Two-round n-out-of-n and Multi-Signatures and Trapdoor Commitment from Lattices[⋆]

Ivan Damgård[1], Claudio Orlandi[1], Akira Takahashi[1], and Mehdi Tibouchi[2]

[1] Department of Computer Science and DIGIT, Aarhus University, Denmark
{ivan,orlandi,takahashi}@cs.au.dk
[2] NTT Corporation, Japan
mehdi.tibouchi.br@hco.ntt.co.jp

September 14, 2020

Abstract. Although they have been studied for a long time, distributed signature protocols have garnered renewed interest in recent years in view of novel applications to topics like blockchains. Most recent works have focused on distributed versions of ECDSA and over variants of Schnorr signatures, however, and in particular, little attention has been given to constructions based on post-quantum secure assumptions like the hardness of lattice problems. A few lattice-based threshold signature and multi-signature schemes have been proposed in the literature, but they either rely on hash-and-sign lattice signatures (which tend to be comparatively inefficient), use expensive generic transformations, or contain subtle issues in their security proofs.

In this paper, we construct several lattice-based distributed signing protocols with low round complexity following the Fiat–Shamir with Aborts paradigm of Lyubashevsky (Asiacrypt 2009). Our protocols can be seen as distributed variants of the fast Dilithium-G signature scheme. A key step to achieve security (overlooked in some earlier papers) is to prevent the leakage that can occur when parties abort after their first message—which can inevitably happen in the Fiat–Shamir with Aborts setting. We manage to do so using lattice-based homomorphic commitments as constructed by Baum et al. (SCN 2018).

We first propose a three-round *n-out-of-n signature* from Module-LWE with tight security (using ideas from lossy identification schemes). Then, we further reduce the complexity to two rounds, at the cost of relying on Module-SIS as an additional assumption, losing tightness due to the forking lemma, and requiring somewhat more expensive *trapdoor commitments*. The construction of suitable trapdoor commitments from lattices is a side contribution of this paper. Finally, we also obtain a two-round *multi-signature* scheme as a variant of our two-round n-out-of-n protocol.

Keywords: n-out-of-n distributed signatures, multi-signatures, lattice-based cryptography, Fiat–Shamir with aborts, trapdoor commitments.

⋆ This research was supported by: the Concordium Blockchain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC); the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC).

Table of Contents

1 Introduction

In recent years, distributed signing protocols have been actively researched, motivated by many new applications for instance in the blockchain domain. One of the main motivations to construct a distributed signature is reducing the risk of compromising the secret key, which could occur in various ways, for instance as a result of attacks on cryptographic devices. In this paper, we study two similar classes of distributed signing protocols that can be constructed from from standard lattice-based computational hardness assumptions, namely n-out-of-n distributed signature schemes and multi-signature schemes.

n-out-of-n signature. An n-out-of-n signature is a special case of general t-out-of-n threshold signature. At a high level, the parties involved in n-out-of-n signature first invoke a key generation protocol in a way that each individual party P_j learns a share sk_j of the single signing key sk, but sk is unknown to anyone, and then they interact with each other to sign the message of interest. The required security property can be informally stated as follows: If all n parties agree to sign the message then they always produce a single signature that can be verified with a single public key pk; if at most $n-1$ parties are corrupted, it is not possible for them to generate a valid signature. Several recent works have studied threshold versions of ECDSA, arguably the most widely deployed signature scheme, both in the 2-of-2 variant [MR01, Lin17, DKLs18, CCL+19] and in the more general t-out-of-n case [GGN16, GG18, LN18, DKLs19, DKO+19, CCL+20, CMP20, GKSS20, DJN+20, GG20].

However it is well known that ECDSA does not withstand quantum attacks since it is based on discrete log, and it is therefore important to study post-quantum alternatives which support threshold signing. Despite this, very few works have considered n-out-of-n (or t-out-of-n) lattice-based signatures. Bendlin et al. [BKP13] proposed a threshold protocol to generate Gentry–Peikert–Vaikuntanathan signature [GPV08]; Boneh et al. [BGG+18] developed a universal thresholdizer that turns any signature scheme into a non-interactive threshold one, at the cost of using relatively heavy threshold fully homomorphic encryption.

However, neither of them particularly investigated signatures following the *Fiat–Shamir with Aborts (FSwA)* paradigm due to Lyubashevsky [Lyu09, Lyu12], which was specifically designed for lattice-based signatures and is nowadays the most efficient and popular approach to constructing such schemes. Recall that in standard Fiat-Shamir signatures, the scheme is based on an underlying 3-move Σ-protocol where transcripts are of form (w, c, z) and where c is a random challenge. This interactive protocol is then turned into a non-interactive signature scheme by choosing the challenge as the hash of the first message w and the message to be signed. The FSwA paradigm follows the same approach, with the important difference that the signer (prover) is allowed to abort the protocol after seeing the challenge. Only the non-aborting instances are used for signatures, and it turns out that this allows the signer to reduce the size of the randomness used and hence reduces signature size. This comes at the cost of marginally larger signing time because some (usually quite small) fraction of the protocol executions are lost due to aborts. An important issue with the FSwA approach is that the underlying Σ-protocol is only zero-knowledge in cases where the protocol does not abort. As a result, the signer should not reveal any of the aborted executions since otherwise the scheme cannot be proved secure. In a single-user scheme, this issue is not serious since there is no reason why the signer would reveal aborted executions. But in distributed schemes this can be a problem, as we shall see later.

Examples of single-user schemes based on FSwA include Dilithium [LDK+19] and qTESLA [BAA+19], both of which are round-2 candidates of the NIST Post-Quantum Cryptography Standardization process. Cozzo and Smart [CS19] recently estimated the concrete communication and computational costs required to build a distributed version of these schemes by computing Dilithium and qTESLA with generic multi-party computation. They pointed out inherent performance issue with MPC due to the mixture of both linear and non-linear operations within the FSwA framework.

Given all this, an obvious open question is to construct secure and efficient n-out-of-n protocols specifically tailored to the FSwA, while also achieving small round complexity.

Multi-signature. A multi-signature protocol somewhat resembles n-out-of-n signature and allows a group of n parties holding a signing key $sk_1, \ldots, sk_n$ to collaboratively sign the same message to obtain a single signature. However, multi-signature protocols differ from n-out-of-n signing in the following ways: 1) there is no dedicated key generation protocol, and instead each party P_j locally generates its own key pair (pk_j, sk_j) and publish pk_j before signing (so-called the *plain public-key model* [BN06]), 2) the group of signing parties is usually not fixed, and each party can initiate the signing protocol with a dynamically chosen subset of parties associated with $L \subseteq \{pk_1, \ldots, pk_n\}$, and 3) the verification algorithm usually doesn't take a single fixed public key, and instead takes a particular set of public keys L that involved in the signing protocol. Hence, roughly speaking, multi-signatures have more flexibility than n-out-of-n

signatures in terms of the choice of co-signers, at the cost of larger joint public key size and verification time (unless more advanced feature like key aggregation [MPSW19] is supported).

There is a long line of research that starts from Schnorr's signature scheme [Sch90] and follows the standard Fiat–Shamir paradigm to build distributed signatures [SS01, NKDM03, AF04, GJKR07, KG20] and multi-signatures [MOR01, BN06, BCJ08, MWLD10, STV$^+$16, MPSW19, NRSW20]. In particular, Drijvers et al. [DEF$^+$19] recently discovered a flaw of the existing two-round Schnorr-based multi-signatures, and accordingly proposed mBCJ scheme, a provably secure variant of Bagherzhandi et al.'s scheme [BCJ08].

Unlike distributed n-out-of-n signatures, several FSwA-type multi-signatures are already present in the literature. Bansarkhani and Sturm [BS16] extended the Güneysu–Lyubashevsky–Pöppelmann (GLP) [GLP12] signature and proposed the first multi-signature following the FSwA paradigm, which was recently followed by multiple similar variants [MJ19, TLT19, TE19, FH19]. These protocols essentially rely on the ideas of Schnorr-based counterparts; for instance, [BS16] can be considered as a direct adaptation of Bellare–Neven's three-round Schnorr-like multi-signature [BN06]. However, as explained below, we find that the security proofs of all these protocols are incomplete, where the underlying problem only emerges in the Fiat–Shamir *with aborts* setting. Therefore, we are also motivated to construct a provably secure multi-signature protocol within this paradigm.

1.1 Contributions and Technical Overview

FSwA-based distributed signatures with full security proof. In this paper we construct FSwA-type n-out-of-n distributed and multi-signature protocols solely relying on the hardness of learning with errors (LWE) and short integer solution (SIS) problems. We first observe that there is an inherent issue when constructing distributed FSwA signatures. That is, previous FSwA-based multi-signatures [BS16, FH19, TLT19, MJ19, TE19] ask all parties to start doing what is essentially a single user FSwA signature, and always reveal the first "commit" message of the underlying Σ-protocol. Then all these messages are added up and the sum is hashed, together with the message to be signed, in order to obtain the challenge. This means that all executions are revealed, whether they abort or not. However, as mentioned in several previous works on FSwA-type zero knowledge proofs or signatures (e.g., [BCK$^+$14, §3.2], [BBE$^+$18, BBE$^+$19], or [Lyu19, p.26]), there is no known general way to simulate the underlying Σ-protocol in case of aborts (see discussion in Section 2.4 for more formal details). Note that the obvious fix where players commit to the initial messages and only reveal them if there is no abort will not work here: the protocols need to add the initial messages together before hashing in order to reduce signature size, only the sum is included in the signature. So with this approach the initial messages must be known in the clear before the challenge can be generated. Despite the lack of such discussion in the proofs there are no known concrete attacks against the existing schemes, and it may be that one could patch the problem by making additional non-standard assumptions, or by carefully choosing the parameter such that the additional assumptions hold unconditionally. Still, we believe it is paramount to strive to find protocols which can be proven secure relying on well-established computational hardness assumptions like LWE and SIS.

In our protocol we circumvent the issue by utilizing Baum et al.'s *additively homomorphic* commitment scheme [BDL$^+$18], which is currently the most efficient construction based on lattices and relies on the hardness of Module-LWE and Module-SIS problems. In Section 3 we present our provably secure, three-round n-out-of-n distributed signing protocol DS_3 based on Dilithium-G signature scheme [DLL$^+$18], with a simple interactive key generation phase relying on random oracle commitments.

In the signing protocol, each player P_j generates an initial Σ-protocol message as in previous schemes, but then commits to it using a homomorphic commitment com_j. He then broadcasts a hash based commitment to com_j. This is just a (randomized) hash of com_j. Once all parties have done this, com_j's are revealed in the next round and checked against the hashes (the extra round doing hashing first is necessary for technical reasons as observed by Bellare and Neven [BN06] in the context of Schnorr-based multi-signature). Once the com_j's are known, they can be added together in a meaningful way by the homomorphic property, and we then hash the sum and the message to get the challenge. Thanks to the computational hiding property, each party leaks no useful information about the initial protocol message until the rejection sampling is successful. We also provide a tight security reduction to Module-LWE for this three-round protocol using the lossy identification technique by Abdalla et al. [AFLT16]. This is made possible by the extra initial round involving the hash-based commitments.

First two-round protocols. The aforementioned FSwA-based multi-signatures required at three rounds of interaction. We show that the application of homomorphic commitment makes it possible to reduce the round complexity to two rounds, by dropping the first round involving hash-based commitments, and instead send the com_j's immediately. In Section 4 we prove security of our two-round, n-out-of-n signature protocol DS_2 assuming that the com_j's actually come from a *trapdoor commitment scheme*. This way, the simulation of the honest party does not require the knowledge of dishonest parties' commitment shares; instead, the simulator can now simply send a commitment (to some random value) and then later equivocate to an arbitrary value using the trapdoor. With a slight modification this n-out-of-n protocol can be also turned into a two-round multi-signature, which is provably secure in the plain public key model. We describe a multi-signature variant MS_2 in Appendix D. Unlike the three-round protocol, we only present non-tight security proofs relying on the forking lemma and we reduce the security to both Module-SIS and Module-LWE assumptions; since a trapdoor commitment can at most be computationally binding, we must extract from the adversary two different openings to the same commitment in order to get reduction to the binding property. We leave for future work a tight security proof for two-round protocols.

