e(Q_h+Q_s+1)\Big[(Q_h+Q_s)(\varepsilon_{\mathsf{td}}+\varepsilon_{\mathsf{td}'})\\ &+ 2(Q_h+Q_s)\cdot\varepsilon_{\mathsf{QPRF}} + \frac{3}{2}\sqrt{Q_h}2^{\frac{-t\cdot\vartheta}{2}} + 2\varepsilon_{\mathsf{QPRF}} + t\cdot Q_s\cdot(m-1)\cdot\mathsf{negl}(\lambda)\\ &+ t\cdot Q_s\cdot\varepsilon_{\mathsf{Rej}} + \frac{3}{2}\sqrt{Q_h}(2^{\frac{-q^{klN}}{2}} + 2^{\frac{-q^{kN}}{2}}) + 4(\varepsilon_{\mathsf{QPRF}}+\varepsilon_{\mathsf{Inj\text{-}QPRF}})\\ &+ \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}} + 2(Q_h+1)2^{-(t\log m)/2} + Q_s\cdot t\cdot\varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2}\cdot 2^{-t\cdot\vartheta}\\ &+ \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}\Big]\end{aligned}$$

Below, we first sketch the proof idea, before presenting the formal proof. According to Definition A.12, we need to prove that for any efficient adversary $\mathcal{A}$ against QDS_2, its advantage $\mathbf{Adv}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}_{\mathsf{QDS}_2}(\mathcal{A})$ is negligible. In order to do this, we conduct the following two steps:

- We first show that the party P_n in the experiment $\mathbf{Adv}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}_{\mathsf{QDS}_2}(\mathcal{A})$ can be simulated by a simulator $\mathcal{B}$ defined in Figure 11, together with its subroutines Figures 13 to 16. And $\mathcal{B}$ do not have any secret key, through using a sequence of hybrid experiments. Particularly, in the key generation and signature query phases, we use the QPRF to simulate the quantum random oracle, which satisfy the requirements of extraction and reprogrammability. In the signature query phase, we use the trapdoor-equivocation commitment scheme and the adaptive programming of H_5 to simulate the signature.
- Then, we show that in such a simulated experiment, the signature is strong unforgeability, through establishing a reduction from MSIS and the binding properties of Inv-TCOM, following from the similar proof idea of [57]. Particularly, we first show that there is an efficient extractor Ext in Figure 12, such that given a valid forged message-signature pair $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$, Ext can output a solution for MSIS problem, if the used Inv-TCOM scheme satisfies the binding property. And then, we bound the probability of generating a valid forged message-signature pair $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$ by the union bound of two events happen: Ext succeeds and Ext fails.

Proof. We first begin with the real experiment denoted as $\mathbf{G}_0$.

$\mathbf{G}_0$ This is the real experiment just as defined in Figure 8. Here $\mathcal{B}$ holds the real random oracles $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}'_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5$, and allows $\mathcal{A}$ to query all H_i and H'_1 in superpositions. Besides, with the honestly generated secret key share $\boldsymbol{s}_n$, $\mathcal{B}$ answers $\mathcal{A}$'s key generation and signature generation queries, just as in Figures 9, which invokes Figures 5 and Figures 6. Let $\Pr[\mathbf{G}_i]$ denote a probability that $\mathcal{A}$ wins the experiment $\mathbf{G}_i$, i.e., outputs a valid forgery, at the game $\mathbf{G}_i$.

Algorithm $\mathcal{B}(\mathbf{A}, \boldsymbol{t})$ The algorithm is initialized with a set $\mathsf{MSset} = \emptyset$ and a flag $\mathsf{BAD}_4 = false$. Here, MSset is used to store the queried messages together with the related signatures.

Honest party oracle simulation. Upon receiving a query of the form (sid, m) from $\mathcal{A}$, reply the query as described in $\mathsf{Sim}\mathcal{O}_n^{\mathsf{QDS}_2}(sid, m)$(Fig.13). If $\mathsf{Sim}\mathcal{O}_n^{\mathsf{QDS}_2}(sid, m)$ halts with output $\perp$ then $\mathcal{B}$ also halts with output $\perp$.

Random oracle simulation. Upon receiving a query to the random oracles from $\mathcal{A}$, reply the query as described in Fig.15.

Forgery. The variable BAD_4 is initially set to 0. Upon receiving a forgery $(\mu^*, \mathsf{Sig}^*) = (\mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i^*\}_{i\in[t]}, \{r_i^*\}_{i\in[t]}, \{r_{i,J_i}^{*\prime}\}_{i\in[t]})$ from $\mathcal{A}$, it conducts:

1. If $(\mu^*, \mathsf{Sig}^*) \in \mathsf{MSset}$ then $\mathcal{B}$ halts with output $\perp$.
2. Make queries $\mathsf{ck}^* \leftarrow \mathsf{H}_3(\mu^*, \mathsf{pk}), \mathsf{ck}'^* \leftarrow \mathsf{H}_4(\mu^*, \mathsf{pk}), c_{i,j}^* \leftarrow \mathsf{H}_0(i, j, \mu^*, \mathsf{pk}, \mathsf{ck}^*, \mathsf{ck}'^*)$ where $i \in [t], j \in [m]$ and $J_1||...||J_t \leftarrow \mathsf{H}_5(\mathsf{pk}, \{\mathsf{com}_i^*\}_{i\in[t]}, \{c_{i,j}^*\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t]})$.
3. If $||\boldsymbol{z}_i^*|| > B_n$ or $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \overline{\mathbf{A}}\boldsymbol{z}_i^* - c_{i,J_i}^*\boldsymbol{t}) \neq 1$ or $\mathsf{Inv\text{-}Open}_{\mathsf{ck}'^*}(\widetilde{\mathsf{com}}_{i,J_i}^*, r_{i,J_i}'^*, \boldsymbol{z}_i^*) \neq 1$, then $\mathcal{B}$ halts with output $\perp$. Otherwise, $\mathcal{B}$ computes $\mathsf{H}_3'(\mu^*, \mathsf{pk})$. If the output of $\mathsf{H}_3'(\mu^*, \mathsf{pk})$ is not $\perp$, (i.e., $\mathsf{Eqv\text{-}TCGen}$ was called), then set flag $\mathsf{BAD}_4 = 1$ and $\mathcal{B}$ halts with output $\perp$.
4. $\mathcal{B}$ halts with output $(\mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i^*\}_{i\in[t]}, \{r_i^*\}_{i\in[t]}, \{r_{i,J_i}'^*\}_{i\in[t]})$.

Fig. 11. The algorithm simulating the view of $\mathcal{A}$ in $\mathbf{Exp}_{\mathsf{QDS}_2}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}(\mathcal{A})$ experiment

Input :$\mathsf{H}_0, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5, \mathsf{pk}, \mu, \mathsf{Sig} = \Big(\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r_{i,J_i}'\}_{i\in[t]}\Big)$,
compute $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$, $\mathsf{td}' \leftarrow \mathsf{H}_4(\mu, \mathsf{pk})$, $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{pk}, \mathsf{ck}, \mathsf{ck}')$ for all $i \in [t], j \in [m]$, and $J_1||...||J_t \leftarrow \mathsf{H}_5(\mathsf{pk}, \mu, \{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]})$.
Furthermore, conduct the followings, for $i = 1$ to t do
    for $j = 1$ to m except J_i do
        for each $\boldsymbol{z}_{i,j}' \leftarrow \mathsf{Inv}(\widetilde{\mathsf{com}}_{i,j}, \mathsf{td}')$ do
            if $||\boldsymbol{z}_{i,j}'|| \leq B \wedge \mathsf{Eqv\text{-}Open}_{\mathsf{ck}}(\mathsf{com}_i, r_i, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_{i,j}' - c_{i,j}\boldsymbol{t})$
                return $\left(\frac{\boldsymbol{z}_i - \boldsymbol{z}_{i,j}'}{c_{i,J_i} - c_{i,j}}\right)$

Fig. 12. Extractor for $\mathbf{G}_9$

Below, we explicit describe the Forgery phase in the experiment as follows, as we will need to modify its certain steps in the following hybrid experiments. **Forgery.** When $\mathcal{A}$ outputs a forgery $\mathsf{Sig}^* = (\{\mathsf{com}_i^*\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t],j\in[m]}$ $\{\boldsymbol{z}_i^*\}_{i\in[t]}, \{r_i^*\}_{i\in[t]}, \{r_{i,J_i}'^*\}_{i\in[t]})$ for μ^* at the end of experiment, $\mathcal{B}$ proceeds as follows.

1. If $(\mu^*, \mathsf{Sig}^*) \in \mathsf{MSset}$ then $\mathcal{B}$ halts with output $\perp$.
2. Compute $\mathsf{ck}^* \leftarrow \mathsf{H}_3(\mu^*, \mathsf{pk}), \mathsf{ck}'^* \leftarrow \mathsf{H}_4(\mu^*, \mathsf{pk}), c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu^*, \mathsf{pk}, \mathsf{ck}^*, \mathsf{ck}'^*)$ where $i \in [t], j \in [m]$ and $J_1^*||...||J_t^* \leftarrow \mathsf{H}_5(\mathsf{pk}, \mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{c_{i,j}^*\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t]})$.
3. If $||\boldsymbol{z}_i^*|| > B_n$ or $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \overline{\mathbf{A}}\boldsymbol{z}_i^* - c_{i,J_i^*}^*\boldsymbol{t}) \neq 1$ or $\mathsf{Inv\text{-}Open}_{\mathsf{ck}'^*}(\widetilde{\mathsf{com}}_{i,J_i^*}^*, r_{i,J_i^*}'^*, \boldsymbol{z}_i^*) \neq 1$ then $\mathcal{B}$ halts with output $\perp$.
4. $\mathcal{B}$ halts with output $(\mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i^*\}_{i\in[t]}, \{r_i^*\}_{i\in[t]}, \{r_{i,J_i^*}'^*\}_{i\in[t]})$.

Thus, we have
$$\Pr[\mathbf{G}_0] = \mathbf{Adv}_{\mathsf{QDS}_2}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}(\mathcal{A}).$$

$\mathbf{G}_1$ This experiment is identical to $\mathbf{G}_0$, except that the random oracles $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_1', \mathsf{H}_2, \mathsf{H}_5$ are simulated by QPRFs. Among them, $\mathsf{H}_0 : \{0,1\}^{l_0^*} \rightarrow C, \mathsf{H}_1' : \{0,1\}^{l_1^*} \rightarrow$

Oracle $\mathcal{O}_n^{\mathsf{QDS}}(sid, m)$
The simulator takes as input the public parameters pp output by the Setup algorithm. The variable $flag$ is initially set to be false.
Key Generation. Upon receiving $(0, m)$, if $flag$ = true then return $\perp$. Otherwise do the following:
- If the oracle is queried with $sid = 0$ for the first time then it initializes a machine $\mathcal{M}_0$ running the instructions $\mathsf{SimGen}_n(\mathsf{pp}, \mathbf{A}, \boldsymbol{t})$ (Fig.14). If P_n sends the first message in the key generation protocol, then this message is the oracle reply.
- If $\mathcal{M}_0$ has been already initialized then the oracle hands the machine $\mathcal{M}_0$ the next incoming message m and returns $\mathcal{M}_0$'s reply. If $\mathcal{M}_0$ fails with output $\perp$ at any point, then the oracle stops the simulation with output $\perp$. If $\mathcal{M}_0$ concludes $\mathsf{SimGen}_n(\mathsf{pp}, \mathbf{A}, \boldsymbol{t})$ with local output $(\boldsymbol{t}_n, \mathsf{pk})$, then set $flag$ =true.

Signature Generation. Upon receiving (sid, m) with $sid \neq 0$, if $flag$ =false then return $\perp$. Otherwise do the following:
- If the oracle is queried with sid for the first time then parse the incoming message m as μ. It initializes a machine $\mathcal{M}_{sid}$ running the instructions of $\mathsf{SimSign}_n(sid, \boldsymbol{t}_n, \mathsf{pk}, \mu)$(Fig. 16). The machine $\mathcal{M}_{sid}$ is initialized with the key share and any state information stored by $\mathcal{M}_0$. The message μ to be signed is included in MSset. If P_n sends the first message in the signing protocol, then this message is the oracle reply.
- If $\mathcal{M}_{sid}$ has been already initialized then the oracle hands the machine $\mathcal{M}_{sid}$ the next incoming message m and returns the next message sent by $\mathcal{M}_{sid}$. If $\mathcal{M}_{sid}$ fails with output $\perp$ at any point then the oracle stops the simulation with output $\perp$. If $\mathcal{M}_{sid}$ concludes with local output Sig, then the output obtained by $\mathcal{M}_{sid}$ is returned, and append such Sig as the signature of μ in MSset.

Fig. 13. Honest party oracle simulator for QDS_2

Protocol $\mathsf{QDS}_2.\mathsf{SimGen}_n(\mathsf{pp}, \mathbf{A}, \boldsymbol{t})$. The simulator is parameterized by public parameters described in Table 2 and relies on the random oracles: $\mathsf{H}_1 : \{0,1\}^{l_1^*} \to \{0,1\}^{l_1}$, $\mathsf{H}_1' : \{0,1\}^{l_1^*} \to \{0,1\}^{l_1'}$, $\mathsf{H}_2 : \{0,1\}^{l_2^*} \to \{0,1\}^{l_2}$. The variables $\mathsf{BAD}_1, \mathsf{BAD}_2$ is initially set to false
Matrix Generation
1. Sample a random seed $s_n \in \{0,1\}^{l_1^* - \log n}$, generate and output a random oracle commitment $g_n \leftarrow \mathsf{H}_1(s_n, n)$.
2. Upon receiving g_u for all $u \in [n-1]$,
   (a) Invoke Algorithm 1 on input $(g_1, ..., g_{n-1})$ to obtain $((s_1, 1), ..., (s_{n-1}, n-1))$.
   (b) Compute $\mathbf{A}_u = \mathsf{H}_1'(s_u, u)$ for all $u \in [n-1]$.
   (c) Compute $\mathbf{A}_n := \mathbf{A} - \sum_{u=1}^{n-1} \mathbf{A}_u$.
   (d) Reprogram the random oracle $\mathsf{H}_1'(s_n, n) = \mathbf{A}_n$ and set public random matrix $\overline{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$, where $\mathbf{A} := \sum_{u\in[n]} \mathbf{A}_u$.
   (e) Send out the seed s_n.
3. Upon receiving s_u for all $u \in [n-1]$, if $\mathsf{H}_1(s_u, u) \neq g_u$ for some u, then send out $\perp$.

Key Pair Generation
1. Sample $g_n' \xleftarrow{\$} \{0,1\}^{l_2}$ and send out g_n'.
2. Upon receiving g_u' for all $u \in [n-1]$ proceed as follows.
   (a) Invoke Algorithm 1 on input $(g_1', ..., g_{n-1}')$ to obtain $((\boldsymbol{t}_1, 1), ..., (\boldsymbol{t}_{n-1}, n-1))$.
   (b) Compute $\boldsymbol{t}_n := \boldsymbol{t} - \sum_{u=1}^{n-1} \boldsymbol{t}_u$.
   (c) Reprogram the random oracle $\mathsf{H}_2(\boldsymbol{t}_n, n) = g_n'$ and then send out $\boldsymbol{t}_n$.
3. Upon receiving $\boldsymbol{t}_u$ for all $u \in [n-1]$, if $\mathsf{H}_2(\boldsymbol{t}_u, u) \neq g_u'$ for some u then send out $\perp$.

If neither the protocol does not output $\perp$, the simulator obtains public key share $\boldsymbol{t}_n$ and $\mathsf{pk} = (\mathbf{A}, \boldsymbol{t})$ as local output.

Fig. 14. Key generation simulator for QDS_2

$\{0,1\}^{l_1'}$, $\mathsf{H}_5 : \{0,1\}^{l_5^*} \to \{0,1\}^{l_5}$ are simulated as QPRFs in Construct 4.1. According to Theorem 4.3, QPRFs and quantum random oracle are computationally indistinguishable except with a negligible probability $\varepsilon_{\mathsf{QPRF}} = \mathsf{negl}(\lambda)$, for any efficient quantum adversary. $\mathsf{H}_1 : \{0,1\}^{l_1^*} \to \{0,1\}^{l_1}$, $\mathsf{H}_2 : \{0,1\}^{l_2^*} \to \{0,1\}^{l_2}$ are simulated as Inj-QPRFs in Construct 4.5. According to Lemma 4.6, Construct 4.1 and Construct 4.5 are computational indistin-

Algorithm Random Oracle Simulation

$\mathsf{H}_0(x)$
1. Simulate H_0 as $\mathsf{QPRF}_{\mathsf{k}_0}: \{0,1\}^{l_0^*} \to \{0,1\}^{l_0}$, where $\mathsf{k}_0 \xleftarrow{\$} \mathcal{K}$
2. Return $\mathsf{H}_0(x)$

$\mathsf{H}_1(x)$
1. Simulate H_1 as $\mathsf{Inj\text{-}QPRF}_{\mathsf{k}_1}: \{0,1\}^{l_1^*} \to \{0,1\}^{l_1}$, where $\mathsf{k}_1 \xleftarrow{\$} \mathcal{K}$
2. Return $\mathsf{H}_1(x)$

$\mathsf{H}_1'(x)$
1. Simulate H_1' as $\mathsf{QPRF}_{\mathsf{k}_1'}: \{0,1\}^{l_1^*} \to \{0,1\}^{l_1'}$, where $\mathsf{k}_1' \xleftarrow{\$} \mathcal{K}$
2. Return $\mathsf{H}_1'(x)$

$\mathsf{H}_2(x)$
1. Simulate H_2 as $\mathsf{Inj\text{-}QPRF}_{\mathsf{k}_2}: \{0,1\}^{l_2^*} \to \{0,1\}^{l_2}$, where $\mathsf{k}_2 \xleftarrow{\$} \mathcal{K}$
2. Return $\mathsf{H}_2(\boldsymbol{t}_u, u)$

$\mathsf{H}_3(x), \mathsf{H}_3'(x)$
1. Parse x as (μ, pk)
2. Invoke $\mathsf{QPRF}_{\mathsf{k}_3}(\mu, \mathsf{pk}): \{0,1\}^{l_3^*} \to (\{0,1\}^{l_{ra_1}} \times \{0,1\}^{l_{ra_2}})$, where $\mathsf{k}_3 \xleftarrow{\$} \mathcal{K}$ and l_{ra_1}, l_{ra_2} are the lengths of r_1, r_2, respectively.
3. Compute $(ra_1, ra_2) = \mathsf{QPRF}_{\mathsf{k}_3}(\mu, \mathsf{pk})$
4. If the number of 1 in ra_1 is more than num, then compute $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{Eqv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Eqv}}, ra_2)$, return tck and td as $\mathsf{H}_3(\mu, \mathsf{pk})$ and $\mathsf{H}_3'(\mu, \mathsf{pk})$, respectively. Here, num is set to make $\Pr[\|ra_1\|_1 > num] = \varpi$
5. Otherwise, compute $\mathsf{ck} \leftarrow \mathsf{Eqv\text{-}CGen}(\mathsf{cpp}_{\mathsf{Eqv}}, r_2)$, return ck and $\perp$ as $\mathsf{H}_3(\mu, \mathsf{pk})$ and $\mathsf{H}_3'(\mu, \mathsf{pk})$, respectively.

$\mathsf{H}_4(x), \mathsf{H}_4'(x)$
1. Parse x as (μ, pk)
2. Invoke $\mathsf{QPRF}_{\mathsf{k}_4}(\mu, \mathsf{pk}): \{0,1\}^{l_4^*} \to \{0,1\}^{l_r}$, where $\mathsf{k}_4 \xleftarrow{\$} \mathcal{K}$ and l_r is the length of r.
3. Compute $r = \mathsf{QPRF}_{\mathsf{k}_4}(\mu, \mathsf{pk})$
4. Then compute $(\mathsf{tck}', \mathsf{td}') \leftarrow \mathsf{Inv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Inv}}, r)$, return tck' and td' as $\mathsf{H}_4(x)$ and $\mathsf{H}_4'(x)$, respectively.

$\mathsf{H}_5(x)$
1. Simulate H_5 as $\mathsf{QPRF}_{\mathsf{k}_5}: \{0,1\}^{l_5^*} \to \{0,1\}^{l_5}$, where $\mathsf{k}_5 \xleftarrow{\$} \mathcal{K}$
2. Return $\mathsf{H}_5(x)$

Fig. 15. Quantum random oracle simulator and the related functions for QDS_2

guishability, except with a negligible probability $\varepsilon_{\mathsf{Inj\text{-}QPRF}} = \mathsf{negl}(\lambda)$, for any efficient quantum adversary. Thus, we have

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_0]| \leq 5\varepsilon_{\mathsf{QPRF}} + 2\varepsilon_{\mathsf{Inj\text{-}QPRF}}.$$

$\mathbf{G}_2$ This experiment is identical to $\mathbf{G}_1$, except with the simulation of H_3, H_4 and the related several differences in $\mathsf{QDS.Sign}_n$.

When receiving a query (μ, pk), $\mathcal{B}$ run $\mathsf{tck}' \leftarrow \mathsf{H}_4(\mu, \mathsf{pk})$ and $\mathsf{td}' \leftarrow \mathsf{H}_4'(\mu, \mathsf{pk})$ as in Figure 15. Particularly, H_4 first computes $r \leftarrow \mathsf{QPRF}_{\mathsf{k}_4}(\mu, \ pk)$, where $\mathsf{QPRF}_{\mathsf{k}_4}$ is a quantum secure pseudorandom function as Construct 4.1, then invokes $(\mathsf{tck}', \mathsf{td}') \leftarrow \mathsf{Inv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Inv}}, r)$, return tck' and td', respectively.

Recall that the core idea of running H_3 is to make sure that for all sign queries, H_3 will return a trapdoor commitment key tck, i.e., $\mathsf{tck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$. Then through using the related $\mathsf{td} \leftarrow \mathsf{H}_3'(\mu, \mathsf{pk})$, $\mathcal{B}$ can equivocate commitments $\mathsf{com}_i \leftarrow \mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ to arbitrary plaintexts $\boldsymbol{w}_i \in R_q^k$ later. And for the forgery submitted by $\mathcal{A}$, H_3 will return the actual commitment key ck, i.e., $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$. Thus, we can simulate H_3 as in Figure 15. Particularly, through using QPRF as follows: if receiving a query (μ, pk), H_3 first

Protocol $\mathsf{QDS}_2.\mathsf{SimSign}_n(sid, \boldsymbol{t}_n, \mathsf{pk}, \mu)$

The simulator is parameterized by public parameters described in Table 2 and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^{l_0^*} \to C$, $\mathsf{H}_3 : \{0,1\}^{l_3^*} \to \mathsf{Eqv}\text{-}S_{\mathsf{ck}}$, $\mathsf{H}_4 : \{0,1\}^{l_4^*} \to \mathsf{Inv}\text{-}S_{\mathsf{ck}}$ and $\mathsf{H}_5 : \{0,1\}^{l_5^*} \to \{0,1\}^{l_5}$. The simulator assumes that $\mathsf{QDS}_2.\mathsf{SimGen}_n(\mathsf{pp})$ has been previously invokes. If a party halts with $\perp$ at any point, then all $\mathsf{SimSign}_n(sid, \boldsymbol{t}_n, \mathsf{pk}, \mu)$ executions are aborted. The variable BAD_3 is initially set to false.