The approaches taken in our two-round protocols are highly inspired by mBCJ Schnorr-like multi-signature [DEF+19]. In particular, we observe that it is crucial to use per-message commitment keys (as in mBCJ) instead of a single fixed key for all signing attempts (as in the original BCJ [BCJ08]): indeed, the latter variant would be susceptible to a variant of Drijvers et al.'s sub-exponential concurrent attack (see Appendix F). In [DEF+19] Drijvers et al. only presented a protocol in the *key verification model*, in which each co-signer has to submit a zero-knowledge proof of knowledge of the secret key. Our protocols confirm that a similar approach securely transfers to the lattice setting even under different security models: distributed n-out-of-n signature with dedicated key generation phase, and multi-signature in the plain public key model.

Lattice-based trapdoor commitment. As an additional contribution, we turn Baum et al.'s scheme into a trapdoor commitment in Section 5, so that the two-round protocols DS_2 and MS_2 are indeed instantiable with only lattice-based assumptions. We make use of the lattice trapdoor by Micciancio and Peikert [MP12] to generate a trapdoor commitment key in the ring setting. The only modification required is that the committer now samples randomness from the discrete Gaussian distribution instead of the uniform distribution. This way, the committer holding a trapdoor of the commitment key can equivocate a commitment to an arbitrary message by sampling a small randomness vector from the Gaussian distribution. Such randomness is indeed indistinguishable from the actual randomness used in the committing algorithm. Since only a limited number of lattice-based trapdoor commitment schemes are known [GSW13, CHKP10, DM14, GVW15, LNTW19] our technique may be of independent interest.

1.2 Related Work

The FSwA paradigm was first proposed by Lyubashevsky [Lyu09, Lyu12] and many efficient signature schemes following this framework have been devised, such as GLP [GLP12], BLISS [DDLL13], Dilithium [LDK+19] and qTESLA [BAA+19]. Bansarkhani ans Sturm [BS16] extended GLP signature and proposed the first multi-signature following the FSwA paradigm. Since then several variants appeared in the literature: four-round protocol with public key aggregation [MJ19], three-round protocol with tight security proof [FH19], ID-based blind multi-signature [TLT19] and ID-based proxy multi-signature [TE19]. However, as we discuss in this paper the security proofs for all these multi-signatures are incomplete. Choi and Kim [CK16] proposed a linearly homomorphic multi-signature from lattices trapdoors. Kansal and Dutta [KD20] recently constructed a single-round multi-signature scheme relying on the hardness of SIS, although the underlying signature follows neither FSwA nor hash-and-sign paradigm. Cayrel et al. [CLRS10] and Bettaieb and Schrek [BS13] developed lattice-based threshold ring signatures. Döroz et al. [DHSS20] devised lattice-based aggregate signature schemes relying on rejection sampling. Very recently, Esgin et al. [EEE20] developed FSwA-based adaptor signatures with application to blockchains.

Our two-round protocol relies on trapdoor commitments so that the simulator for an honest party sends a commitment to some random value as the first message of Σ-protocol, and it can later equivocate to an arbitrary value depending on a possibly biased challenge. Similar tricks have appeared in the ZK literature [Dam00, BKLP15, CPS+16, COSV17b, COSV17a], to turn honest verifier ZK proof into full-fledged ZK. Moreover, recent efficient lattice-based ZK proofs [dLS18, ESS+19, YAZ+19, BLS19, ESLL19] also make use of Baum et al.'s additively homomorphic commitment. The issue of revealing the first "commit" message in the FSwA framework has been also discussed by Barthe et al. [BBE+18] in the context of masking countermeasure against side-channel attacks, and they used Baum et al.'s commitment to circumvent the issue. The homomorphic lattice-based trapdoor commitment could also be instantiated with GSW-FHE [GSW13], homomorphic trapdoor functions [GVW15], Chameleon hash [CHKP10,DM14] or mercurial commitment [LNTW19].

2 Preliminaries

2.1 Notations

For positive integers a and b such that $a < b$ we use the integer interval notation $[a, b]$ to denote $\{a, a+1, \ldots, b\}$; we use $[b]$ as shorthand for $[1, b]$. If S is a set we write $s \leftarrow_\$ S$ to indicate sampling s from the uniform distribution defined over S; if D is a probability distribution we write $s \leftarrow_\$ D$ to indicate sampling s from the distribution D; if we are explicit about the set S over which the distribution D is defined then we write $D(S)$; if $\mathcal{A}$ is an algorithm we write $s \leftarrow \mathcal{A}$ to indicate assigning an output from $\mathcal{A}$ to s.

2.2 Polynomial Rings and Discrete Gaussian Distribution

In this paper most operations work over rings $R = \mathbb{Z}[X]/(f(X))$ and $R_q = \mathbb{Z}_q[X]/(f(X))$, where q is a modulus, N is a power of two defining the degree of $f(X)$, and $f(X) = X^N + 1$ is the $2N$-th cyclotomic polynomial. Following [DLL+18], we consider *centered modular reduction* $\bmod{}^{\pm}q$: for any $v \in \mathbb{Z}_q$, $v' = v \bmod{}^{\pm}q$ is defined to be a unique integer in the range $[-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ such that $v' = v \mod q$. We define the norm of $v \in \mathbb{Z}_q$ such that $\|v\| := |v \bmod{}^{\pm}q|$. Now we define the ℓ_p-norm for a (vector of) ring element $\mathbf{v} = (\sum_{i=0}^{N-1} v_{i,1}X^i, \ldots, \sum_{i=0}^{N-1} v_{i,m}X^i)^T \in R^m$ as follows.

$$\|\mathbf{v}\|_p := \left\|(v_{0,1}, \ldots, v_{N-1,1}, \ldots, v_{0,m}, \ldots, v_{N-1,m})^T\right\|_p.$$

We rely on the following *key set* $S_\eta \subseteq R$ parameterized by $\eta \geq 0$ consisting of small polynomials.

$$S_\eta = \{x \in R : \|x\|_\infty \leq \eta\}$$

Moreover the *challenge set* $C \subseteq R$ parameterized by $\kappa \geq 0$ consists of small and sparse polynomials, which will be used as the image of random oracle H_0.

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

The discrete Gaussian distribution over R^m is defined as follows.

Definition 1 (Discrete Gaussian Distribution over R^m). *For* $\mathbf{x} \in R^m$*, let* $\rho_{\mathbf{v},s}(\mathbf{x}) = \exp\left(-\pi \|\mathbf{x} - \mathbf{v}\|_2^2 / s^2\right)$ *be a* Gaussian function *of parameters* $\mathbf{v} \in R^m$ *and* $s \in \mathbb{R}$*. The* discrete Gaussian distribution $D^m_{\mathbf{v},s}$ *centered at* $\mathbf{v}$ *is*

$$D^m_{\mathbf{v},s}(\mathbf{x}) := \rho_{\mathbf{v},s}(\mathbf{x})/\rho_{\mathbf{v},s}(R^m)$$

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

In what follows we omit the subscript $\mathbf{v}$ if $\mathbf{v} = \mathbf{0}$ and write D^m_s as a shorthand. When s exceeds the so-called *smoothing parameter* $\eta(R^m) \leq \omega(\sqrt{\log(mN)})$ of the ambient space, then the discrete Gaussians $D_{R^m - \mathbf{v},s} = D^m_{\mathbf{v},s} - \mathbf{v}$ supported on all cosets of R^m are statistically close, and hence D^m_s behaves qualitatively like a continuous Gaussian of standard deviation $\sigma = s/\sqrt{2\pi}$. The condition on s will be satisfied for all the discrete Gaussians in this paper, and hence $\sigma = s/\sqrt{2\pi}$ will be called the standard deviation (even though it technically holds only up to negligible error). For the same reason, we will always be in a setting where the following fact [MP13, Theorem 3.3] [ESLL19, Lemma 9] holds.

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

2.3 Lattice Problems

Below we define two standard lattice problems over rings: module short integer solution (MSIS) and learning with errors (MLWE). We also call them MSIS/MLWE *assumption* if for any probabilistic polynomial-time adversaries the probability that they can solve a given problem is negligible.

Definition 2 ($\mathsf{MSIS}_{q,k,\ell,\beta}$ problem). *Given a random matrix $\mathbf{A} \leftarrow_\$ R_q^{k \times \ell}$ find a vector $\mathbf{x} \in R_q^{\ell+k} \setminus \{0\}$ such that $[\mathbf{A}|\mathbf{I}] \cdot \mathbf{x} = \mathbf{0}$ and $\|\mathbf{x}\|_2 \leq \beta$.*

Definition 3 ($\mathsf{MLWE}_{q,k,\ell,\eta}$ problem). *Given a pair $(\mathbf{A}, \mathbf{t}) \in R_q^{k \times \ell} \times R_q^k$ decide whether it was generated uniformly at random from $R_q^{k \times \ell} \times R_q^k$, or it was generated in a way that $\mathbf{A} \leftarrow_\$ R_q^{k \times \ell}, \mathbf{s} \leftarrow_\$ S_\eta^{\ell+k}$ and $\mathbf{t} := [\mathbf{A}|\mathbf{I}] \cdot \mathbf{s}$.*

2.4 Fiat–Shamir with Aborts Framework and Dilithium-G

We present a non-optimized version of Dilithium-G signature scheme in Algorithms 1 to 3, on which we base our distributed signing protocols. The random oracle is defined as $\mathsf{H}_0 : \{0,1\}^* \to C$. Due to Lemma 2 below the maximum ℓ_2-norm of the signature $\mathbf{z} \in R^{\ell+k}$ is set to $B = \gamma\sigma\sqrt{(\ell+k)N}$, where the parameter $\gamma > 1$ is chosen such that the probability $\gamma^{(\ell+k)N} e^{(\ell+k)N(1-\gamma^2)/2}$ is negligible.

Lemma 2 **([Lyu12]).** *For any $\gamma > 1$, $\Pr[\|\mathbf{z}\|_2 > \gamma\sigma\sqrt{mN} : \mathbf{z} \leftarrow_\$ D_s^m] < \gamma^{mN} e^{mN(1-\gamma^2)/2}$.*

The following lemma is crucial for the signing oracle of FSwA to be simulatable, and also to decide the standard deviation σ as well as the expected number of repetitions M. For instance, setting $\alpha = 11$ and $t = 12$ leads to $M \approx 3$. Although M is asymptotically superconstant, t increases very slowly in practice, and hence M behaves essentially like a constant for practical security parameters (in the literature, it is often taken as 12 to ensure $e^{-t^2/2} < 2^{-100}$, thereby ensuring > 100 bits of security).

Lemma 3 **(Rejection Sampling Lemma [Lyu12]).** *Let $V \subseteq R^m$ be a set of polynomials such that for all $\mathbf{v} \in V$, it holds that $\|\mathbf{v}\|_2 \le T$, and let $h : V \to \mathbb{R}$ be a probability distribution. Fix some t such that $t = \omega(\sqrt{\log(mN)})$ and $t = o(\log(mN))$. For any $\alpha > 0$ if $\sigma = \alpha T$ and $M = e^{t/\alpha + 1/(2\alpha)^2}$ then the statistical distance between two output distributions of* RS *(Algorithm 4) and* SimRS *(Algorithm 5) is at most $e^{-t^2/2}/M$, which is negligible. Moreover, the probability that* RS *outputs $(\mathbf{z}, \mathbf{v}) \neq (\perp, \perp)$ is at least $(1 - e^{-t^2/2})/M$, which is noticeable.*

We now present a supporting lemma which is required for Dilithium-G to be UF-CMA secure. This is almost a direct consequence of Lemma 3 and a similar result appears in [KLS18, Lemma 4.3] to prove the security of Dilithium signature instantiated with the uniform distribution. We stress that the simulator in Algorithm 7 can only simulate transcripts of non-abort executions in the underlying *interactive Σ*-protocol; in fact, if Trans output $\mathbf{w}$ in case of rejection as it's done in the interactive protocol then there is no known method to simulate the *joint distribution* of $(\mathbf{w}, c, \perp)$ [BCK+14, Lyu19] (without assuming some ad-hoc assumption like rejection-DCK [BBE+18]). The possibility of unconditionally simulating aborted executions is an open question at the time of writing.