Inputs

1. The simulator receives a unique sessions ID $sid, \boldsymbol{t}_n, \mathsf{pk} = (\mathbf{A}, \boldsymbol{t})$ and message $\mu \in M$ as input.
2. The simulator verifies that sid has not been used before (if it has been, the protocol is not executed).
3. The simulator locally computes a per-message commitment key by querying a random oracle $\mathsf{tck}' \leftarrow \mathsf{H}_4(\mu, \mathsf{pk})$.
4. The simulator locally computes a per-message commitment key by querying a random oracle $\mathsf{tck} \leftarrow \mathsf{H}_3(\mu, \mathsf{pk})$. Compute $\mathsf{QPRF}_{\mathsf{k}_3}(\mu, \mathsf{pk}) = (ra_1, ra_2)$. If the number of 1 in ra_1 is less than num, then set $\mathsf{BAD}_3 = 1$ and simulation fails with output $\perp$. Otherwise obtain trapdoor $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{Eqv}\text{-}\mathsf{TCGen}(\mathsf{cpp}_{\mathsf{Eqv}}, ra_2)$.

Signature Generation P_n works as follows:

1. Compute the first group messages as follows:
   (a) for $i = 1$ to t; conduct as follows:
      i. Sample $(J_1 \| \dots \| J_t) \xleftarrow{\$} \mathbb{Z}_m^t$, where $J_i \in \mathbb{Z}_m$.
      ii. Compute $\mathsf{com}_i^{(n)} \leftarrow \mathsf{Eqv}\text{-}\mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td})$.
      iii. for $j = J_i$, conduct as follows:
         A. Derive challenges $c_{i,J_i} \leftarrow \mathsf{H}_0(i, J_i, \mu, \mathsf{pk}, \mathsf{tck}, \mathsf{tck}')$.
         B. Sample $\boldsymbol{z}_{i,J_i}^{(n)} \xleftarrow{\$} D_s^{\ell+k}$
         C. Output $\boldsymbol{z}_{i,J_i}^{(n)}$ with probability $1/M$.
         D. If the above $\boldsymbol{z}_{i,J_i}^{(n)}$ does not output, then go to the Step ii.
      iv. For $j = J_i$, compute $\widetilde{\mathsf{com}}_{i,J_i}^{(n)} \leftarrow \mathsf{Inv}\text{-}\mathsf{Commit}_{\mathsf{tck}'}(\boldsymbol{z}_{i,J_i}^{(n)}, r_{i,J_i}^{\prime(n)})$ where $r_{i,J_i}^{\prime(n)} \xleftarrow{\$} \mathsf{Inv}\text{-}S_r$.
      v. For all $j \in [t] \backslash \{J_i\}$, sample $\widetilde{\mathsf{com}}_{i,J_i}^{(n)} \xleftarrow{\$} S_{\mathsf{Inv}\text{-}\mathsf{com}}$.
   (b) Send out $(\{\mathsf{com}_i^{(n)}\}_{i \in [t]}, \{\widetilde{\mathsf{com}}_{i,j}^{(n)}\}_{i \in [t], j \in [m]})$.
2. Upon receiving $(\{\mathsf{com}_i^{(u)}\}_{i \in [t]}, \{\widetilde{\mathsf{com}}_{i,j}^{(u)}\}_{i \in [t], j \in [m]})$for all $u \in [n-1]$, compute the signature shares as follows:
   (a) Set $\mathsf{com}_i := \sum_{u \in [n]} \mathsf{com}_i^{(u)}$ and $\widetilde{\mathsf{com}}_{i,j} := \sum_{u \in [n]} \widetilde{\mathsf{com}}_{i,j}^{(u)}$ for all $i \in [t], j \in [m]$.
   (b) Reprograme H_5 as $\mathsf{H}_5(\mathsf{pk}, \mu, \{\mathsf{com}_i\}_{i \in [t]}, \{c_{i,j}\}_{i \in [t], j \in [m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i \in [t], j \in [m]}) := (J_1 || \dots || J_t)$. And derive randomness $r_i^{(n)} \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}_i^{(n)}, \boldsymbol{w}_i^{(n)} = \overline{\mathbf{A}} \boldsymbol{z}_{i,J_i}^{(n)} - c_{i,J_i} \boldsymbol{t}_n)$.
   (c) Send out $(\{\boldsymbol{z}_{i,J_i}^{(n)}\}_{i \in [t]}, \{r_i^{(n)}\}_{i \in [t]}, \{r_{i,J_i}^{\prime(n)}\}_{i \in [t]})$.
3. Upon receiving $(\{\boldsymbol{z}_{i,J_i}^{(u)}\}_{i \in [t]}, \{r_i^{(u)}\}_{i \in [t]}, \{r_{i,J_i}^{\prime(u)}\}_{i \in [t]})$ for all $u \in [n]$, compute the combined signature as follows:
   (a) For each $u \in [n-1]$, reconstruct $\boldsymbol{w}_i^{(u)} := \overline{\mathbf{A}} \boldsymbol{z}_{i,J_i}^{(u)} - c_{i,J_i} \boldsymbol{t}_u$ and validate the signature shares
   $$||\boldsymbol{z}_{i,J_i}^{(u)}|| \le B, \mathsf{Eqv}\text{-}\mathsf{Open}_{\mathsf{tck}}(\mathsf{com}_i^{(u)}, r_i^{(u)}, \boldsymbol{w}_i^{(u)}) = 1$$
   and
   $$\mathsf{Inv}\text{-}\mathsf{Open}_{\mathsf{tck}'}(\widetilde{\mathsf{com}}_{i,J_i}^{(u)}, r_{i,J_i}^{\prime(u)}, \boldsymbol{z}_{i,J_i}^{(u)}) = 1.$$
   for all $i \in [t]$. If the check fails for some u then send out $\perp$.
   (b) Compute $\boldsymbol{z}_{i,J_i} := \sum_{u \in [n]} \boldsymbol{z}_{i,J_i}^{(u)}$, $r_i := \sum_{u \in [n]} r_i^{(u)}$ and $r'_{i,J_i} := \sum_{u \in [n]} r_{i,J_i}^{\prime(u)}$ for all $i \in [t]$.

If the protocol does not abort, P_n obtains a signature $(\{\mathsf{com}_i\}_{i \in [t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i \in [t], j \in [m]}, \{\boldsymbol{z}_{i,J_i}\}_{i \in [t]}, \{r_i\}_{i \in [t]}, \{r'_{i,J_i}\}_{i \in [t]})$ as local output.

Fig. 16. Signature generation simulator for QDS_2

computes $(ra_1, ra_2) \leftarrow \mathsf{QPRF}_{\mathsf{k}_3}(\mu, \mathsf{pk})$, where $\mathsf{QPRF}_{\mathsf{k}_3}$ is a quantum secure pseudorandom function as Construct 4.1, then

- If the number of 1 in ra_1 is more than num, then $\mathcal{B}$ invokes $\mathsf{Eqv}\text{-}\mathsf{TCGen}$ with $\mathsf{cpp}_{\mathsf{Eqv}}$ and ra_2 as public parameter and randomness respectively, to obtain $(\mathsf{tck}, \mathsf{td})$. Finally, $\mathcal{B}$ returns tck as the output of $\mathsf{H}_3(\mu, \mathsf{pk})$.

- Otherwise, $\mathcal{B}$ invokes $\mathsf{Eqv\text{-}CGen}$ with $\mathsf{cpp}_{\mathsf{Eqv}}$ and ra_2 as public parameter and randomness respectively, to obtain ck. Finally, $\mathcal{B}$ returns ck as the output of $\mathsf{H}_3(\mu, \mathsf{pk})$.

In the above process, $\mathcal{B}$ also runs the H'_3 as in Figure 15 to get $\mathsf{td} = \mathsf{H}'_3(\mu, \mathsf{pk})$. Here, we set the value num such that the probability that the number of 1 in ra_1 is more than num is ϖ.

Based on the above simulation for H_3, H_4, $\mathbf{G}_2$ has the following concrete differences with $\mathsf{QDS.Sign}_n$ in $\mathbf{G}_1$.

- With respect to **Inputs 3**: Given (μ, pk), compute $(ra_1, ra_2) \leftarrow \mathsf{QPRF}_{\mathsf{k}_3}(\mu, \mathsf{pk})$. If the number of 1 in ra_1 is less than num (i.e., $\mathsf{Eqv\text{-}TCGen}$ was not called), then set the flag $\mathsf{BAD}_3 = 1$ and halts with output $\perp$. Otherwise obtain the trapdoor $(\mathsf{tck}, \mathsf{td}) \leftarrow \mathsf{Eqv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Eqv}}; ra_2)$.
- With respect to **Signature Generation** 1.(a).ii: Generate $\mathsf{com}_i^{(n)} \leftarrow \mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ instead of committing to $\boldsymbol{w}_i^{(n)}$, for $i \in [t]$. Besides, in this step, $\mathcal{B}$ does not sample the corresponding randomness $r_i^{(n)}$.
- With respect to **Signature Generation** 2.(b): After getting challenge $J_1 || \cdots || J_n$, $\mathcal{B}$ derives randomness $r_i^{(n)} \leftarrow \mathsf{Eqv}_{\mathsf{tck}}(\mathsf{td}, \mathsf{com}_i^{(n)}, \boldsymbol{w}_i^{(n)})$, where $\boldsymbol{w}_i^{(n)}$ has been computed in the step of **Signature Generation** 1.(a).i.

Moreover, $\mathbf{G}_2$ has the following concrete differences with **Forgery** phase in $\mathbf{G}_1$. Particularly, when $\mathcal{A}$ outputs a successful forgery $(\{\mathsf{com}_i\}^*_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}^*_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i\}^*_{i\in[t]}, \{r_i\}^*_{i\in[t]}, \{r'_{i,J_i}\}^*_{i\in[t]}, \mu^*)$ at the end of the experiment, we modify the step 3 of $\mathbf{G}_2$ as follows.

Forgery 3. If $||\boldsymbol{z}_i^*|| > B_n$ or $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \overline{\mathbf{A}}\boldsymbol{z}_i^* - c^*_{i,J_i}\boldsymbol{t}) \neq 1$ or $\mathsf{Inv\text{-}Open}_{\mathsf{ck}'^*}(\widetilde{\mathsf{com}}^*_{i,J_i}, r'^*_{i,J_i}, \boldsymbol{z}_i^*) \neq 1$ then $\mathcal{B}$ halts with output $\perp$. Furthermore, $\mathcal{B}$ computes $\mathsf{H}'_3(\mu^*, \mathsf{pk})$. If the output of $\mathsf{H}'_3(\mu^*, \mathsf{pk})$ is not $\perp$, (i.e., $\mathsf{Eqv\text{-}TCGen}$ was called), then set flag $\mathsf{BAD}_4 = 1$ and $\mathcal{B}$ halts with output $\perp$.

Note that due to the way H_3 is simulated, if $\mathcal{B}$ does not output $(0, \perp)$, it is now guaranteed that ck^* is generated by $\mathsf{Eqv\text{-}CGen}$ instead of $\mathsf{Eqv\text{-}TCGen}$. Furthermore, according to the security of $\mathsf{Inv/Eqv\text{-}TCOM}$, we have

$$\Pr[\mathbf{G}_2] \geq \varpi^{Q_h+Q_s} \cdot (1-\varpi) \cdot \Pr[\mathbf{G}_1] - (Q_h+Q_s) \cdot (\varepsilon_{\mathsf{td}} + \varepsilon_{\mathsf{td}'}) - 2(Q_h+Q_s) \cdot \varepsilon_{\mathsf{QPRF}},$$

where $\varepsilon_{\mathsf{td}}, \varepsilon_{\mathsf{td}'}$ are the statistical distances of true commitment and trapdoor commitment for $\mathsf{Eqv\text{-}TCOM}$ and $\mathsf{Inv\text{-}TCOM}$, respectively.
In other word, it is only successful neither BAD_3 nor BAD_4 is set above. Note that by setting $\varpi = (Q_h + Q_s)/(Q_h + Q_s + 1)$ since $(1/(1 + 1/(Q_h + Q_s)))^{(Q_h+Q_s)} \geq 1/e$ for $Q_h + Q_s \geq 0$ we obtain

$$\Pr[\mathbf{G}_2] \geq \frac{\Pr[\mathbf{G}_1]}{e(Q_h + Q_s + 1)} - (Q_h + Q_s) \cdot (\varepsilon_{\mathsf{td}} + \varepsilon_{\mathsf{td}'}) - 2(Q_h + Q_s) \cdot \varepsilon_{\mathsf{QPRF}}.$$

$\mathbf{G}_3$ This game is identical to $\mathbf{G}_2$ except at the choice of the challenge $J_1||...||J_t$. Particularly, instead of getting the challenge $J_1||...||J_t \leftarrow \mathsf{H}_5(\cdot)$, $\mathcal{B}$ firstly

chooses $J_1||...||J_t$ at random before choosing $\boldsymbol{y}_i^{(n)} \leftarrow D_\sigma^{\ell+k}$. Then, after the step of **Signature Generation** 2.(a), $\mathcal{B}$ programs the random oracle H_5 at the particular input $x := (\mathsf{pk}, \mu, \{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]})$, such that $\mathsf{H}_5(x) := J_1||...||J_t$.
Notice that, according to the properties of $\mathsf{Eqv\text{-}TCommit}$, it holds $\mathsf{com}_i^{(n)} \leftarrow \mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ has sufficient min-entropy ϑ, so does $\mathsf{com}_i := \sum_{u\in[n]} \mathsf{com}_i^{(u)}$. This further implies the programmed point x has sufficient min-entropy. Thus, according to Theorem 4.8, it holds

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2]| \leq (4+\sqrt{2})\sqrt{Q_h}2^{\frac{-t\cdot\vartheta}{2}} + 2\varepsilon_{\mathsf{QPRF}}.$$

$\mathbf{G}_4$ This game is identical to $\mathbf{G}_3$ except at the following steps. Particularly, at **Signature Generation** 1.(b), for $j \in [m]\backslash J_i$, $\mathcal{B}$ directly chooses $\widetilde{\mathsf{com}}_{i,j}^{(n)}$ uniformly from the corresponding commitment space of $\mathsf{Inv\text{-}TCommit}$. For $j = J_i$, $\mathcal{B}$ still computes $\widetilde{\mathsf{com}}_{i,J_i}^{(n)}$ in the original way.
Thus, according to the pseudorandomness of $\mathsf{Inv\text{-}TCommit}$, it holds

$$|\Pr[\mathbf{G}_4] - \Pr[\mathbf{G}_3]| \leq Q_s \cdot t \cdot (m-1) \cdot \mathsf{negl}(\lambda).$$

$\mathbf{G}_5$ This game is identical to $\mathbf{G}_4$ except at the following steps. Particularly, at **Signature Generation** 1.(a).iii, for $j \in [m]\backslash J_i$, $\mathcal{B}$ directly omit the computations of $\boldsymbol{z}_{i,j}^{(n)}$. For $j = J_i$, $\mathcal{B}$ still computes $\boldsymbol{z}_{i,j}^{(n)}$ in the original way.
Thus, as the $\{\boldsymbol{z}_{i,j}^{(n)}\}_{j\in[m]\backslash J_i}$ have never been used in the following steps, it holds it holds

$$\Pr[\mathbf{G}_5] = \Pr[\mathbf{G}_4].$$

$\mathbf{G}_6$ This game is identical to $\mathbf{G}_5$ except at the following points.
Honest party oracle simulatuon. The $\mathcal{B}$ doesn't honestly generate $\boldsymbol{z}_{i,J_i}^{(n)}$ through using the secret key share sk_n anymore, but instead sampling it according to the rejection sampling algorithm as follows.

- **Signature Generation** 1.(a).i. $\mathcal{B}$ does nothing here.
- **Signature Generation** 1.(a).iii. B. Samples $\boldsymbol{z}_{i,J_i}^{(n)} \leftarrow D_\sigma^{\ell+k}$, output it with probability $1/M$.

The above mentioned $\boldsymbol{z}_{i,j}^{(n)}$ sampled from $D_\sigma^{\ell+k}$ and then output with probability $1/M$, are statistically indistinguishable from the real ones, according to the property of rejection sampling in Lemma A.8. Thus, we have

$$|\Pr[\mathbf{G}_6] - \Pr[\mathbf{G}_5]| = t \cdot Q_s \cdot \varepsilon_{\mathsf{Rej}}.$$

Notice that up until now, i.e., in $\mathbf{G}_6$, the signing queries are answered through using the simulated algorithm $\mathsf{SimSign}_n$ in Figure 16, and it doesn't rely on the actual secret key $\boldsymbol{s}_n$ anymore.

$\mathbf{G}_7$ This experiment is identical to $\mathbf{G}_6$, except with the generation of $\mathbf{A}_n$. Rather than directly sampling $s_n \xleftarrow{\$} \{0,1\}^{l_2^*}$ and computing $\mathbf{A}_n \leftarrow \mathsf{H}_1'(s_n)$, $\mathcal{B}$ first picks the random matrix $\mathbf{A} \in R_q^{k\times\ell}$ and a random seed $s_n \xleftarrow{\$} \{0,1\}^{l_2^*}$, and send out a random oracle commitment $g_n \xleftarrow{\$} \mathsf{H}_1(s_n)$. Then, after receiving all other random oracle commitments $g_u \in \{0,1\}^{l_1}$, $\mathcal{B}$ can extract the adversary's corresponding committed seeds $s_1, ..., s_{n-1} \in R_q^{k\times\ell}$, and compute $\mathbf{A}_u = \mathsf{H}'_1(s_u)$ for all $u \in [n-1]$. As H_1 has been simulated by $\mathsf{Inj\text{-}QPRF}_{\mathsf{k}_1}$ in Construction 4.5, according to Theorem 4.7, this extraction can be efficiently done through using Algorithm 1. Furthermore, $\mathcal{B}$ computes $\mathbf{A}_n = \mathbf{A} - \sum_{i=1}^{n-1} \mathbf{A}_i$. And for the consistency of the following queries by $\mathcal{A}$, we need to reprogram $\mathsf{QPRF}_{\mathsf{k}'_1}(\mathsf{H}'_1)$ at (s_n, n) such that $\mathsf{QPRF}_{\mathsf{k}'_1}(s_n, n) := \mathbf{A}_n$ (i.e., $\mathsf{H}'_1(s_n, n) := \mathbf{A}_n$). Note that the distribution of $\mathbf{A}_n$ are uniform, which follows that of $\mathbf{A}$. The formal simulation strategy is described in **Matrix Generation** part of Figure 14.

According to Theorem 4.8, $\mathcal{B}$ reprograms the random oracle H'_1 to make $\mathbf{A}_n \leftarrow \mathsf{H}'_1(s_n, n)$ will not be noticed by $\mathcal{A}$. Because the distribution of s_n are uniform, we have

$$|\Pr[\mathbf{G}_7] - \Pr[\mathbf{G}_6]| \leq \left(4 + \sqrt{2}\right) \sqrt{Q_h} 2^{-\frac{l_1^*}{2}} + 2(\varepsilon_{\mathsf{QPRF}} + \varepsilon_{\mathsf{Inj\text{-}QPRF}}).$$

$\mathbf{G}_8$ This experiment is identical to $\mathbf{G}_7$ except that $\mathcal{B}$ simply picks the random public key share $\boldsymbol{t}_n \xleftarrow{\$} R_q^k$ during the key generation phase, rather than computing $\boldsymbol{t}_n = \overline{\mathbf{A}}\boldsymbol{s}_n$ with $\boldsymbol{s}_n \xleftarrow{\$} S_\eta^{\ell+k}$. As $\mathbf{A}$ follows the uniform distribution over $R_q^{k\times\ell}$, if the adversary $\mathcal{A}$ can distinguish $\mathbf{G}_8$ and $\mathbf{G}_7$ then we can use $\mathcal{A}$ as a distinguisher that breaks $\mathsf{MLWE}_{q,k,\ell,\eta}$ assumption; hence we have

$$|\Pr[\mathbf{G}_8] - \Pr[\mathbf{G}_7]| \leq \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}}.$$

$\mathbf{G}_9$ This experiment is identical to $\mathbf{G}_8$, except with the generation of $\boldsymbol{t}_n$. Rather than sampling $\boldsymbol{t}_n \xleftarrow{\$} R_q^k$, $\mathcal{B}$ first choose $\boldsymbol{t} \xleftarrow{\$} R_q^k$, and send out a random $g'_n \xleftarrow{\$} \{0,1\}^{l_2}$. Then, after receiving all others random oracle commitments $g'_u \in \{0,1\}^{l_2}$, $\mathcal{B}$ can extract the adversary's corresponding committed shares $\boldsymbol{t}_1, ..., \boldsymbol{t}_{n-1} \in R_q^k$. As H_2 has been simulated by $\mathsf{Inj\text{-}QPRF}_{\mathsf{k}_2}$ in Construction 4.5, according to Theorem 4.7, this extraction can be efficiently done through using Algorithm 1. Furthermore, $\mathcal{B}$ computes $\boldsymbol{t}_n = \boldsymbol{t} - \sum_{i=1}^{n-1} \boldsymbol{t}_i$. And for the consistency of the following queries by $\mathcal{A}$, we need to reprogram $\mathsf{Inj\text{-}QPRF}_{\mathsf{k}_2}(\mathsf{H}_2)$ at $(\boldsymbol{t}_n, n)$ such that $\mathsf{Inj\text{-}QPRF}_{\mathsf{k}_2}(\boldsymbol{t}_n, n) := g'_n$(i.e., $\mathsf{H}_2(\boldsymbol{t}_n, n) := g'_n$). Note that the distribution of $\boldsymbol{t}_n$ are uniform, which follows that of $\boldsymbol{t}$. The formal simulation strategy is described in **Key Pair Generation** part of Figure 14.

According to Theorem 4.8, $\mathcal{B}$ reprograms the random oracle H_2 to make $g'_n \leftarrow \mathsf{H}_2(\boldsymbol{t}_n, n)$ will not be noticed by $\mathcal{A}$. Becausethe distribution of $\boldsymbol{t}_n$ are uniform, we have

$$|\Pr[\mathbf{G}_9] - \Pr[\mathbf{G}_8]| \leq \left(4 + \sqrt{2}\right) \sqrt{Q_h} 2^{-\frac{q^{kN}}{2}} + 2(\varepsilon_{\mathsf{QPRF}} + \varepsilon_{\mathsf{Inj\text{-}QPRF}}).$$

Up until now, notice that the key generation query is simulated according to SimGen_n in Figure 14. This implies that $\mathcal{B}$ can be fully simulated without using any secret key.

Based on this, our next goal is to show that in $\mathbf{G}_9$, the probability of $\mathcal{A}$ forging a valid message-signature pair $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$ is negligible in λ. In order to do this, we need to establish an efficient reduction: if $\mathcal{A}$ outputs a valid forge $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$, then $\mathcal{B}$ can solve some underlying hard problems. Particularly, we need to embed a challenge commitment key $\mathsf{ck} \leftarrow \mathsf{Eqv\text{-}CGen}(\mathsf{cpp}_{\mathsf{Eqv}})$ and an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, which is denoted as $[\mathbf{A}'|\mathbf{I}]$ with $\mathbf{A}' \xleftarrow{\$} R_q^{k\times(\ell+1)}$. As in $\mathbf{G}_9$ the combined public key $(\mathbf{A}, \boldsymbol{t})$ is uniformly distributed in $R_q^{k\times\ell} \times R_q^k$, replacing it with $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ instance doesn't change the view of the adversary at all, where $\mathbf{A}' := [\mathbf{A}|\boldsymbol{t}]$. Moreover, according to the simulation of H_3, it is guaranteed that ck follows the uniform distribution over $\mathsf{Eqv}\text{-}S_{\mathsf{ck}}$, which is perfectly indistinguishable from honestly generated $\mathsf{ck} \leftarrow \mathsf{Eqv\text{-}CGen}(\mathsf{cpp}_{\mathsf{Eqv}})$.

Below, we follow the proof idea of [57] for proving the strong unforgeability in $\mathbf{G}_9$, i.e., $\Pr[\mathbf{G}_9] \leq \mathsf{negl}(\lambda)$. Particularly, we first show that there is an efficient extractor Ext in Figure 12, such that given a valid message-signature pair $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$ in $\mathbf{G}_9$, $\mathsf{Ext}(\mathsf{pp}, \mathsf{pk}, \mu^*, \mathsf{Sig}^*)$ can output a solution for MSIS problem with overwhelming probability, just as formalized in the following Lemma C.2. And then, we bound the probability $\Pr[\mathbf{G}_9]$ by the union bound of two events happen: Ext succeeds and Ext fails.