Lemma 4 **(Non-abort Special Honest Verifier Zero Knowledge).** *Suppose T is defined as follows: if $\mathbf{s} \in S_\eta^{\ell+k}$ is sampled uniformly at random, it holds that $T \ge \|c\mathbf{s}\|_2$ with overwhelming probability for any $c \in C$. Fix some t such that $t = \omega(\sqrt{\log(mN)})$ and $t = o(\log(mN))$. For any $\alpha > 0$ if $\sigma = \alpha T$ and $M = e^{t/\alpha + 1/(2\alpha)^2}$, then for any $c \in C$ the output distributions of* $\mathsf{Trans}(sk, c)$ *(Algorithm 6) and* $\mathsf{SimTrans}(pk, c)$ *(Algorithm 7) are statistically indistinguishable, where $(pk, sk) = ((\bar{\mathbf{A}}, \mathbf{t}), \mathbf{s})$ is output from Algorithm 1.*

Proof. Suppose a set $V_\mathbf{s} = \{\mathbf{v} \in R^{\ell+k} : \mathbf{v} = c\mathbf{s} \text{ for } c \in C\}$ for a fixed signing key $\mathbf{s} \in S_\eta^{\ell+k}$ chosen uniformly at random. Due to the choice of T Lemma 3 ensures that the statistical distance would be negligible if both algorithms output $\mathbf{v} \in V_\mathbf{s}$ instead of c. Since for every $\mathbf{v} \in V_\mathbf{s}$ there exists a unique $c \in C$ such that $c\mathbf{s} = \mathbf{v}$ holds [Lyu12, DLL+18], replacing the output with c doesn't increase the statistical distance. Finally, outputting $\mathbf{w}$ doesn't affect the statistical distance either since the underlying identification protocol is *commitment recoverable* [KLS18] and therefore $\mathbf{w}$ can be simply reconstructed using c, $\mathbf{z}$ and pk anyway.

Algorithm 1 Key generation

```
Require: $pp = (R_q, k, \ell, \eta, B, \sigma, M)$
Output: $(sk, pk)$
1: $\mathbf{A} \leftarrow_\$ R_q^{k \times \ell}$
2: $\bar{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k \times (\ell + k)}$
3: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow_\$ S_\eta^\ell \times S_\eta^k; \mathbf{s} := \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix}$
4: $\mathbf{t} := \bar{\mathbf{A}}\mathbf{s}$
5: $sk := \mathbf{s}$
6: $pk := (\bar{\mathbf{A}}, \mathbf{t})$
7: return $(sk, pk)$
```

Algorithm 3 Signature generation

```
Require: $sk, \mu, pp = (R_q, k, \ell, \eta, B, \sigma, M)$
Output: valid signature pair $(\mathbf{z}, \mathbf{c})$
1: $(\mathbf{y}_1, \mathbf{y}_2) \leftarrow_\$ D_s^\ell \times D_s^k; \mathbf{y} := \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}$
2: $\mathbf{w} := \bar{\mathbf{A}}\mathbf{y}$
3: $c \leftarrow \mathsf{H}_0(\mathbf{w}, \mu, \mathbf{t})$
4: $\mathbf{z} := c\mathbf{s} + \mathbf{y}$
5: With prob. $\min\left(1, D_s^{\ell+k}(\mathbf{z})/(M \cdot D_{c\mathbf{s},\sigma}^{\ell+k}(\mathbf{z}))\right)$:
6:     return $(\mathbf{z}, c)$
7: Restart otherwise
```

Algorithm 2 Signature verification

```
Require: $pk, (\mathbf{z}, c), \mu, pp$
1: If $\|\mathbf{z}\|_2 \le B$ and $c = \mathsf{H}_0(\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}, \mu, \mathbf{t})$:
2:     return 1
3: Otherwise: return 0
```

Algorithm 4 RS

```
1: $\mathbf{v} \leftarrow_\$ h$
2: $\mathbf{z} \leftarrow_\$ D_{\mathbf{v},s}^m$
3: With prob. $\min\left(1, D_s^m(\mathbf{z})/(M \cdot D_{\mathbf{v},s}^m(\mathbf{z}))\right)$:
4:     return $(\mathbf{z}, \mathbf{v})$
5: Otherwise:
6:     return $(\perp, \perp)$
```

Algorithm 5 SimRS

```
1: $\mathbf{v} \leftarrow_\$ h$
2: $\mathbf{z} \leftarrow_\$ D_s^m$
3: With prob. $1/M$:
4:     return $(\mathbf{z}, \mathbf{v})$
5: Otherwise:
6:     return $(\perp, \perp)$
```

Algorithm 6 $\mathsf{Trans}(sk, c)$

```
1: $\mathbf{y} \leftarrow_\$ D_s^{\ell+k}$
2: $\mathbf{w} := \bar{\mathbf{A}}\mathbf{y}$
3: $\mathbf{z} := c\mathbf{s} + \mathbf{y}$
4: With prob. $\min\left(1, D_s^{\ell+k}(\mathbf{z})/(M \cdot D_{c\mathbf{s},s}^{\ell+k}(\mathbf{z}))\right)$:
5:     return $(\mathbf{w}, c, \mathbf{z})$
6: Otherwise:
7:     return $(\perp, c, \perp)$
```

Algorithm 7 $\mathsf{SimTrans}(pk, c)$

```
1: $\mathbf{z} \leftarrow_\$ D_s^{\ell+k}$
2: $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$
3: With prob. $1/M$:
4:     return $(\mathbf{w}, c, \mathbf{z})$
5: Otherwise:
6:     return $(\perp, c, \perp)$
```

2.5 Trapdoor Homomorphic Commitment Scheme with Uniform Key

Below we formally define a commitment scheme with three additional properties: uniform key, additive homomorphism, and trapdoor. The lattice-based commitments described in Section 5 indeed satisfy these. The uniform property is required since our protocols rely on commitment key derivation via random oracles mapping to a key space S_{ck}, and thus its output distribution should look like the one from CGen. Many other standard schemes like Pedersen commitment [Ped92] trivially satisfy this property. The additive homomorphism is also needed to preserve the algebraic structure of the first "commit" message of FSwA. Finally, the trapdoor commitment is crucial to achieve two-round protocols.

Definition 4 (Commitment Scheme). *A commitment scheme* COM *consists of the following algorithms.*

- $\mathsf{CSetup}(1^\lambda) \to cpp$*: The setup algorithm outputs a public parameter cpp defining sets* $S_{ck}, S_{msg}, S_r, S_{com}$ *and the distribution* $D(S_r)$ *from which the randomness is sampled.*
- $\mathsf{CGen}(cpp) \to ck$*: The key generation algorithm that samples a commitment key from* S_{ck}*.*
- $\mathsf{Commit}_{ck}(msg; r) \to com$*: The committing algorithm that takes a message* $msg \in S_{msg}$ *and randomness* $r \in S_r$ *as input and outputs* $com \in S_{com}$*. We simply write* $\mathsf{Commit}_{ck}(msg)$ *when it uses r sampled from* $D(S_r)$*.*
- $\mathsf{Open}_{ck}(com, r, msg) \to b$*: The opening algorithm outputs* $b = 1$ *if the input tuple is valid, and* $b = 0$ *otherwise.*

A commitment is said to be secure if the following properties hold.

- Correctness *It is correct if for any $msg \in S_{msg}$*

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

- Hiding *It is unconditionally (resp. computationally) hiding if the following probability is negligible for any probabilistic adversary (resp. probabilistic polynomial-time adversary) $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.*

$$\epsilon_{hide} \coloneqq \left| \Pr\left[b = b' : \begin{array}{l} cpp \leftarrow \mathsf{CSetup}(1^\lambda);\, ck \leftarrow \mathsf{CGen}(cpp) \\ (msg_0, msg_1) \leftarrow \mathcal{A}_1(ck, cpp) \\ b \leftarrow_\$ \{0,1\};\, com \leftarrow \mathsf{Commit}_{ck}(msg_b) \\ b' \leftarrow \mathcal{A}_2(com) \end{array}\right] - \frac{1}{2} \right|$$

- Binding *It is unconditionally (resp. computationally) binding if the following probability is negligible for any probabilistic adversary (resp. probabilistic polynomial-time adversary) $\mathcal{A}$.*

$$\epsilon_{bind} \coloneqq \Pr\left[\begin{array}{l} msg \neq msg' \\ \mathsf{Open}_{ck}(com, r, msg) \to 1 \\ \mathsf{Open}_{ck}(com, r', msg') \to 1 \end{array} : \begin{array}{l} cpp \leftarrow \mathsf{CSetup}(1^\lambda) \\ ck \leftarrow \mathsf{CGen}(cpp) \\ (com, msg, r, msg', r') \leftarrow \mathcal{A}(ck) \end{array}\right]$$

  In particular, unconditionally binding implies that the following probability is also negligible, 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.

$$\epsilon_{ubind} \coloneqq \Pr\left[\begin{array}{l} \exists (com, r, msg, r', msg') : \\ msg \neq msg' \\ \mathsf{Open}_{ck}(com, r, msg) \to 1 \\ \mathsf{Open}_{ck}(com, r', msg') \to 1 \end{array} : \begin{array}{l} cpp \leftarrow \mathsf{CSetup}(1^\lambda) \\ ck \leftarrow \mathsf{CGen}(cpp) \end{array}\right]$$

Definition 5 (Uniform Key). *A commitment key is said to be* uniform *if the output of* $\mathsf{CGen}(cpp)$ *follows the uniform distribution over the key space S_{ck}.*

Definition 6 (Additive Homomorphism). *A commitment is said to be* additively homomorphic *if for any $msg, msg' \in S_{msg}$*

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

Definition 7 (Trapdoor Commitment Scheme). *A trapdoor commitment scheme* TCOM *consists of the following algorithms in addition to* $(\mathsf{CSetup}, \mathsf{CGen}, \mathsf{Commit}, \mathsf{Open})$ *in Definition 4.*

- $\mathsf{TCGen}(cpp) \to (ck, td)$*: The trapdoor key generation algorithm that outputs $ck \in S_{ck}$ and the trapdoor $td \in S_{td}$.*
- $\mathsf{TCommit}_{ck}(td) \to (com, st)$*: The trapdoor committing algorithm that outputs a commitment $com \in S_{com}$ and state st.*
- $\mathsf{Eqv}_{ck}(td, st, com, msg) \to r$*: The equivocation algorithm that outputs randomness $r \in S_r$.*

A trapdoor commitment is said to be ϵ_{td}-secure if it is unconditionally hiding, computationally binding, and the following probability is negligible for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

$$\epsilon_{td} \coloneqq \left| \Pr\left[b = 0 : \begin{array}{ll} cpp & \leftarrow \mathsf{CSetup}(1^\lambda) \\ ck & \leftarrow \mathsf{CGen}(cpp) \\ msg & \leftarrow \mathcal{A}_1(ck) \\ r & \leftarrow_\$ D(S_r) \\ com & \leftarrow \mathsf{Commit}_{ck}(msg; r) \\ b & \leftarrow \mathcal{A}_2(com, r) \end{array}\right] - \Pr\left[b = 0 : \begin{array}{ll} cpp & \leftarrow \mathsf{CSetup}(1^\lambda) \\ (ck, td) & \leftarrow \mathsf{TCGen}(cpp) \\ msg & \leftarrow \mathcal{A}_1(ck) \\ (com, st) & \leftarrow \mathsf{TCommit}_{ck}(td) \\ r & \leftarrow \mathsf{Eqv}_{ck}(td, st, com, msg) \\ b & \leftarrow \mathcal{A}_2(com, r) \end{array}\right] \right|$$

```
$\mathsf{Exp}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}}(\mathcal{A})$
1 : $\mathcal{M} \leftarrow \varnothing; \mathit{flag} \leftarrow \mathsf{false}$
2 : $pp \leftarrow \mathsf{Setup}(1^\lambda)$
3 : $(\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{OGen},\mathsf{OSign}(\cdot)}(pp)$
4 : $b \leftarrow \mathsf{Ver}(\mu^*, \sigma^*, pk)$
5 : return $(b = 1) \wedge \mu^* \notin \mathcal{M}$

$\mathsf{OGen}$
1 : If $\mathit{flag}$ is set: return $\bot$
2 : Interact with $\mathcal{A}$ using the instructions of
      $(sk_n, pk, L) \leftarrow \mathsf{Gen}(n, n, pp)$
3 : Set $\mathit{flag} \leftarrow \mathsf{true}$

$\mathsf{OSign}(\mu)$
1 : If $\mathit{flag}$ is not set: return $\bot$
2 : $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
3 : Interact with $\mathcal{A}$ using the instructions of
      $\sigma \leftarrow \mathsf{Sign}(sk_n, \mu, L)$
```