Lemma C.2 *There exists an extractor* Ext *presented in Figure 12, such that if* $\mathcal{A}$ *could output a valid forge* $(\mu^*, \mathsf{Sig}^*) \notin \mathsf{MSset}$ *in* $\mathbf{G}_9$*, then* $\mathsf{Ext}(\mathsf{pp}, \mathsf{pk}, \mu^*, \mathsf{Sig}^*)$ *will output a solution for* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *problem except with probability* $(2(Q_s + 1)2^{-(t\log m)/2} + Q_s \cdot t \cdot \varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2} \cdot 2^{-t\cdot\vartheta})$*, where* ε'_{bind} *is the advantages of breaking* $\mathsf{Inv\text{-}TCOM}$ *for any adversary, and* ϑ *is the min-entropy of the output of* $\mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$.

Proof. According to the basic structure of valid forge signature Sig^*, for any $i \in [t]$, if there exists one different index $j \neq J_i^*$ such that $\boldsymbol{z}_{i,j}$ satisfies: (1) $\|\boldsymbol{z}_{i,j}\| \leq B_n$; (2) $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - c_{i,j}^*\boldsymbol{t}) = 1$, then we know

$$\begin{aligned}&\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i^* := \overline{\mathbf{A}}\boldsymbol{z}_i^* - c_{i,J_i^*}^*\boldsymbol{t})\\ =&\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - c_{i,j}^*\boldsymbol{t}) = 1,\end{aligned}$$

where $\mathsf{ck}^* \leftarrow \mathsf{H}_3(\mu^*, \mathsf{pk})$, $\mathsf{ck}'^* \leftarrow \mathsf{H}_4(\mu^*, \mathsf{pk})$, $c_{i,j}^* \leftarrow \mathsf{H}_0(i, j, \mu^*, \mathsf{pk}, \mathsf{ck}^*, \mathsf{ck}'^*)$ for all $i \in [t], j \in [m]$, $J_1^*||...||J_t^* \leftarrow \mathsf{H}_5(\mathsf{pk}, \mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{c_{i,j}^*\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t],j\in[m]})$, and $\boldsymbol{z}_{i,j} = \mathsf{Inv}_{\mathsf{ck}'^*}(\widetilde{\mathsf{com}}_{i,j}^*, \mathsf{td}')$ with $\mathsf{td}' \leftarrow \mathsf{Inv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Inv}}, r), r = \mathsf{QPRF}_{\mathsf{k}_4}(\mu^*, \mathsf{pk})$.

We know that if the above equality holds, then we have $\overline{\mathbf{A}}\boldsymbol{z}_i^* - c_{i,J_i^*}^*\boldsymbol{t} = \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - c_{i,j}^*\boldsymbol{t}$, from which we get

$$(\mathbf{A}|\mathbf{I}|\boldsymbol{t})\begin{pmatrix}\boldsymbol{z}_i^* - \boldsymbol{z}_{i,j}\\ c_{i,j}^* - c_{i,J_i^*}^*\end{pmatrix} = 0.$$

Recalling that $(\mathbf{A}'|\mathbf{I}) = (\mathbf{A}|\boldsymbol{t}|\mathbf{I})$ is an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ problem, we have found a valid solution if $\beta = \sqrt{(2B_n)^2 + 4\kappa}$, since $||\boldsymbol{z}_i^* - \boldsymbol{z}_{i,j}|| \leq 2B_n$ and $0 < ||c_{i,J_i^*}^* - c_{i,j}^*|| \leq \sqrt{4\kappa}$.

Then, similar to Theorem 18 in [57], we first define the following two events:

- E_1: The valid forge signature in $\mathbf{G}_9$ is malleable. This means if

$$\mathsf{Sig}^* = (\mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t],j\in[m]}, \{r_i^*\}_{i\in[t]}, \{(r'^*_{i,J_i^*}, \boldsymbol{z}^*_{i,J_i^*})\}_{i\in[t]})$$

  is a valid forge output by the adversary in $\mathbf{G}_6$. Then, there exists another signature

$$\hat{\mathsf{Sig}}^* = (\mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^*\}_{i\in[t],j\in[m]}, \{r_i^*\}_{i\in[t]}, \{(\hat{r}'^*_{i,J_i^*}, \hat{\boldsymbol{z}}^*_{i,J_i^*})\}_{i\in[t]})$$

  is valid too. But the differences between Sig^* and $\hat{\mathsf{Sig}}^*$ are only on the pairs $(r'^*_i, \boldsymbol{z}^*_i)$ and $(\hat{r}'^*_i, \hat{\boldsymbol{z}}^*_i)$.
- E_2: The valid forge signature in $\mathbf{G}_9$ can be only verified successfully for $\boldsymbol{z}_{i,j} = \mathsf{Inv}_{\mathsf{tck}'^*}(\widetilde{\mathsf{com}}_{i,j}, \mathsf{td}')$, with $j = J_i^*$, where $\mathsf{td}' \leftarrow \mathsf{H}'_4(\mu^*, \mathsf{pk})$. According to the binding property of Eqv-TCOM, this means the following two conditions happen simultaneously:

$$\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i^* := \overline{\mathbf{A}}\boldsymbol{z}_i^* - c^*_{i,J_i^*}\boldsymbol{t}) = 1,$$

$$\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - c^*_{i,j}\boldsymbol{t}) \neq 1, \text{ for } j \neq J_i^*.$$

Intuitively, E_1 implies that the forged signature is computed from one of the simulated signatures from MSset. E_2 implies that for each com_i with $i \in [t]$, there are exactly one position $j \in [m]$ such that $z_{i,j}$ can be verified as the valid response. Clearly, if the above defined events E_1, E_2 do not happen and binding property of Eqv-TCOM holds, then the above extraction by Ext should be successful. Particularly, it holds

$$\begin{aligned}\Pr[\mathsf{Ext}\text{ succeeds}] &\geq 1 - \Pr[E_1 \cup E_2] \\ &\geq 1 - (\Pr[E_1] + \Pr[E_2]).\end{aligned}$$

Thus, it suffices to show the upper bounds of $\Pr[E_1]$ and $\Pr[E_2]$ are negligible in λ, i.e., $\Pr[E_1] \leq Q_s \cdot t \cdot \varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2} \cdot 2^{-t\cdot\vartheta}$ and $\Pr[E_2] \leq 2(Q_h+1) \cdot 2^{-(t\cdot\log m)/2}$, in the following Lemmas C.3 and C.4. □

Lemma C.3 (Non-malleability of valid signature in $\mathbf{G}_9$) *Suppose* Inv-TCOM *is secure and ε'_{bind} is the advantage of breaking its binding for any adversary, and let Q_s be the number of signature queries conducted by $\mathcal{A}$ in $\mathbf{G}_9$, then*

$$\Pr[E_1] \leq Q_s \cdot t \cdot \varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2} \cdot 2^{-t\cdot\vartheta}.$$

Proof (Sktech). For $\mathbf{G}_9$, we define the event E_1 more formally as follows. Suppose $(\mu, \mathsf{Sig}) = (\mu, \{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{r_i\}_{i\in[t]}, \{(r'_{i,J_i}, \boldsymbol{z}_{i,J_i})\}_{i\in[t]}) \in$ MSset to be one of simulated message-signature pairs output by $\mathcal{B}$. Then, as a malleable forgery, we assume that the adversary outputs a new valid message-signature pair $(\mu^*, \mathsf{Sig}^*) \notin$ MSset such that $\mathsf{QDS}_2.\mathsf{Ver}(\mathsf{pk}, \mu^*, \mathsf{Sig}^*) = 1$, with

$(\mu^*, \mathsf{Sig}^*) = (\mu, \{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{r_i\}_{i\in[t]}, \{({r'}^*_{i,J_i}, \boldsymbol{z}^*_{i,J_i})\}_{i\in[t]})$. In other words, such a valid message-signature forgery (μ^*, Sig^*) differs from the related $(\mu, \mathsf{Sig}) \in \mathsf{MSset}$ only in the $(\boldsymbol{r}', \boldsymbol{z})$-components.

According to $\mathbf{G}_3$, we know that H_5 needs to be reprogrammed for each signature generation query. Thus, for certain $(\mu, \mathsf{Sig}) \in \mathsf{MSset}$, we use H'_5 to denote the corresponding state of H_5 just after $\mathcal{B}$ outputting this Sig for μ. Similarly, we use H''_5 to denote the final state of H_5, after $\mathcal{B}$ receiving the forged message-signature pair (μ^*, Sig^*) from the adversary $\mathcal{A}$. This means when verifying the validness of the forged signature, we will use $\mathsf{H}''_5(\cdot)$ to compute $J^*_1||...||J^*_t$. But in order to verify the validness of the simulated signature Sig output by $\mathcal{B}$, we use $\mathsf{H}'_5(\cdot)$ to compute $J^*_1||...||J^*_t$. Here for simplicity, we denote $\pi_{\mathsf{half}} := (\{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]})$. And thus, we can rewrite $J^*_1||...||J^*_t = \mathsf{H}''_5(\mathsf{pk}, \mu, \pi_{\mathsf{half}})$ and $J_1||...||J_t = \mathsf{H}'_5(\mathsf{pk}, \mu, \pi_{\mathsf{half}})$.

Below, we analyze the event E_1. Let D_1 be the event $\mathsf{H}'_5(\mathsf{pk}, \mu, \pi_{\mathsf{half}}) = \mathsf{H}''_5(\mathsf{pk}, \mu, \pi_{\mathsf{half}})$, and $\bar{D}_1$ be the event $\mathsf{H}'_5(\mathsf{pk}, \mu, \pi_{\mathsf{half}}) \neq \mathsf{H}''_5(\mathsf{pk}, \mu, \pi_{\mathsf{half}})$.

According to the total probability, it holds

$$\begin{aligned}\Pr[E_1] &= \Pr[E_1|D_1]\Pr[D_1] + \Pr[E_1|\bar{D}_1]\Pr[\bar{D}_1]\\ &\leq \Pr[E_1|D_1] + \Pr[\bar{D}_1].\end{aligned}$$

Conditioned on D_1, E_1 implies that there exists at least one $i \in [t]$ such that $\mathsf{Inv\text{-}Commit}_{\mathsf{ck}'}(\boldsymbol{z}^*_i, {r'}^*_{i,J_i}) = \mathsf{Inv\text{-}Commit}_{\mathsf{ck}'}(\boldsymbol{z}_i, r'_{i,J_i}) = \widetilde{\mathsf{com}}_{i,J_i}$, but $(\boldsymbol{z}^*_i, {r'}^*_{i,J_i}) \neq (\boldsymbol{z}_i, r'_{i,J_i})$. Clearly, this contradicts with the binding property of $\mathsf{Inv\text{-}TCOM}$ scheme. Thus, it holds $\Pr[E_1|D_1] \leq Q_s \cdot t \cdot \varepsilon'_{bind}$, since the adversary has conducted signature queries for Q_s times.

On the other hand, the event $\bar{D}_1$ implies that H_5 has been reprogrammed at the same point $x := (\mathsf{pk}, \mu, \pi_{\mathsf{half}})$ for two times, during all the Q_s times signature queries. According to the fact (1) $\mathsf{com}_i := \sum_{u\in[n]} \mathsf{com}^{(u)}_i$ with $\mathsf{com}^{(n)}_i \leftarrow \mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ and (2) the output of $\mathsf{Eqv\text{-}TCommit}_{\mathsf{tck}}(\mathsf{td})$ has sufficient min-entropy ϑ, it holds that the min-entropy of x is at least $t \cdot \vartheta$. Thus, this happens with probability at most $\frac{Q_s(Q_s+1)}{2} \cdot 2^{-t\cdot\vartheta}$.

As a corollary , it holds

$$\Pr[E_1] \leq Q_s \cdot t \cdot \varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2} \cdot 2^{-t\cdot\vartheta}.$$

□

Lemma C.4 ($\boldsymbol{E_2}$) *Suppose after making Q_s times signature queries for μ_i in $\mathbf{G}_6$, $\mathcal{A}$ gives a forgery Sig^* such that $\mathsf{QDS}_2.\mathsf{Ver}(\mathsf{pk}, \mu^*, \mathsf{Sig}^*) = 1$, where $\mathsf{Sig}^* = (\{\mathsf{com}^*_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}^*_{i,j}\}_{i\in[t],j\in[m]}, \{r^*_i\}_{i\in[t]}, \{{r'}^*_i\}_{i\in[t]}, \{\boldsymbol{z}^*_i\}_{i\in[t]})$. Then*

$$\Pr[E_2] \leq 2(Q_s+1)2^{-(t\cdot\log m)/2},$$

Proof (Sktech). This proof is almost identical to the Lemma 17 of [57], but with "$\mathsf{Inv\text{-}Commit}$, Inv" instead of G, G^{-1}, respectively, i.e., we replace G with a homomorphic trapdoor commitment that can be inverted. And according to the computational binding of $\mathsf{Eqv\text{-}TCOM}$, we can use $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - c_{i,j}\boldsymbol{t}) = 1$ and $||\boldsymbol{z}_{i,j}|| \leq B_n$ to represent the validness of Σ-protocol in Lemma 17 of [57]. □

According to Lemma C.2, if the extraction is successful, $\mathcal{B}$ can solve the $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ problem with $\beta = \sqrt{(2B_n)^2 + 4\kappa}$.

Thus, we get

$$\Pr[\mathsf{ExSucess}] \leq \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}.$$

and

$$\Pr[\mathsf{ExFail}] \leq 2(Q_s+1)2^{-(t\log m)/2} + Q_s \cdot t \cdot \varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2} \cdot 2^{-t\cdot\vartheta}.$$

Finally, we know

$$\begin{aligned}
\Pr[\mathbf{G}_9] &= \Pr[\mathbf{G}_9|\mathsf{ExFail}]\Pr[\mathsf{ExFail}] + \Pr[\mathbf{G}_9|\mathsf{ExSucess}]\Pr[\mathsf{ExSucess}] \\
&\leq \Pr[\mathsf{ExFail}] + \Pr[\mathsf{ExSucess}] \\
&\leq 2(Q_s+1)2^{-(t\log m)/2} + Q_s \cdot t \cdot \varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2} \cdot 2^{-t\cdot\vartheta} + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}.
\end{aligned}$$

Summing up all above analysis, we conclude that the statement of theorem holds. □

D Two Round Multi-Signature from lattices in the QROM

In this section we describe our two-round multi-signature scheme QMS_2 in the key-verification model. We remark that with the help of the multi-proof straight-line extractable $\mathsf{NIZKPoK}$ in the key generation stage, our QMS_2 can be proven secure relying on essentially the same idea as QDS_2. And the main difference from n-out-of-n signature is that, the protocol requires no interactive key generation at all, and instead for each signing execution a party receives a set of public keys Ł together with a message to be signed. Particularly, our construction of two-round multi-signature $\mathsf{QMS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{KVer})$ is formally specified in Figures 17, 18, 19. As the number of participants may change for each signing attempt, in this section we define n to be the maximum number of signers allowed in a single execution of signing protocol, i.e., only Ł of cardinality at most n is a valid input. Without loss of generality, we assume that each signer assign the index n to itself, and consider other signers' indices as $1, ..., n'-1$, where $n' = |Ł| \leq n$. As we will use the multi-proof straight-line extractable $\mathsf{NIZKPoK}$ in the QROM [19] as a building block, we first recall it before presenting the formal construction of our QMS.

D.1 Non-interactive Zero-knowledge Proof of Knowledge

Let's recall the notion of non-interactive zero-knowledge proof of knowledge ($\mathsf{NIZKPoK}$) system.

Definition D.1 ([19]) *Let $\mathcal{R}$ be a relation (and $\mathcal{L}_{\mathcal{R}}$ is the related language). A non-interactive proof system Π for $\mathcal{R}$ (or $\mathcal{L}_{\mathcal{R}}$) is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *having the following interfaces (where 1^λ are implicit inputs to* $\mathsf{Prove}, \mathsf{Verify}$*):*

- $\mathsf{Setup}(1^\lambda)$: *given a security parameter λ, outputs a string* CRS.
- $\mathsf{Prove}(\mathsf{CRS}, x, w)$*: given a string* CRS *and a statement-witness pair $(x, w) \in \mathcal{R}$ (or w is the witness for $x \in \mathcal{L}_{\mathcal{R}}$), outputs a proof π.*
- $\mathsf{Verify}(\mathsf{CRS}, x, \pi)$*: given a string* CRS*, a statement x, and a proof π, either accepts or rejects.*

A secure $\mathsf{NIZKPoK}$ should have four properties: Completeness, Soundness, and Zero-knowledge, Straight-line extractability.

- Completeness: for every $(x, w) \in \mathcal{R}$ and every λ, $\mathsf{Verify}(\mathsf{CRS}, x, \pi)$ accepts with probability 1, over the choice of $\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$ and $\pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, x, w)$.
- Soundness: let $\mathcal{L}_{\mathcal{R}}$ be the language defined by relation $\mathcal{R}$. For any PPT adversary $\mathcal{A}$,

  $$\mathsf{Pr}_{\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)}\left[\exists x \ s.t. \pi^* \leftarrow \mathcal{A}(\mathsf{CRS}, x) : \mathsf{Verify}(\mathsf{CRS}, x, \pi^*) \text{ accepts} \wedge x \notin \mathcal{L}_{\mathcal{R}}\right] \leq \mathsf{negl}(\lambda).$$

- Zero-Knowledge: There exists two PPT algorithms $(\mathsf{SimSetup}, \mathsf{SimProve})$, such that, for any PPT adversary $\mathcal{A}$ we have $|\mathsf{Pr}[\mathcal{A} \ wins] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$ in the following game:
  1. The challenger samples $(\widehat{\mathsf{CRS}}, \mathsf{tk}) \leftarrow \mathsf{SimSetup}(1^\lambda)$ such that $\widehat{\mathsf{CRS}}$ is indistinguishable from CRS output by Setup, and gives the simulated $\widehat{\mathsf{CRS}}$ to $\mathcal{A}$.
  2. The adversary $\mathcal{A}$ chooses $(x, w) \in \mathcal{R}$ and gives these to the challenger.
  3. The challenger samples $\pi_0 \leftarrow \mathsf{Prove}(\mathsf{CRS}, x, w), \pi_1 \leftarrow \mathsf{SimProve}(\widehat{\mathsf{CRS}}, x, \mathsf{tk}), b \leftarrow \{0, 1\}$ and gives π_b to $\mathcal{A}$.
  4. The adversary $\mathcal{A}$ outputs a bit b' and wins if $b' = b$.

Notice that in the above zero-knowledge game, if we allow the adversary $\mathcal{A}$ to choose any polynomial numbers of $(x_i, w_i) \in \mathcal{R}$, and all the resulting $\{\pi_{i,0}\}$ and $\{\pi_{i,1}\}$ are still indistinguishable, we say that Π is a multi-proof $\mathsf{NIZKPoK}$ system.

Moreover, we consider straight-line extractability. Here, the adversary can pick the statements adaptively. In order to perform extraction in this stronger setting, the common reference string is simulated and the corresponding trapdoor is provided to the extractor.

Definition D.2 (Multi-Proof Straight-Line Extractability [19]) *An* $\mathsf{NIZKPoK}$ *system is multi-proof straight-line extractable, if there exists a PPT oracle simulator* $\mathsf{SimSetup}$ *and a PPT extractor* Ext *with the following properties:*

CRS indistinguishability. *For any PPT adversary $\mathcal{A}$, we have*

$$\begin{aligned}\mathsf{Adv}(\mathcal{A}) := \Big| & \Pr[\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda) : \mathcal{A}(\mathsf{CRS}) = 1] \\ & - \Pr[(\widehat{\mathsf{CRS}}, \mathsf{tk}) \leftarrow \mathsf{SimSetup}(1^\lambda) : \mathcal{A}(\widehat{\mathsf{CRS}}) = 1]\Big| \leq \mathsf{negl}(\lambda).\end{aligned}$$

Straight-Line Extractability. *There exists constants c, e_1, e_2 and polynomial $p(\lambda)$ such that for any $Q_H = \mathsf{poly}(\lambda)$, $Q_s = \mathsf{poly}(\lambda)$ and* PPT *adversary $\mathcal{A}$ that makes at most Q_H random oracle queries, and generates at most Q_s statement proof pairs with*

$$\Pr\Big[(\widehat{\mathsf{CRS}}, \mathsf{tk}) \leftarrow \mathsf{SimSetup}(1^\lambda), \{(x_i, \pi_i)\}_{i\in[Q_s]} \leftarrow \mathcal{A}^{H(\cdot)}(\widehat{\mathsf{CRS}}) : \forall i \in [Q_s], \mathsf{Verify}^{H(\cdot)}(\widehat{\mathsf{CRS}}, x_i, \pi_i) = 1\Big] \geq \mu(\lambda),$$

we have

$$\Pr\Big[(\widehat{\mathsf{CRS}}, \mathsf{tk}) \leftarrow \mathsf{SimSetup}(1^\lambda), \{(x_i, \pi_i)\}_{i\in[Q_s]} \leftarrow \mathcal{A}^{H(\cdot)}(\widehat{\mathsf{CRS}}), \{w_i \leftarrow \mathsf{Ext}(1^\lambda, Q_H, Q_s, 1/\mu, \mathsf{tk}, x_i, \pi_i)\}_{i\in[Q_s]} : \forall i \in [Q_s], (x_i, w_i) \in \mathfrak{R} \wedge \mathsf{Verify}^{H(\cdot)}(\widehat{\mathsf{CRS}}, x_i, \pi_i) = 1\Big] \geq \frac{1}{2} \cdot \mu(\lambda) - \mathsf{negl}(\lambda).$$

Moreover, the running time of Ext *is upper bounded by $Q_H^{e_1} \cdot Q_s^{e_2} \cdot \frac{1}{\mu^c} \cdot p(\lambda)$.*

Below, we recall the instantiation of NIZKPoK for MSIS relation. Particularly, for the following language

$$\mathcal{L}_{B,q} = \Big\{(\bar{\mathbf{A}}, \boldsymbol{u}) \in R_q^{k\times(\ell+k)} \times R_q^k : \exists\, \boldsymbol{x} \in R^{\ell+k} \text{ such that } 0 < \|\boldsymbol{x}\| \leq B \text{ and } \bar{\mathbf{A}} \cdot \boldsymbol{x} = \boldsymbol{u}\Big\},$$

there are practical multi-proof straight-line extractable NIZKPoK systems for $L_{B,q}$, according to [19].

D.2 Construction

Protocol $\mathsf{QMS}_2.\mathsf{Gen}(\mathsf{pp})$

The protocol is parameterized by public parameters described in Table 2, matrix $\overline{\mathbf{A}}$, together with the common reference string CRS of the used multi-proof NIZKPoK system Π. Then, conduct the following steps:

1. Sample a secret key shares $\boldsymbol{s}_n \xleftarrow{\$} S_\eta^{\ell+k}$ and compute a public key share $\boldsymbol{t}_n := \overline{\mathbf{A}}\boldsymbol{s}_n$;
2. Runs $\Pi.\mathsf{Prove}(\mathsf{CRS}, \overline{\mathbf{A}}, \boldsymbol{t}_n, \boldsymbol{s}_n)$ to output a NIZKPoK proof π_n as an appendix of public key.

If the protocol does not abort, P_n obtain $(\mathsf{sk}_n, \mathsf{pk}_n) = (\boldsymbol{s}_n, (\boldsymbol{t}_n, \pi_n))$ as local output.

Fig. 17. Gen Protocol of Our Two-Round Multi-Signature Scheme

As the construction and proof of QMS_2 is almost same to these of QDS_2, below we highlight the differences in red color.

Given a multi-proof straight-line extractable NIZKPoK system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for $\mathcal{L}_{B,q}$ just as defined in Definition D.1, we make a brief overview of our QMS_2 scheme as follows.