Fig. 1. DS-UF-CMA experiment for distributed signature protocol, where L denotes a set of co-signers' public keys generated during the key generation protocols and $\mathcal{M}$ denotes a set of all messages queried during the signing protocol. We fix wlog the index of challenger to n. The oracles OGen and OSign are stateful oracles that run the instructions of Gen and Sign, respectively. The protocol Gen has to be completed before OSign is invoked, and otherwise OSign ignores the incoming queries.

```
$\mathsf{Exp}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}}(\mathcal{A})$
1 : $\mathcal{M} \leftarrow \varnothing$
2 : $pp \leftarrow \mathsf{Setup}(1^\lambda)$
3 : $(sk_n, pk_n) \leftarrow \mathsf{Gen}(pp)$
4 : $(\mu^*, \sigma^*, L^*) \leftarrow \mathcal{A}^{\mathsf{OSign}(\cdot,\cdot)}(pk_n, pp)$
5 : $b \leftarrow \mathsf{Ver}(\mu^*, \sigma^*, L^*)$
6 : return $(b = 1) \wedge pk_n \in L^* \wedge (\mu^*, L^*) \notin \mathcal{M}$

$\mathsf{OSign}(\mu, L)$
1 : If $pk_n \notin L$ : return $\bot$
2 : $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\mu, L)\}$
3 : Interact with $\mathcal{A}$ using the instructions of
      $\sigma \leftarrow \mathsf{Sign}(sk_n, \mu, L)$
```

Fig. 2. MS-UF-CMA experiment for multi-signature protocol, where L denotes a set of co-signers' public keys chosen by the adversary and $\mathcal{M}$ denotes a set of all messages and sets queried during the signing protocol. We assume that the cardinality of L is at most n and wlog we fix the index of challenger to n. Note that the above experiment always requires that a set L contains challenger's public key, as otherwise the adversary can trivially forge a signature.

2.6 Security Notions for n-out-of-n Distributed and Multi-Signature Protocols

We first define the n-out-of-n distributed signature protocol and its security notion. The game-based security notion below is based on the one presented by Lindell [Lin17] for two-party signing protocol. Our definition can be regarded as its generalization to n-party setting. Throughout the paper we fix wlog the index of honest party and challenger to n.

Definition 8 (Distributed Signature Protocol). *A distributed signature protocol* $\mathsf{DS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ *consists of four algorithms.*

– $\mathsf{Setup}(1^\lambda) \to pp$: *The set up algorithm that outputs a public parameter pp on a security parameter λ as input.*

– $\mathsf{Gen}(j, n, pp) \to (sk_j, pk, L)$: *The interactive key generation algorithm that is run by all the co-signers. Each party runs the protocol on an index of party $j \in [n]$ and common inputs: the total number of parties n and a public parameter pp. At the end of the protocol each party obtains a secret key share sk_j, public key pk and a set of co-signer's public key shares $L = \{pk_1, \ldots, pk_n\}$.*

– $\mathsf{Sign}(sk_j, \mu, L) \to \sigma$: *The interactive signing algorithm that is run by all the co-signers. Each party runs the protocol on its signing key share sk_j and common inputs: message μ, and a set of public key shares L. At the end of the protocol each party obtains a signature σ as output.*

– $\mathsf{Vf}(\sigma, \mu, pk) \to b$: *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 9 (DS-UF-CMA Security). *A distributed signature protocol* DS *is said to be* DS-UF-CMA *(distributed signature unforgeability against chosen message attacks) secure, if for any probabilistic poly-*

nomial time adversary $\mathcal{A}$, its advantage

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

is negligible in λ, where $\mathsf{Exp}_{\mathsf{DS}}^{\mathsf{DS\text{-}UF\text{-}CMA}}(\mathcal{A})$ is described in Fig. 1.

Next we define the standard security notion of multi-signature protocol in the plain public-key model. The following definitions are adapted from [BN06]. The main difference from the distributed signature is, that there is no interactive key generation protocol anymore and the adversary is not required to fix its key pair at the beginning of the game. Accordingly, the adversary can dynamically choose a set of public keys involving the challenger's key, and query the signing oracle to receive signatures. On the other hand, assuming that key aggregation is not always supported the verification algorithm takes a set of public key, instead of the single combined public key as in the prior case.

Definition 10 (Multi-signature Protocol). *A multisignature protocol $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ consists of four algorithms.*

- $\mathsf{Setup}(1^\lambda) \to pp$*: The set up algorithm that outputs a public parameter pp on a security parameter λ as input.*
- $\mathsf{Gen}(pp) \to (sk, pk)$*: The key generation algorithm that outputs a key pair on a public parameter pp as input.*
- $\mathsf{Sign}(sk, \mu, L) \to \sigma$*: The interactive signing algorithm that is run by all the cosigners. Each party runs the protocol on its signing key sk and common inputs: message μ and a set of public keys L. At the end of the protocol each party obtains a signature σ as output.*
- $\mathsf{Vf}(\sigma, \mu, L) \to b$*: The verification algorithm that takes a signature, message, and a set of public keys and outputs $b = 1$ if the signature is valid and otherwise $b = 0$.*

Definition 11 ($\mathsf{MS\text{-}UF\text{-}CMA}$ Security). *A multisignature protocol MS is said to be $\mathsf{MS\text{-}UF\text{-}CMA}$ (multisignature unforgeability against chosen message attacks) secure, if for any probabilistic polynomial time adversary $\mathcal{A}$, its advantage*

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

is negligible in λ, where $\mathsf{Exp}_{\mathsf{MS}}^{\mathsf{MS\text{-}UF\text{-}CMA}}(\mathcal{A})$ is described in Fig. 2.

3 Three-round Distributed Signature Protocol with Tight Security Proof

3.1 Protocol Specification and Overview

In this section, we give a detailed description of our three-round, n-out-of-n distributed signature protocol $\mathsf{DS}_3 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$, formally specified in Figs. 3 to 5. The protocol is built on top of additively homomorphic commitment scheme $\mathsf{COM} = (\mathsf{CSetup}, \mathsf{CGen}, \mathsf{Commit}, \mathsf{Open})$ with uniform key (see Section 2 for the formal definition), and we will describe concrete instances of COM later in Section 5. High-level ideas for each step are as follows.

Parameter Setup We assume that a trusted party invokes $\mathsf{DS}_3.\mathsf{Setup}(1^\lambda)$ that outputs a set of public parameters described in Table 1 as well as the parameter for commitment scheme *cpp* (which is obtained by internally invoking $\mathsf{COM}.\mathsf{CSetup}(1^\lambda)$). Most parameters commonly appear in the literature about the Fiat–Shamir with aborts paradigm (e.g. [DLL+18, Lyu12]) and we therefore omit the details here. The bit length l_1, l_2 and l_4 should be sufficiently long for the random oracle commitments to be secure. The only additional parameters are B_n and M_n, which we describe below.

Key generation. Upon receiving public parameters, all participants first interactively generate a random matrix $\mathbf{A} \in R_q^{k\times\ell}$, a part of Dilithium-G public key. This can be securely done with simple random oracle commitments[3]; as long as there is at least one honest party sampling a matrix share correctly, the

[3] We remark that the "commitments" generated by H_1 and H_2 in Fig. 3 are not randomized, and therefore they are not hiding. In our protocol, however, all committed values have high min-entropy and this is indeed sufficient for the security proof to hold. Alternatively, one could cheaply turn them into full-fledged secure and extractable commitments by additionally hashing random strings that are to be sent out during the opening phase [Pas03].

Protocol $\mathsf{DS}_3.\mathsf{Gen}$

The protocol is parameterized by public parameters described in Table 1 and relies on the random oracles $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ and $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{l_2}$.

Matrix Generation

1. Upon receiving $(\textsc{mgen}, pp, 0)$, sample a random matrix share $\mathbf{A}_n \leftarrow_\$ R_q^{k\times\ell}$ and generate a random oracle commitment $g_n \leftarrow \mathsf{H}_1(\mathbf{A}_n, n)$. Send out $(\textsc{exchange}, g_n, 0, n)$.
2. Upon receiving $(\textsc{exchange}, g_j, 0, j)$ for all $j \in [n-1]$ send out $(\textsc{open}, \mathbf{A}_n, 0, n)$.
3. Upon receiving $(\textsc{open}, \mathbf{A}_j, 0, j)$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_1(\mathbf{A}_j, j) \neq g_j$ for some j then send out $\textsc{abort}$.
   b. Otherwise set public random matrix $\bar{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$, where $\mathbf{A} := \sum_{j\in[n]} \mathbf{A}_j$. Now P_n will ignore the future invocation of $\textsc{mgen}$.

Key Pair Generation

1. Upon receiving $(\textsc{keygen}, pp, 1)$, compute a secret and public key shares $\mathbf{s}_n \leftarrow_\$ S_\eta^{\ell+k}$ and $\mathbf{t}_n := \bar{\mathbf{A}}\mathbf{s}_n$, respectively, and generate a random oracle commitment $g'_n \leftarrow \mathsf{H}_2(\mathbf{t}_n, n)$. Send out $(\textsc{exchange}, g'_n, 1, n)$.
2. Upon receiving $(\textsc{exchange}, g'_j, 1, j)$ for all $j \in [n-1]$ send out $(\textsc{open}, \mathbf{t}_n, 1, n)$.
3. Upon receiving $(\textsc{open}, \mathbf{t}_j, 1, j)$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_2(\mathbf{t}_j, j) \neq g'_j$ for some j then send out $\textsc{abort}$.
   b. Otherwise set a combined public key $\mathbf{t} := \sum_{j\in[n]} \mathbf{t}_j$ and its shares $L := \{\mathbf{t}_1, \ldots, \mathbf{t}_n\}$. Now P_n will ignore the future invocation of $\textsc{keygen}$.

Fig. 3. Distributed key generation protocol

resulting combined matrix is guaranteed to follow the uniform distribution. For the exact same reason, the exchange of public key shares are done with random oracle. This way, we can prevent the adversary from choosing some malicious public key share depending on the honest party's share (the so-called *rogue key attack* [MOR01]). Furthermore, the party's index j is concatenated with the values to be hashed for the sake of "domain separation" [BR93, BDG20]. This way, we prevent rushing adversaries from simply sending back the hash coming from the honest party and claiming that they know the preimage after seeing the honest party's opening.

Signature generation. The first goal of the distributed signing protocol is to exchange the first "commit" messages of Σ-protocol, from which all parties derive a single joint challenge $c \in C$ via a random oracle. The most crucial step is 1.c.; as we discussed earlier no participants are allowed to reveal $\mathbf{w}_j$ until the rejection sampling phase, and instead they send its commitment com_j, which is to be opened only if the signature share $\mathbf{z}_j$ passes the rejection sampling. Moreover, following Bellare and Neven [BN06] (or its GLP-based variant [BS16]) we need an extra round where parties exchange hashes of com_j and later check that everyone knows the correct preimage. The intuition behind this seemingly redundant step is analogous to the rogue key attack; without this step the adversary might be able to adaptively choose a malicious $\widetilde{com}$ after seeing the honest party's share. However, this extra round can be indeed dropped by instantiating the protocol with a trapdoor commitment scheme (see Section 4). We remark that generating a per-message commitment key as in 1.a. is not necessary for the three-round protocol and one could alternatively use a single fixed $ck \leftarrow \mathsf{CGen}(cpp)$ generated by the trusted party. However, this step becomes crucial for the two-round protocols to be secure.

One efficiency issue is, that the protocol has to be restarted until *all* parties pass the rejection sampling step. All previous FSwA-based multi-signatures also had the same issues, but we can mitigate this to some extent by adapting a base scheme with Gaussian distribution, instead of uniform. This is the reason why we chose to instantiate the protocol with Dilithium-"G" instead of the one submitted to NIST competition [LDK+19].

Verification and correctness. Thanks to the linearity of underlying scheme and homomorphism of the commitment, the verifier only needs to validate the sum of signature shares, commitments and randomness. Here the Euclidean-norm bound B_n is set according to Lemma 1; if all parties honestly follow the protocol then the sum of n Gaussian shares is only $\sqrt{n}$ times larger (while if we employed the

Protocol $\mathsf{DS}_3.\mathsf{Sign}$

The protocol is parameterized by public parameters described in Table 1 and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{ck}$ and $\mathsf{H}_4 : \{0,1\}^* \to \{0,1\}^{l_4}$. The protocol assumes that $\mathsf{DS}_3.\mathsf{Gen}$ in Fig. 3 has been previously invoked and works on $sk_n = \mathbf{s}_n, pk = (\bar{\mathbf{A}}, \mathbf{t})$ and $L = \{pk_1, \ldots, pk_n\}$ as input.

Signature Generation: If MGEN and KEYGEN have been already called before, P_n works as follows:

1. Upon receiving $(\textsc{sign}, \mu, sid)$, compute the first message as follows.
   a. Derive a per-message commitment key $ck \leftarrow \mathsf{H}_3(\mu, \mathbf{t})$.
   b. Sample $\mathbf{y}_n \leftarrow_\$ D_s^{\ell+k}$ and compute $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{y}_n$.
   c. Compute $com_n \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_n; r_n)$ with $r_n \leftarrow_\$ D(S_r)$, and $h_n \leftarrow \mathsf{H}_4(com_n)$.
   d. Send out $(\textsc{exchange}, h_n, sid, n)$.
2. Upon receiving $(\textsc{exchange}, h_j, sid, j)$ for all $j \in [n-1]$ send out $(\textsc{open}, com_n, sid, n)$.
3. Upon receiving $(\textsc{open}, com_j, sid, j)$ for all $j \in [n-1]$ compute the signature share as follows.
   a. If $\mathsf{H}_4(com_j) \neq h_j$ for some $j \in [n-1]$ send out ABORT.
   b. Set $com := \sum_{j\in[n]} com_j$.
   c. Derive a challenge $c \leftarrow \mathsf{H}_0(com, \mu, \mathbf{t})$.
   d. Computes a signature share $\mathbf{z}_n := c\mathbf{s}_n + \mathbf{y}_n$.
   e. Run the rejection sampling on input $(c\mathbf{s}_n, \mathbf{z}_n)$, i.e., with probability

   $$\min\left(1, D_s^{\ell+k}(\mathbf{z}_n)/(M \cdot D_{c\mathbf{s}_n,s}^{\ell+k}(\mathbf{z}_n))\right)$$

   announce $(\textsc{proceed}, \mathbf{z}_n, r_n, sid, n)$; otherwise send out $(\textsc{restart}, \perp, \perp, sid, n)$.
4. Upon receiving $(\textsc{restart}, \perp, \perp, sid, j)$ from some party go to 1. Otherwise upon receiving $(\textsc{proceed}, \mathbf{z}_j, r_j, sid, j)$ for all $j \in [n-1]$ compute the combined signature as follows
   a. For each $j \in [n-1]$ reconstruct $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c\mathbf{t}_j$ and validate the signature share:

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

   If the check fails for some j then send out ABORT.
   b. Compute $\mathbf{z} := \sum_{j\in[n]} \mathbf{z}_j$ and $r := \sum_{j\in[n]} r_j$.
   c. Output $(com, \mathbf{z}, r)$ as a signature.

Fig. 4. Three-round distributed signing protocol.

Algorithm $\mathsf{DS}_3.\mathsf{Vf}$

Signature Verification Upon receiving a message μ, signature $(com, \mathbf{z}, r)$, and combined public key $pk = (\bar{\mathbf{A}}, \mathbf{t})$, generate a commitment key $ck \leftarrow \mathsf{H}_3(\mu, \mathbf{t})$, derive a challenge $c \leftarrow \mathsf{H}_0(com, \mu, \mathbf{t})$ and reconstruct $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$. Then accept if the following holds:

$$\|\mathbf{z}\|_2 \leq B_n \quad \text{and} \quad \mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1.$$

Fig. 5. Verification algorithm for distributed signature protocol.

Table 1. Parameters for our distributed signature protocols.

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 $\mathbf{A}$
γ	Parameter defining the tail-cut bound
$B = \gamma\sigma\sqrt{N(\ell+k)}$	The maximum ℓ_2-norm of signature share $\mathbf{z}_j \in R^{\ell+k}$ for $j \in [n]$
$B_n = \sqrt{n}B$	The maximum ℓ_2-norm of combined signature $\mathbf{z} \in R^{\ell+k}$
κ	The maximum ℓ_1-norm of challenge vector c
$C = \left\{c \in R : \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\right\}$	Challenge space where $\|C\| = \binom{N}{\kappa}2^\kappa$
$S_\eta = \left\{x \in R : \|x\|_\infty \leq \eta\right\}$	Set of small secrets
$T = \kappa\eta\sqrt{N(\ell+k)}$	Chosen such that Lemma 4 holds
α	Parameter defining σ and M
$\sigma = s/\sqrt{2\pi} = \alpha T$	Standard deviation of the Gaussian distribution
$t = \omega(\sqrt{\log(mN)}) \wedge t = o(\log(mN))$	Parameter defining M such that Lemma 3 holds
$M = e^{t/\alpha + 1/(2\alpha)^2}$	The expected number of restarts until a single party announces PROCEED
$M_n = M^n$	The expected number of restarts until all n parties agree on PROCEED
cpp	Parameters for commitment scheme honestly generated with CSetup
l_1, l_2, l_4	Output bit lengths of random oracles $\mathsf{H}_1, \mathsf{H}_2$ and H_4

plain Dilithium as a base scheme the signature size would grow almost linearly). Hence together with the tail-cut bound of Lemma 2 it is indeed sufficient to set $B_n = \sqrt{n}B$ for the correctness to hold with overwhelming probability. To guarantee the perfect correctness, the bound check can be also done during the signing protocol so that it simply restarts when the generated signature is too large (which of course only happens with negligible probability and shouldn't matter in practice).

Expected number of aborts. As indicated in Table 1 the probability that all participants agree on PROCEED is $1/M_n = 1/M^n$, where $1/M$ is the probability that each party asks to proceed. To make M_n reasonably small, say $M_n = 3$, we should set $\alpha \geq 11n$ [DLL+18], leading to $\sigma \geq 11nT$. This already increases the bound B of each signature share linearly compared to a non-distributed signature like Dilithium-G. In addition, we should set the bound B_n for combined signature to $\sqrt{n}B$ for the correctness to hold, and thus the SIS solution that we find in the security reduction grows by a factor of $n^{3/2}$. This is a substantial increase, but it is an improvement over a solution using rejection sampling with respect to the uniform distribution, for which the SIS solution would grow faster. We leave for future work the question of making parameters less dependent on the number of parties n, so that the protocol scales better to a large number of parties; this might require the use of stronger assumptions for security, and incur a loss in the security reduction.

3.2 Security

We give a tight security proof for DS_3 by instantiating the protocol with an unconditionally binding commitment scheme and by setting the parameters so that the underlying SIS problem becomes vacuously hard. The full security proof is given in Appendix B. Here we give a sketch of our proof. First, thanks to the computational hiding of COM, the oracle simulator can replace the commitment share com_n of honest party P_n with a commitment to some random vector in R_q^k in case it wants to abort. For non-abort executions we can essentially invoke the special HVZK simulator of Algorithm 7 to answer the oracle queries from the adversary. Hence we can indeed simulate the honest execution of P_n.

The core idea for proving the soundness essentially follows the lossy identification technique by Abdalla et al. [AFLT16]; since the public key share of the honest signer $\mathbf{t}_n$ is indistinguishable from the vector sampled from R_q^k uniformly at random due to the LWE assumption, the oracle simulator can replace $\mathbf{t}_n$ with such a vector (i.e. a *lossy key*). Moreover, thanks to the programmability and extractability of random oracle commitments in the key generation phase, the oracle simulator can even sample the resulting combined public key $\mathbf{t}$ from the uniform distribution *in advance* and set its share $\mathbf{t}_n$ a posteriori depending on the other shares. Now, the unconditional binding of COM guarantees that there cannot *exist* commitments having two openings except a negligible fraction on the random choice of ck. (See Definition 4. Also recall that we defined a uniform key in Definition 5, so the keys given by the random oracle are perfectly indistinguishable from the ones from CGen.) Finally, we argue that on the random choice of joint public key $(\mathbf{A}, \mathbf{t})$ there cannot *exist* two valid transcripts that share the first message of the underlying Σ-protocol (i.e. $\mathbf{w} \in R_q^k$) except a negligible fraction. In that case, the only way for an adversary to come up with a forgery is to luckily receive a specific challenge $c \in C$ from the random oracle H_0, which can only happen with probability $1/|C|$.

The last step of the security proof essentially follows the one for Dilithium-QROM given by Kiltz, Lyubashevsky and Schaffner [KLS18], and we impose an additional condition on the modulus q so that the polynomials of small norm are invertible in the ring R_q [LS18][4]. We also remark that Fukumitsu and Hasegawa [FH19] also attempted a tight security reduction for their Dilithium-like three-round multi-signature scheme, although the proof disregards the simulation of rejected transcripts. Since the rest of their proof seems sound, by applying an additively homomorphic commitment as we do one could patch the scheme while maintaining the reduction tightness.

Theorem 1. *Suppose the commitment scheme* COM *is unconditionally binding, computationally hiding, uniform, additively homomorphic, and the output of committing algorithm* Commit *has* ξ*-bit min-entropy. Assume the modulus* q *satisfies* $q = 5 \mod 8$*,* $2B_n < \sqrt{q/2}$ *and* $2\kappa < \sqrt{q/2}$*. For any probabilistic polynomial-time adversary* $\mathcal{A}$ *that makes a single query to* OGen*, at most* Q_s *queries to* OSign *and at most* Q_h *queries in total to the random oracles* $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4$*, the protocol* $\mathsf{DS}_3 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ *is* $\mathsf{DS\text{-}UF\text{-}CMA}$ *secure under* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumption.*

4 Two-round Distributed Signature Protocol

4.1 Overview

In this section we show how to reduce the round complexity of DS_3 from the previous section and define the two-round variant $\mathsf{DS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$. For the full description of $\mathsf{DS}_2.\mathsf{Sign}$ see Fig. 6. The Setup, Gen and Vf are identical to DS_3. The only difference with $\mathsf{DS}_3.\mathsf{Sign}$ is that now $\mathsf{DS}_2.\mathsf{Sign}$ doesn't have a preliminary round to exchange hashes of commitments (i.e. Step 2 in Fig. 4). In Appendix D we present a multi-signature variant which can be proven secure in the *plain public key model.*

For the protocol to be provably secure, two crucial properties are required. First, the protocol needs to be instantiated with a trapdoor commitment scheme $\mathsf{TCOM} = (\mathsf{CGen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{TCGen}, \mathsf{TCommit}, \mathsf{Eqv})$ (Definition 7). This way, the oracle simulator can commit to some random vector in R_q^k at the first round, and then later equivocate to a simulated first message of the Σ-protocol (i.e. $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n$) *after* the joint random challenge $c \in C$ has been derived. The idea here is analogous to Bagherzhandi et al.'s two-round Schnorr-like multi-signature [BCJ08].

Second, this time we have to be careful about the derivation of the commitment key; in fact, if some fixed key ck was used for all signing attempts, one could come up with a sub-exponential attack that outputs a valid forgery with respect to the joint public key $\mathbf{t}$. In Appendix F we sketch a variant of the concurrent attack due to Drijvers et al. [DEF+19]. The original attack was against two-round Schnorr multi-signatures including BCJ scheme [BCJ08], but due to the very similar structure of FSwA-based lattice signatures an attack would become feasible against a fixed-key variant of DS_2. This motivates us to derive a per-message commitment key during the signing protocol as in Drivers et al.'s mBCJ scheme.