– The Setup works most like the one for QDS_2, but it additionally outputs a matrix $\overline{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$ as part of public parameters, so we assume that $\overline{\mathbf{A}}$ is generated by a trusted third party. And the input lengths of QROM is changed and we show these as follows.
  - $l_0^* = \log(m \cdot t \cdot |\mathcal{M}|) + k \cdot N \cdot \log q \cdot (n+1) + \log|\mathsf{Eqv}\text{-}S_{\mathsf{ck}}| + \log|\mathsf{Inv}\text{-}S_{\mathsf{ck}}|$
  - $l_3^* = l_4^* = \log|\mathcal{M}| + n \cdot k \cdot N \cdot \log q$
  - $l_5^* = nk \cdot N \log q + \log|\mathcal{M}| + t\log|\mathsf{Eqv}\text{-}S_{\mathsf{com}}| + mt\log(2N\kappa|\mathsf{Inv}\text{-}S_{\mathsf{com}}|)$

  Besides, the Setup algorithm runs $\Pi.\mathsf{Setup}$ to output the common string reference CRS.
– The Gen is formally specified in Figure 17, which consists of the following two stages:
  - Samples $\boldsymbol{s}_n \xleftarrow{\$} S_\eta^{\ell+k}$, and computes $\boldsymbol{t}_n = \overline{\mathbf{A}}\boldsymbol{s}_n \in R_q^k$.
  - Takes $\mathsf{CRS}, \overline{\mathbf{A}}, \boldsymbol{t}_n, \boldsymbol{s}_n$ as input, and runs $\Pi.\mathsf{Prove}(\mathsf{CRS}, \overline{\mathbf{A}}, \boldsymbol{t}_n, \boldsymbol{s}_n)$ to output a NIZKPoK proof π_n as an appendix of public key.

  Finally, the algorithm outputs $(\mathsf{pk}, \mathsf{sk}) = ((\boldsymbol{t}_n, \pi_n), \boldsymbol{s}_n)$.
– The signing protocol Sign and verification Ver are described in Figures 18 and 19. The main differences from $\mathsf{QDS}_2.\mathsf{Sign}$ and $\mathsf{QDS}_2.\mathsf{Ver}$ is that at the beginning stages of $\mathsf{QDS}_2.\mathsf{Sign}$ and $\mathsf{QDS}_2.\mathsf{Ver}$, each participant need to first verify the well-formedness of other participant's public keys.
– The key verification algorithm KVer is just run the verification algorithm $\Pi.\mathsf{Verify}$ of the NIZKPoK system Π.

D.3 Correctness and Security

As the correctness of QMS2 is quite similar to that of Theorem 5.1, here we omit it for simplicity. Below, we just focus on the security. Similar to QDS_2 in Section 5, here we also focus on the strong unforgeability, i.e., we show that our QMS_2 is SUF-CMA secure.

Theorem D.3 *Suppose the trapdoor commitment schemes* Inv-TCOM *and* Eqv-TCOM *are secure, additively homomorphic, have uniform keys and uniform commitment. Particularly, the output of* $\mathsf{Eqv}\text{-}\mathsf{TCommit}_{\mathsf{tck}}(\mathsf{td})$ *has sufficient min-entropy* ϑ. *Suppose* $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify}))$ *is a multi-proof straight-line extractable* NIZKPoK *system for* $\mathcal{L}_{B,q}$, *just as defined in Definition D.1. And suppose there exists* QPRF *that can be programable and invertible simultaneously. For any quantum polynomial-time adversary* $\mathcal{A}$ *that initiates a single key generation protocol by querying* $\mathcal{O}_n^{\mathsf{QMS}_2}$ *with sid* $= 0$, *initiates* Q_s *signature generation protocols by querying* $\mathcal{O}_n^{\mathsf{QMS}_2}$ *with sid* $\neq 0$, *and makes* Q_h *quantum superpositions queries to random oracle* $\mathsf{H}_0, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5$, *the protocol* QMS_2 *of Figures 17, 18, 19 is* QMS-SUF-CMA *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions in the* QROM, *where* $\beta = \sqrt{(2B_n)^2 + 4\kappa + \eta^2(4\kappa\cdot(\ell+k))}$. *Concretely, using other parameters specified in Table 2, the advantage of* $\mathcal{A}$ *is bounded as follows.*

Protocol $\mathsf{QMS}_2.\mathsf{Sign}(sid, \mathsf{sk}_n, \mathsf{pk}_n, \mu, \mathrm{L})$

The protocol is parameterized by public parameters described in Table 2 and matrix $\overline{\mathbf{A}}$, and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^{l_0^*} \to C$, $\mathsf{H}_3 : \{0,1\}^{l_3^*} \to \mathsf{Eqv}\text{-}S_{\mathsf{ck}}$, $\mathsf{H}_4 : \{0,1\}^{l_4^*} \to \mathsf{Inv}\text{-}S_{\mathsf{ck}}$ and $\mathsf{H}_5 : \{0,1\}^{l_5^*} \to \{0,1\}^{l_5}$. The protocol assumes that $\mathsf{QMS}_2.\mathsf{Gen}(\mathsf{pp})$ has been previously invokes. If a party halts with $\perp$ at any point, then all $\mathsf{Sign}(sid, \mathsf{sk}_n, \mathsf{pk}_n, \mu, \mathrm{L})$ executions are aborted.

Inputs

1. P_n receives a unique sessions ID sid, $\mathsf{sk}_n = \boldsymbol{s}_n$, $\mathsf{pk} = \boldsymbol{t}_n$ and message $\mu \in M$ as input and a list of public keys L as input. If $n' := |\mathrm{L}| > n$ or $\boldsymbol{t}_n \notin \mathrm{L}$ or $\Pi.\mathsf{Verify}(\mathsf{CRS}, \bar{\mathbf{A}}, \boldsymbol{t}_j, \pi_j) = 0$ for certain $(\boldsymbol{t}_j, \pi_j) \in \mathrm{L}$, then send out $\perp$. Otherwise parse L as $\{\boldsymbol{t}_1, \ldots \boldsymbol{t}_{n'-1}, \boldsymbol{t}_n\}$.
2. P_n verifies that sid has not been used before (if it has been, the protocol is not executed).
3. P_n locally computes a per-message commitment key $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathrm{L})$, $\mathsf{ck}' \leftarrow \mathsf{H}_4(\mu, \mathrm{L})$.

Signature Generation P_n works as follows:

1. Compute the first group messages as follows:
   (a) for $i = 1$ to t; conduct as follows:
      i. Sample $y_i^{(n)} \leftarrow D_s^{\ell+k}$ and compute $\boldsymbol{w}_i^{(n)} := \overline{\mathbf{A}} y_i^{(n)}$.
      ii. Compute $\mathsf{com}_i^{(n)} \leftarrow \mathsf{Eqv}\text{-}\mathsf{Commit}_{\mathsf{ck}}(\boldsymbol{w}_i^{(n)}, r_i^{(n)})$ with $r_i^{(n)} \xleftarrow{\$} D(\mathsf{Eqv}\text{-}S_r)$.
      iii. for $j = 1$ to m, conduct as follows:
         A. Derive challenges $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{ck}, \mathsf{ck}', \mathrm{L})$.
         B. Computes signature shares $\boldsymbol{z}_{i,j}^{(n)} = c_{i,j}\boldsymbol{s}_n + \boldsymbol{y}_i^{(n)}$.
         C. Run the rejection sampling $\mathsf{Rej}(\boldsymbol{z}_{i,j}^{(n)}, c_{i,j}\boldsymbol{s}_n, \sigma) \to \{0,1\}$.
      iv. If the above rejection sampling algorithm outputs 0 for certain $j \in [m]$, then send out $\perp$, and go to the step i.
   (b) Compute $\widetilde{\mathsf{com}}_{i,j}^{(n)} \leftarrow \mathsf{Inv}\text{-}\mathsf{Commit}_{\mathsf{ck}'}(\boldsymbol{z}_{i,j}^{(n)}, r_{i,j}'^{(n)})$ where $r_{i,j}'^{(n)} \xleftarrow{\$} D(\mathsf{Inv}\text{-}S_r)$ for all $i \in [t], j \in [m]$.
   (c) Send out $(\{\mathsf{com}_i^{(n)}\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^{(n)}\}_{i\in[t],j\in[m]})$.
2. Upon receiving $(\{\mathsf{com}_i^{(u)}\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}^{(n)}\}_{i\in[t],j\in[m]})$ for all $u \in [n'-1]$ compute the signature shares as follows:
   (a) For each $u \in [n'-1]$, derive per-user challenges $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{ck}, \mathsf{ck}', \mathrm{L})$ where $i \in [t], j \in [m]$. Set $\mathsf{com}_i := \sum_{u\in[n'-1]} \mathsf{com}_i^{(u)} + \mathsf{com}_i^{(n)}$ and $\widetilde{\mathsf{com}}_{i,j} := \sum_{u\in[n'-1]} \widetilde{\mathsf{com}}_{i,j}^{(u)} + \widetilde{\mathsf{com}}_{i,j}^{(n)}$ for all $i \in [t], j \in [m]$.
   (b) Get challenges $J_1||\ldots||J_t \leftarrow \mathsf{H}_5(\mathrm{L}, \mu, \{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]})$.
   (c) Send out $(\{\boldsymbol{z}_{i,J_i}^{(n)}\}_{i\in[t]}, \{r_i^{(n)}\}_{i\in[t]}, \{r_{i,J_i}'^{(n)}\}_{i\in[t]})$.
3. Upon receiving $(\{\boldsymbol{z}_{i,J_i}^{(u)}\}_{i\in[t]}, \{r_i^{(u)}\}_{i\in[t]}, \{r_{i,J_i}'^{(u)}\}_{i\in[t]})$ for all $u \in [n'-1]$ compute the combined signature as follows:
   (a) For each $u \in [n'-1]$, compute J_i and c_{i,J_i} as before, and reconstruct $\boldsymbol{w}_i^{(u)} := \overline{\mathbf{A}}\boldsymbol{z}_{i,J_i}^{(u)} - c_{i,J_i}\boldsymbol{t}_u$ and validate the signature shares

$$||\boldsymbol{z}_{i,J_i}^{(u)}|| \leq B, \mathsf{Eqv}\text{-}\mathsf{Open}_{\mathsf{ck}}(\mathsf{com}_i^{(u)}, r_i^{(u)}, \boldsymbol{w}_i^{(u)}) = 1$$

   and

$$\mathsf{Inv}\text{-}\mathsf{Open}_{\mathsf{ck}'}(\widetilde{\mathsf{com}}_{i,J_i}^{(u)}, r_{i,J_i}'^{(u)}, \boldsymbol{z}_{i,J_i}^{(u)}) = 1.$$

   for all $i \in [t]$. If the check fails for some u then send out $\perp$.
   (b) Compute $\boldsymbol{z}_{i,J_i} := \sum_{u\in[n'-1]} \boldsymbol{z}_{i,J_i}^{(u)} + \boldsymbol{z}_{i,J_i}^{(n)}$, $r_i := \sum_{u\in[n'-1]} r_i^{(u)} + r_i^{(n)}$ and $r'_{i,J_i} := \sum_{u\in[n'-1]} r_{i,J_i}'^{(u)} + r_{i,J_i}'^{(n)}$ for all $i \in [t]$.

If the protocol does not abort, P_n obtains a signature $(\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_{i,J_i}\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r'_{i,J_i}\}_{i\in[t]})$ as local output.