4.2 Security

We give a sketch of the security proof for DS_2, assuming the hardness of both Module-LWE and Module-SIS. The full security proof is given in Appendix C. Unlike the previous section, we only give a non-tight proof relying on the forking lemma [PS96, BN06]. This is mainly because we instantiated the protocol with a trapdoor commitment, which inevitably implies that the binding is only computational. Hence to construct a reduction that breaks binding, we do have to make the adversary submit two valid openings for a single commitment, which seems to require some kind of rewinding technique. The oracle simulation follows the one for mBCJ [DEF+19]. Concretely, the oracle simulator programs H_3 so that for all sign queries it returns ck generated via $(ck, td) \leftarrow \mathsf{TCGen}(cpp)$, with which the simulator can later responds with an opening to a challenge-dependent, simulated $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n$, thanks to the equivocation algorithm; for the specific crucial query that is used to create a forgery we embed an actual commitment

[4] This condition could be actually relaxed somewhat by applying the result due to Nguyen [Ngu19]

Protocol $\mathsf{DS}_2.\mathsf{Sign}$

The protocol is parameterized by public parameters described in Table 1 and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{ck}$. The protocol assumes that $\mathsf{DS}_2.\mathsf{Gen}$ (identical to Fig. 3) has been previously invoked and works on $sk_n = \mathbf{s}_n, pk = (\bar{\mathbf{A}}, \mathbf{t})$ and $L = \{pk_1, \ldots, pk_n\}$ as input.

Signature Generation: If MGEN and KEYGEN have been already called before, P_n works as follows:

1. Upon receiving $(\textsc{sign}, \mu, sid)$, compute the first message as follows.
   a. Derive a per-message commitment key $ck \leftarrow \mathsf{H}_3(\mu, \mathbf{t})$.
   b. Sample $\mathbf{y}_n \leftarrow_{\$} D_s^{\ell+k}$ and compute $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{y}_n$.
   c. Compute $com_n \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_n; r_n)$ with $r_n \leftarrow_{\$} D(S_r)$.
   d. Send out $(\textsc{open}, com_n, sid, n)$.
2. Upon receiving $(\textsc{open}, com_j, sid, j)$ for all $j \in [n-1]$ compute the signature share as follows.
   a. Set $com := \sum_{j\in[n]} com_j$.
   b. Derive a challenge $c \leftarrow \mathsf{H}_0(com, \mu, \mathbf{t})$.
   c. Computes a signature share $\mathbf{z}_n := c\mathbf{s}_n + \mathbf{y}_n$.
   d. Run the rejection sampling on input $(c\mathbf{s}_n, \mathbf{z}_n)$, i.e., with probability
$$\min\left(1, D_s^{\ell+k}(\mathbf{z}_n)/(M \cdot D_{c\mathbf{s}_n,s}^{\ell+k}(\mathbf{z}_n))\right)$$
   announce $(\textsc{proceed}, \mathbf{z}_n, r_n, sid, n)$; otherwise send out $(\textsc{restart}, \perp, \perp, sid, n)$.
3. Upon receiving $(\textsc{restart}, \perp, \perp, sid, j)$ from some party go to 1. Otherwise upon receiving $(\textsc{proceed}, \mathbf{z}_j, r_j, sid, j)$ for all $j \in [n-1]$ compute the combined signature as follows
   a. For each $j \in [n-1]$ reconstruct $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c\mathbf{t}_j$ and validate the signature share:
$$\|\mathbf{z}_j\|_2 \le B \quad \text{and} \quad \mathsf{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1.$$
   If the check fails for some j then send out ABORT.
   b. Compute $\mathbf{z} := \sum_{j\in[n]} \mathbf{z}_j$ and $r := \sum_{j\in[n]} r_j$.
   c. Output $(com, \mathbf{z}, r)$ as a signature.

Fig. 6. Two-round distributed signing protocol.

key $ck \leftarrow \mathsf{CGen}(cpp)$, so that when the reduction obtains two openings after applying the forking lemma it can indeed break the binding property with respect to an honestly generated key ck.

The use of rewinding also motivates us to give a security reduction to SIS just as many FSwA-type schemes do. To give a reduction to SIS we first replace the joint public key $(\mathbf{A}, \mathbf{t})$ with a random matrix sampled from $R_q^{k\times\ell} \times R_q^k$ as in the proof for DS_3. This is indeed doable under the $\mathsf{MLWE}_{q,k,\ell,\eta}$ assumption. Now we can embed an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, which is denoted as $[\mathbf{A}'|\mathbf{I}]$ with $\mathbf{A}' \leftarrow_{\$} R_q^{k\times(\ell+1)}$; due to the way we simulated the joint public key, replacing it with $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ instance doesn't change the view of adversary at all, if $\mathbf{A}'$ is regarded as $\mathbf{A}' = [\mathbf{A}|\mathbf{t}]$. After applying the forking lemma, the adversary submits two forgeries with distinct challenges $c \neq c'$, with which we can indeed find a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$. Thus the formal statement of security is as follows.

Theorem 2. *Suppose the trapdoor commitment scheme* TCOM *is* ϵ_{td}*-secure, additively homomorphic and has uniform keys. For any probabilistic polynomial-time adversary* $\mathcal{A}$ *that makes a single query to* OGen, Q_s *queries to* OSign *and* Q_h *queries to the random oracle, the protocol* $\mathsf{DS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ *is* $\mathsf{DS\text{-}UF\text{-}CMA}$ *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions, where* $\beta = 2\sqrt{B_n^2 + \kappa}$.

5 Lattice-Based Commitments

In this section, we describe possible constructions for the lattice-based commitment schemes used in our protocols. The three-round protocol of Section 3 requires a statistically binding, computationally hiding homomorphic commitment scheme, whereas the two-round protocol of Section 4 needs a statistically hiding *trapdoor* homomorphic commitment scheme. We show that both types of commitments can be obtained using the techniques of Baum et al. [BDL+18].

More precisely, the first type of commitment scheme is a simple variant of the scheme of [BDL+18], in a parameter range that ensures statistical instead of just computational binding. The fact that such a parameter choice is possible is folklore, and does in fact appear in an earlier version of [BDL+18], so we do not claim any novelty in that regard.

The construction of a lattice-based trapdoor commitment scheme does not seem to appear in the literature, but we show that it is again possible by combining [BDL+18] with Micciancio–Peikert style trapdoors [MP12]. To prevent statistical learning attacks on the trapdoor sampling, however, it is important to sample the randomness in commitment according to a discrete Gaussian distribution, in contrast with Baum et al.'s original scheme.

5.1 Statistically Binding Commitment Scheme

We first describe a statistically binding commitment scheme from lattices. The scheme, described in Fig. 8, is a simple variant of the scheme from [BDL+18], that mainly differs in the choice of parameter regime: we choose parameters so as to make the underlying SIS problem vacuously hard, and hence the scheme statistically binding. Another minor change is the reliance on discrete Gaussian distributions, for somewhat more standard and compact LWE parameters.

The correctness and security properties, as well as the constraints on parameters, are obtained as follows.

- **Correctness**. By construction. We select the bound B as $\Omega(s \cdot \sqrt{m' \cdot N})$. By [MP12, Lemma 2.9], this ensures that the probability to retry in the committing algorithm is negligible.
- **Statistically binding**. Suppose that an adversary can construct a commitment $\mathbf{f}$ on two distinct messages $x \neq x'$, with the associated randomness $\mathbf{r}, \mathbf{r}'$. Since $x \neq x'$, the correctness condition ensures that $\mathbf{r}$ and $\mathbf{r}'$ are distinct and of norm $\leq B$, and satisfy $\hat{\mathbf{A}}_1 \cdot (\mathbf{r} - \mathbf{r}') \equiv 0 \pmod{q}$. This means in particular that there are non zero elements in the Euclidean ball $B_{m'}(0, 2B)$ of radius $2B$ in $R_q^{m'}$ that map to $\mathbf{0}$ in R_q^m. But this happens with negligible probability on the choice of $\hat{\mathbf{A}}_1$ when $\left|B_{m'}(0,2B)\right|/q^{mN} = 2^{-\Omega(N)}$. Now $\left|B_{m'}(0,2B)\right| = o\big((2\pi e/m'N)^{m'N/2} \cdot (2B)^{m'N}\big)$. Hence, picking for example $m' = 2m$, we get:
$$\frac{\left|B_{m'}(0,2B)\right|}{q^{mN}} \ll \left(\frac{4\pi e \cdot B^2}{mNq}\right)^{mN},$$
and the condition is satisfied for example with $q > 8\pi e B^2/mN$.
- **Computationally hiding**. The randomness $\mathbf{r}$ can be written in the form $\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{s}\end{bmatrix}^T$ where $\mathbf{r}_1 \in R_q^m$, $\mathbf{r}_2 \in R_q^k$, $\mathbf{s} \in R_q^{m'-m-k}$ are all sampled from discrete Gaussians of parameter s. The commitment elements then become:
$$\mathbf{f}_1 = \mathbf{r}_1 + \hat{\mathbf{A}}_1 \cdot \begin{bmatrix}\mathbf{r}_2 \\ \mathbf{s}\end{bmatrix} \qquad\qquad \mathbf{f}_2 = \mathbf{r}_2 + \hat{\mathbf{A}}_2 \cdot \mathbf{s},$$
and distinguishing those values from uniform are clearly instances of decision MLWE. Picking $k = m$, $m' = 2m$, $s = \Theta(\sqrt{mN})$, $B = \Theta(mN)$, $q = \Theta\big((mN)^{3/2}\big)$ yields a simple instatiation with essentially standard security parameters.

5.2 Trapdoor Commitment Scheme

We now turn to the construction of a trapdoor commitment scheme with suitable homomorphic properties for our purposes. Our proposed scheme is described in Fig. 7. It is presented as a commitment for a *single* ring element $x \in R_q$. It is straightforward to extend it to support a vector $\mathbf{x} \in R_q^k$, but the efficiency gain from doing so is limited compared to simply committing to each coefficient separately, so we omit the extension.

We briefly discuss the various correctness and security properties of the scheme, together with the constraints that the various parameters need to satisfy. In short, we need to pick the standard deviation of the coefficients of the trapdoor matrix $\mathbf{R}$ large enough to ensure that the trapdoor key is statistically close to a normal commitment key; then, the randomness $\mathbf{r}$ in commitments should have large enough standard deviation to make commitments statistically close to uniform (and in particular statistically hiding), and also be sampleable using the trapdoor. These are constraints on $\bar{s}$ and s respectively. Finally, the bound B

Protocol Lattice-based Commitment Scheme

CSetup(1^λ) takes a security parameter and outputs $cpp = (N, q, \bar{s}, s, B, \ell, w)$.

CGen(cpp) takes a commitment parameter and samples $\hat{a}_{1,1} \leftarrow\$ R_q^\times$ (a uniform invertible element of R_q) and $\hat{a}_{1,j} \leftarrow\$ R_q$ for $j = 2, \ldots, \ell + 2w$, $\hat{a}_{2,j} \leftarrow\$ R_q$ for $j = 3, \ldots, \ell + 2w$. It then outputs:

$$\hat{\mathbf{A}} = \begin{bmatrix} \hat{a}_{1,1} & \hat{a}_{1,2} & \hat{a}_{1,3} & \cdots & \hat{a}_{1,\ell+2w} \\ 0 & 1 & \hat{a}_{2,3} & \cdots & \hat{a}_{2,\ell+2w} \end{bmatrix}$$

as ck.

Commit$_{ck}(x)$ takes $x \in R_q$ and samples a discrete Gaussian vector of randomness $\mathbf{r} \leftarrow\$ D_s^{\ell+2w}$. It then outputs

$$\mathbf{f} = \hat{\mathbf{A}} \cdot \mathbf{r} + \begin{bmatrix} 0 \\ x \end{bmatrix} \in R_q^2.$$

To ensure perfect correctness, retry unless $\|\mathbf{r}\|_2 \leq B$.

Open$_{ck}(\mathbf{f}, \mathbf{r}, x)$ takes commitments, randomness and message, and checks that

$$\mathbf{f} = \hat{\mathbf{A}} \cdot \mathbf{r} + \begin{bmatrix} 0 \\ x \end{bmatrix} \quad \text{and} \quad \|\mathbf{r}\|_2 \leq B.$$

TCGen(cpp) takes a commitment parameter and samples $\bar{\mathbf{A}} \in R_q^{2\times\ell}$ of the form:

$$\bar{\mathbf{A}} = \begin{bmatrix} \bar{a}_{1,1} & \bar{a}_{1,2} & \bar{a}_{1,3} & \cdots & \bar{a}_{1,\ell} \\ 0 & 1 & \bar{a}_{2,3} & \cdots & \bar{a}_{2,\ell} \end{bmatrix}$$

where all the $\bar{a}_{i,j}$ are uniform in R_q, except $\bar{a}_{1,1}$ which is uniform in $R_q^\times$. It also samples $\mathbf{R} \leftarrow\$ D_{\bar{s}}^{\ell\times 2w}$ with discrete Gaussian entries. It then outputs $\mathbf{R}$ as the trapdoor td and $\hat{\mathbf{A}} = \left[\bar{\mathbf{A}} | \mathbf{G} - \bar{\mathbf{A}}\mathbf{R}\right]$ as the commitment key ck, where $\mathbf{G}$ is given by:

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & \cdots & 2^{w-1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 2 & \cdots & 2^{w-1} \end{bmatrix} \in R^{2\times 2w}.$$

TCommit$_{ck}(td)$ simply returns a uniformly random commitment $\mathbf{f} \leftarrow\$ R_q^{2\times 1}$. There is no need to keep a state.

Eqv$_{ck}(\mathbf{R}, \mathbf{f}, x)$ uses the trapdoor discrete Gaussian sampling algorithm of Micciancio–Peikert [MP12, Algorithm 3] (or faster variants such as the one described in [GM18]) to sample $\mathbf{r} \leftarrow\$ D_{\Lambda_{\mathbf{u}}^\perp(\hat{\mathbf{A}}), s}$ according to the discrete Gaussian of parameter s supported on the lattice coset:

$$\Lambda_{\mathbf{u}}^\perp(\hat{\mathbf{A}}) = \left\{\mathbf{z} \in R^{\ell+2w} \;:\; \hat{\mathbf{A}} \cdot \mathbf{z} \equiv \mathbf{u} \pmod{q}\right\} \quad \text{where} \quad \mathbf{u} = \mathbf{f} - \begin{bmatrix} 0 \\ x \end{bmatrix}.$$

Fig. 7. Equivocable variant of the commitment from [BDL+18].

for verification should be large enough to accomodate valid commitments, and small enough compared to q to still make the scheme computationally binding (which corresponds to the hardness of an underlying Ring-SIS problem). Let us now discuss the properties one by one.

– **Correctness**. By construction. We select the bound B as $C \cdot s \cdot \sqrt{N}(\sqrt{\ell + 2w} + 1)$ where $C \approx 1/\sqrt{2\pi}$ is the constant in [MP12, Lemma 2.9]. By that lemma, this ensures that the probability to retry in the committing algorithm is negligible (and in particular, the distribution of $\mathbf{r}$ after the rejection sampling is statistically close to the original Gaussian).
– **Computationally binding**. Suppose that an adversary can construct a commitment $\mathbf{f}$ on two distinct messages $x \neq x'$, with the associated randomness $\mathbf{r}, \mathbf{r}'$. Since $x \neq x'$, the correctness condition ensures that $\mathbf{r}$ and $\mathbf{r}'$ are distinct and of norm $\leq B$, and satisfy $\hat{\mathbf{A}}_1 \cdot (\mathbf{r} - \mathbf{r}') \equiv 0 \pmod q$ where $\hat{\mathbf{A}}_1$ is the first row of $\hat{\mathbf{A}}$. Therefore, the vector $\mathbf{z} = \mathbf{r} - \mathbf{r}'$ is a solution of the Ring-SIS problem with bound $2B$ associated with $\hat{\mathbf{A}}_1$, and finding such a solution is hard.

  Note that, technically, the distribution of $\hat{\mathbf{A}}_1$ differs slightly from the standard Ring-SIS distribution, due to the fact that the first entry is invertible. However, for any prime q, this variant reduces tightly to standard Ring-SIS, because a random row vector in $R_q^{\ell+2w}$ contains an invertible entry except with probability at most $(N/q)^{\ell+2w} = 1/N^{\Omega(\log N)}$, which is negligible.
– **Statistically hiding**. It suffices to make sure that

$$\hat{\mathbf{A}} \cdot D_s^{\ell+2w} \approx_s U(R_q^2)$$

  with high probability on the choice of $\hat{\mathbf{A}}$. This is addressed by [LPR13, Corollary 7.5], which shows that it suffices to pick $s > 2N \cdot q^{(2+2/N)/(\ell+2w)}$.
– **Indistinguishability of the trapdoor**. To ensure that the commitment key $\hat{\mathbf{A}}$ generated by TCGen is indistinguishable from a regular commitment key, it suffices to ensure that $\bar{\mathbf{A}}\mathbf{R}$ is statistically close to uniform. Again by [LPR13, Corollary 7.5], this is guaranteed for $\bar{s} > 2N \cdot q^{(2+2/N)/\ell}$. By setting $\ell = w = \lceil \log_2 q \rceil$, we can thus pick $\bar{s} = \Theta(N)$.
– **Equivocability**. It is clear that an $\mathbf{r}$ sampled according to the given lattice coset discrete Gaussian is distributed as in the regular commitment algorithm (up to the negligible statistical distance due to rejection sampling). The only constraint is thus on the Gaussian parameter that can be achieved by the trapdoor Gaussian sampling algorithm. By [MP12, §5.4], the constraint on s is as follows:

$$s \geq \|\mathbf{R}\| \cdot \omega(\sqrt{\log N})$$

  where $\|\mathbf{R}\| \leq C \cdot \bar{s}\sqrt{N}(\sqrt{\ell} + \sqrt{2w} + 1)$ by [MP12, Lemma 2.9]. Thus, one can pick $s = \Theta(N^{3/2} \log^2 N)$.

To summarize, we can choose the parameters as follows in terms of N:

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

Note that we did not strive for optimality in the parameter selection; a more careful analysis can likely yield a more compact scheme.

Furthermore, although the commitment has a linear structure that gives it homomorphic features, we need to increase parameters slightly to support additive homomorphism: this is because the standard deviation of the sum of n randomness vectors $\mathbf{v}$ is $\sqrt{n}$ times larger. Therefore, B (and accordingly q) should be increased by a factor of $\sqrt{n}$ to accomodate for n-party additive homomorphism. For constant n, of course, this does not affect the asymptotic efficiency.

References

AF04. M. Abe and S. Fehr. Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. In *CRYPTO 2004*, vol. 3152 of *LNCS*, pp. 317–334. Springer, Heidelberg, 2004.

AFLT16. M. Abdalla, P.-A. Fouque, V. Lyubashevsky, and M. Tibouchi. Tightly secure signatures from lossy identification schemes. *Journal of Cryptology*, 29(3):597–631, 2016.

BAA$^+$19. N. Bindel, S. Akleylek, E. Alkim, P. S. L. M. Barreto, J. Buchmann, E. Eaton, G. Gutoski, J. Kramer, P. Longa, H. Polat, J. E. Ricardini, and G. Zanon. qTESLA. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

BBE$^+$18. G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, B. Grégoire, M. Rossi, and M. Tibouchi. Masking the GLP lattice-based signature scheme at any order. In *EUROCRYPT 2018, Part II*, vol. 10821 of *LNCS*, pp. 354–384. Springer, Heidelberg, 2018.

BBE$^+$19. G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, M. Rossi, and M. Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In *ACM CCS 2019*, pp. 2147–2164. ACM Press, 2019.

BCJ08. A. Bagherzandi, J. H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In *ACM CCS 2008*, pp. 449–458. ACM Press, 2008.

BCK$^+$14. F. Benhamouda, J. Camenisch, S. Krenn, V. Lyubashevsky, and G. Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In *ASIACRYPT 2014, Part I*, vol. 8873 of *LNCS*, pp. 551–572. Springer, Heidelberg, 2014.

BDG20. M. Bellare, H. Davis, and F. Günther. Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In *EUROCRYPT 2020, Part II*, vol. 12106 of *LNCS*, pp. 3–32. Springer, Heidelberg, 2020.

BDL$^+$18. C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert. More efficient commitments from structured lattice assumptions. In *SCN 18*, vol. 11035 of *LNCS*, pp. 368–385. Springer, Heidelberg, 2018.

BGG$^+$18. 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 *CRYPTO 2018, Part I*, vol. 10991 of *LNCS*, pp. 565–596. Springer, Heidelberg, 2018.

BKLP15. F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *ESORICS 2015, Part I*, vol. 9326 of *LNCS*, pp. 305–325. Springer, Heidelberg, 2015.

BKP13. R. Bendlin, S. Krehbiel, and C. Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In *ACNS 13*, vol. 7954 of *LNCS*, pp. 218–236. Springer, Heidelberg, 2013.

BLS19. J. Bootle, V. Lyubashevsky, and G. Seiler. Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, pp. 176–202. Springer, Heidelberg, 2019.

BN06. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 2006*, pp. 390–399. ACM Press, 2006.

BR93. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pp. 62–73. ACM Press, 1993.

BS13. S. Bettaieb and J. Schrek. Improved lattice-based threshold ring signature scheme. In *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pp. 34–51. Springer, Heidelberg, 2013.

BS16. R. E. Bansarkhani and J. Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In *CANS 16*, vol. 10052 of *LNCS*, pp. 140–155. Springer, Heidelberg, 2016.

CCL$^+$19. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Two-party ECDSA from hash proof systems and efficient instantiations. In *CRYPTO 2019, Part III*, vol. 11694 of *LNCS*, pp. 191–221. Springer, Heidelberg, 2019.

CCL$^+$20. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker. Bandwidth-efficient threshold EC-DSA. In *PKC 2020, Part II*, vol. 12111 of *LNCS*, pp. 266–296. Springer, Heidelberg, 2020.

CHKP10. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 523–552. Springer, Heidelberg, 2010.

CK16. R. Choi and K. Kim. Lattice-based multi-signature with linear homomorphism. In *2016 Symposium on Cryptography and Information Security (SCIS 2016)*, 2016.

CLRS10. P. Cayrel, R. Lindner, M. Rückert, and R. Silva. A lattice-based threshold ring signature scheme. In *LATINCRYPT 2010*, vol. 6212 of *LNCS*, pp. 255–272. Springer, 2010.

CMP20. R. Canetti, N. Makriyannis, and U. Peled. Uc non-interactive, proactive, threshold ecdsa. Cryptology ePrint Archive, Report 2020/492, 2020. https://eprint.iacr.org/2020/492.

COSV17a. M. Ciampi, R. Ostrovsky, L. Siniscalchi, and I. Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In *TCC 2017, Part I*, vol. 10677 of *LNCS*, pp. 711–742. Springer, Heidelberg, 2017.

COSV17b. M. Ciampi, R. Ostrovsky, L. Siniscalchi, and I. Visconti. Four-round concurrent non-malleable commitments from one-way functions. In *CRYPTO 2017, Part II*, vol. 10402 of *LNCS*, pp. 127–157. Springer, Heidelberg, 2017.

CPS$^+$16. M. Ciampi, G. Persiano, A. Scafuro, L. Siniscalchi, and I. Visconti. Improved OR-composition of sigma-protocols. In *TCC 2016-A, Part II*, vol. 9563 of *LNCS*, pp. 112–141. Springer, Heidelberg, 2016.

CS19. D. Cozzo and N. P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In *17th IMA International Conference on Cryptography and Coding*, vol. 11929 of *LNCS*, pp. 128–153. Springer, Heidelberg, 2019.

Dam00. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 418–430. Springer, Heidelberg, 2000.

DDLL13. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In *CRYPTO 2013, Part I*, vol. 8042 of *LNCS*, pp. 40–56. Springer, Heidelberg, 2013.

DEF$^+$19. 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*, pp. 1084–1101. IEEE Computer Society Press, 2019.

DHSS20. Y. Doröz, J. Hoffstein, J. H. Silverman, and B. Sunar. Mmsat: A scheme for multimessage multiuser signature aggregation. Cryptology ePrint Archive, Report 2020/520, 2020. https://eprint.iacr.org/2020/520.

DJN$^+$20. I. Damgård, T. P. Jakobsen, J. B. Nielsen, J. I. Pagter, and M. B. Østergård. Fast threshold ecdsa with honest majority. Cryptology ePrint Archive, Report 2020/501, 2020. https://eprint.iacr.org/2020/501.

DKLs18. J. Doerner, Y. Kondi, E. Lee, and a. shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy*, pp. 980–997. IEEE Computer Society Press, 2018.

DKLs19. 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*, pp. 1051–1066. IEEE Computer Society Press, 2019.