Fig. 18. Sign Protocol of Our Two-Round Multi-Signature Scheme

$$\begin{aligned}
\mathbf{Adv}_{\mathsf{QDS}_2}^{\mathsf{QDS\text{-}SUF\text{-}CMA}}(\mathcal{A}) \leq\ & 2\varepsilon_{\mathsf{Inj\text{-}QPRF}} + 5\varepsilon_{\mathsf{QPRF}} + e(Q_h + Q_s + 1)\Big[(Q_h + Q_s)(\varepsilon_{\mathsf{td}} + \varepsilon_{\mathsf{td}'}) \\
& + 2(Q_h + Q_s)\cdot\varepsilon_{\mathsf{QPRF}} + (4+\sqrt{2})\sqrt{Q_h}2^{\frac{-t\cdot\vartheta}{4}} + 2\varepsilon_{\mathsf{QPRF}} + t\cdot(m-1)\cdot\mathsf{negl}(\lambda) \\
& + t\cdot Q_s\cdot\varepsilon_{\mathsf{Rej}} + (4+\sqrt{2})\sqrt{Q_h}(2^{\frac{-q^{klN}}{4}} + 2^{\frac{-q^{kN}}{4}}) + 4(\varepsilon_{\mathsf{QPRF}} + \varepsilon_{\mathsf{Inj\text{-}QPRF}}) \\
& + \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}} + 2(Q_h+1)2^{-(t\log m)/2} + Q_s\cdot t\cdot\varepsilon'_{bind} + \frac{Q_s(Q_s+1)}{2}\cdot 2^{-t\cdot\vartheta} \\
& + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}\Big]
\end{aligned}$$

Protocol $\mathsf{QMS}_2.\mathsf{Ver}(\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r'_{i,J_i}\}_{i\in[t]}), \mu, \mathrm{Ł})$

Upon receive a message μ, signature $\mathsf{Sig} = (\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r'_{i,J_i}\}_{i\in[t]})$, and a set of public keys Ł, if $|\mathrm{Ł}| > n$ then reject the signature. Otherwise work as follows:

1. Check public key in Ł, i.e., run Π.Verify algorithm. If there exist certain $(\boldsymbol{t}_j, \pi_j) \in \mathrm{Ł}$ such that $\Pi.\mathsf{Verify}(\mathsf{CRS}, \bar{\mathbf{A}}, \boldsymbol{t}_j, \pi_j) = 0$, then send out $\perp$. Otherwise, conduct the following steps.
2. Generate commitment keys $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \mathrm{Ł})$ and $\mathsf{ck}' \leftarrow \mathsf{H}_4(\mu, \mathrm{Ł})$, Compute challenges $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{ck}, \mathsf{ck}', \mathrm{Ł})$ Then Compute $J_1||...||J_t \leftarrow \mathsf{H}_5(\mathrm{Ł}, \mu, \{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j,u}\}_{i\in[t],j\in[m],u\in[n']}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]})$, where $n' = |\mathrm{Ł}|$. Compute $\boldsymbol{t} = \sum_{u\in \mathrm{Ł}} \boldsymbol{t}_u$.
3. Perform the checks as follows:
   (a) for $i = 1$ to t do:
       Check that $c_{i,1}, ..., c_{i,m}$ pairwise distinct.
   (b) for $i = 1$ to t do:
       check that $||z_i|| \leq B_n$.
   (c) for $i = 1$ to t do:
       Reconstruct $\boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_i - c_{i,J_i}\boldsymbol{t}$, check $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}}(\mathsf{com}_i, r_i, \boldsymbol{w}_i) = 1$.
   (d) for $i = 1$ to t do:
       Check $\mathsf{Inv\text{-}Open}_{\mathsf{ck}'}(\widetilde{\mathsf{com}}_{i,J_i}, r'_{i,J_i}, \boldsymbol{z}_i) = 1$.

If all checks succeed then return 1, otherwise, return 0.

Fig. 19. Ver Algorithm of Our Two-Round Multi-Signature Scheme

As this proof is almost same as the proof for QDS_2 in Theorem 5.2, for simplicity of presentation, we just highlight the difference: how to use $\mathsf{Ext}(\mathsf{pp}, \mu^*, \mathsf{Sig}^*)$ in Figure 20 to output a solution for $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ problem for a different β, with the help of multi-proof NIZKPoK system Π.

According to the basic structure of valid forge signature Sig^*, for any $i \in [t]$, if there exists one different index $j \neq J_i^*$ such that $\boldsymbol{z}_{i,j}$ satisfies: (1) $\|\boldsymbol{z}_{i,j}\| \leq B_n$; (2) $\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - \sum_{u\in[n^*-1]} c_{i,j}\boldsymbol{t}_u - c_{i,j}\boldsymbol{t}_n)$ $= 1$, then we know

$$\begin{aligned}&\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i^* := \overline{\mathbf{A}}\boldsymbol{z}^*_{i,J_i^*} - \sum_{u\in[n^*-1]} c^*_{i,J_i^*}\boldsymbol{t}_u - c^*_{i,J_i^*}\boldsymbol{t}_n)\\ =&\mathsf{Eqv\text{-}Open}_{\mathsf{ck}^*}(\mathsf{com}_i^*, r_i^*, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - \sum_{u\in[n^*-1]} c^*_{i,j}\boldsymbol{t}_u - c^*_{i,j}\boldsymbol{t}_n),\end{aligned}$$

where $\mathsf{ck}^* \leftarrow \mathsf{H}_3(\mu^*, \mathrm{Ł}^*)$, $\mathsf{ck}'^* \leftarrow \mathsf{H}_4(\mu^*, \mathrm{Ł}^*)$, $c^*_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu^*, \mathsf{ck}^*, \mathsf{ck}'^*, \mathrm{Ł}^*)$ for all $i \in [t], j \in [m]$, $J_1^*||...||J_t^* \leftarrow \mathsf{H}_5(\mu^*, \{\mathsf{com}_i^*\}_{i\in[t]}, \{c^*_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}^*_{i,j}\}_{i\in[t],j\in[m]},$ $\mathrm{Ł}^*)$, and $\boldsymbol{z}_{i,j} = \mathsf{Inv}_{\mathsf{ck}'^*}(\widetilde{\mathsf{com}}^*_{i,j}, \mathsf{td}')$ with $\mathsf{td}' \leftarrow \mathsf{Inv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Inv}}, r)$, $r = \mathsf{QPRF}_{\mathsf{k}_4}(\mu^*,$ $\mathrm{Ł}^*)$.

We know that if the above equality holds, then we have

$$\overline{\mathbf{A}}\boldsymbol{z}^*_{i,J_i^*} - \sum_{u\in[n^*-1]} c^*_{i,J_i^*}\boldsymbol{t}_u - c^*_{i,J_i^*}\boldsymbol{t}_n = \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - \sum_{u\in[n^*-1]} c^*_{i,j}\boldsymbol{t}_u - c^*_{i,j}\boldsymbol{t}_n. \quad (12)$$

Furthermore, for any $u \in [n^*-1]$, we can extract $\mathsf{sk}_u = \boldsymbol{s}_u$, i.e., run $\Pi.\mathsf{Ext}(\mathsf{CRS}, \mathsf{tk}, \overline{\mathbf{A}}, \boldsymbol{t}_u, \pi_u)$ to get $\boldsymbol{s}_u$, such that $\overline{\mathbf{A}} \cdot \boldsymbol{s}_u = \boldsymbol{t}_u$. In this case, (12) can be rewritten as

$$\overline{\mathbf{A}}\boldsymbol{z}^*_{i,J_i^*} - \sum_{u\in[n^*-1]} (c^*_{i,J_i^*}\overline{\mathbf{A}}\boldsymbol{s}_u) - c^*_{i,J_i^*}\boldsymbol{t}_n = \overline{\mathbf{A}}\boldsymbol{z}_{i,j} - \sum_{u\in[n^*-1]} (c^*_{i,j}\overline{\mathbf{A}}\boldsymbol{s}_u) - c^*_{i,j}\boldsymbol{t}_n. \tag{13}$$

Furthermore, from (13), we have

$$\overline{\mathbf{A}}\left(\boldsymbol{z}^*_{i,J_i^*} - \sum_{u\in[n^*-1]} c^*_{i,J_i^*}\boldsymbol{s}_u\right) - c^*_{i,J_i^*}\boldsymbol{t}_n = \overline{\mathbf{A}}\left(\boldsymbol{z}_{i,j} - \sum_{u\in[n^*-1]} c^*_{i,j}\boldsymbol{s}_u\right) - c^*_{i,j}\boldsymbol{t}_n. \tag{14}$$

From (14), we get

$$(\mathbf{A}|\mathbf{I}|\boldsymbol{t}_n)\begin{pmatrix}\boldsymbol{z}^*_{i,J_i^*} - \boldsymbol{z}_{i,j} + \sum_{u\in[n^*-1]}(c^*_{i,j^*} - c^*_{i,J_i^*})\boldsymbol{s}_u \\ c^*_{i,J_i^*} - c^*_{i,j}\end{pmatrix} = 0.$$

Recalling that $(\mathbf{A}'|\mathbf{I}) = (\mathbf{A}|\boldsymbol{t}_n|\mathbf{I})$ is an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ problem, we have found a valid solution if $\beta = \sqrt{(2B_n)^2 + 4\kappa + \eta^2(4\kappa\cdot(\ell+k))}$, since $||\boldsymbol{z}_i^* - \boldsymbol{z}_{i,j}|| \leq 2B_n$, $0 < ||c^{(n)*}_{i,J_i^*} - c^{(n)*}_{i,j}|| \leq \sqrt{4\kappa}$ and $\|\boldsymbol{s}_u\| = \eta\sqrt{\ell+k}$.

Input :$\mathsf{H}_0, \mathsf{H}_3, \mathsf{H}_4, \mathsf{H}_5, \mathsf{Sig} = (\{\mathsf{com}_i\}_{i\in[t]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \{\boldsymbol{z}_i\}_{i\in[t]}, \{r_i\}_{i\in[t]}, \{r'_{i,J_i}\}_{i\in[t]}, \mu, \text{Ł})$, $\mathsf{ck} \leftarrow \mathsf{H}_3(\mu, \text{Ł})$, $\mathsf{ck}' \leftarrow \mathsf{H}_4(\mu, \text{Ł}), r \leftarrow \mathsf{QPRF}_{\mathsf{k}_4}(\mu, \text{Ł}), \mathsf{td}' \leftarrow \mathsf{Inv\text{-}TCGen}(\mathsf{cpp}_{\mathsf{Inv}}, r)$, compute $\boldsymbol{t} = \sum_{u\in \text{Ł}} \boldsymbol{t}_u$ and derive challenges $c_{i,j} \leftarrow \mathsf{H}_0(i, j, \mu, \mathsf{ck}, \mathsf{ck}', \text{Ł})$ for all $i \in [t], j \in [m]$, $J_1||...||J_t \leftarrow \mathsf{H}_5(\{\mathsf{com}_i\}_{i\in[t]}, \{c_{i,j}\}_{i\in[t],j\in[m]}, \{\widetilde{\mathsf{com}}_{i,j}\}_{i\in[t],j\in[m]}, \text{Ł})$.
For any $u \in [n^*-1]$, we can extract $\mathsf{sk}_u = \boldsymbol{s}_u$, i.e., run $\Pi.\mathsf{Ext}(\mathsf{CRS}, \mathsf{tk}, \overline{\mathbf{A}}, \boldsymbol{t}_u, \pi_u)$ to get $\boldsymbol{s}_u$.
for $i = 1$ to t do

  for $j = 1$ to m except J_i do
    for each $\boldsymbol{z}' \leftarrow \mathsf{Inv}(\widetilde{\mathsf{com}}_{i,j}, \mathsf{td}')$ do
      if $||\boldsymbol{z}'|| \leq B \wedge \mathsf{Eqv\text{-}Open}_{\mathsf{ck}}(\mathsf{com}_i, r_i, \boldsymbol{w}_i := \overline{\mathbf{A}}\boldsymbol{z}' - c_{i,j}\boldsymbol{t} = \overline{\mathbf{A}}\boldsymbol{z}' - \sum_{u\in[n^*-1]} c_{i,j}\boldsymbol{t}_u - c_{i,j}\boldsymbol{t}_n)$,
      where $n^* = |L|$.
        return $\left(\frac{\boldsymbol{z}_i - \boldsymbol{z}' + \sum_{u\in[n^*-1]}(c_{i,j} - c_{i,J_i})\boldsymbol{s}_u}{c_{i,J_i} - c_{i,j}}\right)$.

Fig. 20. Extractor for QMS_2