DKO$^+$19. A. Dalskov, M. Keller, C. Orlandi, K. Shrishak, and H. Shulman. Securing dnssec keys via threshold ecdsa from generic mpc. Cryptology ePrint Archive, Report 2019/889, 2019. https://eprint.iacr.org/2019/889.

DLL$^+$18. L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé. Crystals–dilithium: Digital signatures from module lattices. 2018.

dLS18. R. del Pino, V. Lyubashevsky, and G. Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *ACM CCS 2018*, pp. 574–591. ACM Press, 2018.

DM14. L. Ducas and D. Micciancio. Improved short lattice signatures in the standard model. In *CRYPTO 2014, Part I*, vol. 8616 of *LNCS*, pp. 335–352. Springer, Heidelberg, 2014.

EEE20. M. F. Esgin, O. Ersoy, and Z. Erkin. Post-quantum adaptor signatures and payment channel networks. Cryptology ePrint Archive, Report 2020/845, 2020. https://eprint.iacr.org/2020/845.

ESLL19. M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, pp. 115–146. Springer, Heidelberg, 2019.

ESS$^+$19. M. F. Esgin, R. Steinfeld, A. Sakzad, J. K. Liu, and D. Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *ACNS 19*, vol. 11464 of *LNCS*, pp. 67–88. Springer, Heidelberg, 2019.

FH19. M. Fukumitsu and S. Hasegawa. A tightly-secure lattice-based multisignature. In *Proceedings of the 6th on ASIA Public-Key Cryptography Workshop, APKC@AsiaCCS 2019, Auckland, New Zealand, July 8, 2019*, pp. 3–11. ACM, 2019.

GG18. R. Gennaro and S. Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *ACM CCS 2018*, pp. 1179–1194. ACM Press, 2018.

GG20. R. Gennaro and S. Goldfeder. One round threshold ecdsa with identifiable abort. Cryptology ePrint Archive, Report 2020/540, 2020. https://eprint.iacr.org/2020/540.

GGN16. R. Gennaro, S. Goldfeder, and A. Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In *ACNS 16*, vol. 9696 of *LNCS*, pp. 156–174. Springer, Heidelberg, 2016.

GJKR07. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, 2007.

GKSS20. A. Gagol, J. Kula, D. Straszak, and M. Swietek. Threshold ecdsa for decentralized asset custody. Cryptology ePrint Archive, Report 2020/498, 2020. https://eprint.iacr.org/2020/498.

GLP12. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES 2012*, vol. 7428 of *LNCS*, pp. 530–547. Springer, Heidelberg, 2012.

GM18. N. Genise and D. Micciancio. Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In *EUROCRYPT 2018, Part I*, vol. 10820 of *LNCS*, pp. 174–203. Springer, Heidelberg, 2018.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *40th ACM STOC*, pp. 197–206. ACM Press, 2008.

GSW13. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO 2013, Part I*, vol. 8042 of *LNCS*, pp. 75–92. Springer, Heidelberg, 2013.

GVW15. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *47th ACM STOC*, pp. 469–477. ACM Press, 2015.

HJ10. N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In *EUROCRYPT 2010*, vol. 6110 of *LNCS*, pp. 235–256. Springer, Heidelberg, 2010.

KD20. M. Kansal and R. Dutta. Round optimal secure multisignature schemes from lattice with public key aggregation and signature compression. In *AFRICACRYPT 20*, vol. 12174 of *LNCS*, pp. 281–300. Springer, Heidelberg, 2020.

KG20. C. Komlo and I. Goldberg. Frost: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Report 2020/852, 2020. https://eprint.iacr.org/2020/852.

KLS18. E. Kiltz, V. Lyubashevsky, and C. Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In *EUROCRYPT 2018, Part III*, vol. 10822 of *LNCS*, pp. 552–586. Springer, Heidelberg, 2018.

LDK$^+$19. V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, and D. Stehlé. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

Lin17. Y. Lindell. Fast secure two-party ECDSA signing. In *CRYPTO 2017, Part II*, vol. 10402 of *LNCS*, pp. 613–644. Springer, Heidelberg, 2017.

LN18. Y. Lindell and A. Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *ACM CCS 2018*, pp. 1837–1854. ACM Press, 2018.

LNTW19. B. Libert, K. Nguyen, B. H. M. Tan, and H. Wang. Zero-knowledge elementary databases with more expressive queries. In *PKC 2019, Part I*, vol. 11442 of *LNCS*, pp. 255–285. Springer, Heidelberg, 2019.

LPR13. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT 2013*, vol. 7881 of *LNCS*, pp. 35–54. Springer, Heidelberg, 2013.

LS18. V. Lyubashevsky and G. Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In *EUROCRYPT 2018, Part I*, vol. 10820 of *LNCS*, pp. 204–224. Springer, Heidelberg, 2018.

Lyu09. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, vol. 5912 of *LNCS*, pp. 598–616. Springer, Heidelberg, 2009.

Lyu12. V. Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, vol. 7237 of *LNCS*, pp. 738–755. Springer, Heidelberg, 2012.

Lyu19. V. Lyubashevsky. Lattice-based zero-knowledge and applications. CIS 2019, 2019. https://crypto.sjtu.edu.cn/cis2019/slides/Vadim.pdf.

MJ19. C. Ma and M. Jiang. Practical lattice-based multisignature schemes for blockchains. *IEEE Access*, 7:179765–179778, 2019.

MOR01. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *ACM CCS 2001*, pp. 245–254. ACM Press, 2001.

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT 2012*, vol. 7237 of *LNCS*, pp. 700–718. Springer, Heidelberg, 2012.

MP13. D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO 2013, Part I*, vol. 8042 of *LNCS*, pp. 21–39. Springer, Heidelberg, 2013.

MPSW19. G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019.

MR01. P. D. MacKenzie and M. K. Reiter. Two-party generation of DSA signatures. In *CRYPTO 2001*, vol. 2139 of *LNCS*, pp. 137–154. Springer, Heidelberg, 2001.

MWLD10. C. Ma, J. Weng, Y. Li, and R. H. Deng. Efficient discrete logarithm based multi-signature scheme in the plain public key model. *Des. Codes Cryptogr.*, 54(2):121–133, 2010.

Ngu19. N. K. Nguyen. On the non-existence of short vectors in random module lattices. In *ASIACRYPT 2019, Part II*, vol. 11922 of *LNCS*, pp. 121–150. Springer, Heidelberg, 2019.

NKDM03. A. Nicolosi, M. N. Krohn, Y. Dodis, and D. Mazières. Proactive two-party signatures for user authentication. In *NDSS 2003*. The Internet Society, 2003.

NRSW20. J. Nick, T. Ruffing, Y. Seurin, and P. Wuille. Musig-dn: Schnorr multi-signatures with verifiably deterministic nonces. Cryptology ePrint Archive, Report 2020/1057, 2020. https://eprint.iacr.org/2020/1057.

Pas03. R. Pass. On deniability in the common reference string and random oracle model. In *CRYPTO 2003*, vol. 2729 of *LNCS*, pp. 316–337. Springer, Heidelberg, 2003.

Ped92. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO'91*, vol. 576 of *LNCS*, pp. 129–140. Springer, Heidelberg, 1992.

PS96. D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, vol. 1070 of *LNCS*, pp. 387–398. Springer, Heidelberg, 1996.

Sch90. C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, vol. 435 of *LNCS*, pp. 239–252. Springer, Heidelberg, 1990.

SS01. D. R. Stinson and R. Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In *ACISP 01*, vol. 2119 of *LNCS*, pp. 417–434. Springer, Heidelberg, 2001.

STV$^+$16. E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy*, pp. 526–545. IEEE Computer Society Press, 2016.

TE19. R. Toluee and T. Eghlidos. An efficient and secure ID-based multi-proxy multi-signature scheme based on lattice. Cryptology ePrint Archive, Report 2019/1031, 2019. https://eprint.iacr.org/2019/1031.

TLT19. R. Tso, Z. Liu, and Y. Tseng. Identity-based blind multisignature from lattices. *IEEE Access*, 7:182916–182923, 2019.

Wag02. D. Wagner. A generalized birthday problem. In *CRYPTO 2002*, vol. 2442 of *LNCS*, pp. 288–303. Springer, Heidelberg, 2002.

YAZ^{+}19. R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO 2019, Part I*, vol. 11692 of *LNCS*, pp. 147–175. Springer, Heidelberg, 2019.

Protocol Lattice-based statistically binding commitment

$\mathsf{CSetup}(1^\lambda)$ takes a security parameter and outputs $cpp = (q, N, k, m, m', \eta)$.

$\mathsf{CGen}(cpp)$ takes a commitment parameter and outputs ck consisting of $\hat{\mathbf{A}}_1 \in R_q^{m \times m'}$ and $\hat{\mathbf{A}}_2 \in R_q^{k \times m'}$.

$$\hat{\mathbf{A}}_1 = [\mathbf{I}_m | \hat{\mathbf{A}}_1'] \text{ where } \hat{\mathbf{A}}_1' \leftarrow\!\!\$\, R_q^{m \times (m'-m)}$$
$$\hat{\mathbf{A}}_2 = [\mathbf{0}^{k \times m} | \mathbf{I}_k | \hat{\mathbf{A}}_2'] \text{ where } \hat{\mathbf{A}}_2' \leftarrow\!\!\$\, R_q^{k \times (m'-m-k)}$$

$\mathsf{Commit}_{ck}(\mathbf{x})$ takes $\mathbf{x} \in R_q^k$, samples the randomness vector $\mathbf{r} \leftarrow\!\!\$\, D_s^{m'}$ and outputs

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{A}}_1 \\ \hat{\mathbf{A}}_2 \end{bmatrix} \cdot \mathbf{r} + \begin{bmatrix} \mathbf{0}^m \\ \mathbf{x} \end{bmatrix}.$$

$\mathsf{Open}_{ck}(\mathbf{f}_1, \mathbf{f}_2, \mathbf{x}, \mathbf{r})$ checks that

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{A}}_1 \\ \hat{\mathbf{A}}_2 \end{bmatrix} \cdot \mathbf{r} + \begin{bmatrix} \mathbf{0}^m \\ \mathbf{x} \end{bmatrix} \text{ and } \|\mathbf{r}\|_2 \leq B$$

Fig. 8. Statistically binding homomorphic commitment from [BDL+18]

A Lattice-based Statistically Binding Commitment

In Fig. 8, we give a formal specification of our statistically binding variant Baum et al.'s commitment scheme [BDL+18].

B Security Proof for DS$_3$

Here we give a full security proof for our three-round protocol DS$_3$ (see Figs. 3 to 5 for the protocol specification).

Theorem 1. *Suppose the commitment scheme* COM *is unconditionally binding, computationally hiding, uniform, additively homomorphic, and the output of committing algorithm* Commit *has ξ-bit min-entropy. Assume the modulus q satisfies $q = 5 \mod 8$, $2B_n < \sqrt{q/2}$ and $2\kappa < \sqrt{q/2}$. For any probabilistic polynomial-time adversary $\mathcal{A}$ that makes a single query to* OGen*, at most Q_s queries to* OSign *and at most Q_h queries in total to the random oracles $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4$, the protocol $\mathsf{DS}_3 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ is* DS-UF-CMA *secure under $\mathsf{MLWE}_{q,k,\ell,\eta}$ assumption.*

Proof. Suppose we are given $\mathcal{A}$ that breaks DS$_3$ with advantage $\mathbf{Adv}_{\mathsf{DS}_3}^{\mathsf{DS\text{-}UF\text{-}CMA}}(\mathcal{A})$. Without loss of generality we assume that P_n is an honest party. Our first goal is to construct an algorithm $\mathcal{B}$ around $\mathcal{A}$ that simulates the behaviors of P_n without using honestly generated key pairs. In Figs. 10 and 11 we present the resulting oracle simulators SimOGen and SimOSign which are eventually invoked by $\mathcal{B}$. Below we discuss how these are derived via several intermediate hybrid games.

G$_0$ Random Oracle simulation. The random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$, $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{l_1}$, $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{l_2}$, $\mathsf{H}_3 : \{0,1\}^* \to S_{ck}$ and $\mathsf{H}_4 : \{0,1\}^* \to \{0,1\}^{l_4}$ are are simulated as follows.

$\mathsf{H}_i(x)$ The table HT_i is initially empty. When queried with x, if $\mathrm{HT}_i[x]$ is set then return $\mathrm{HT}_i[x]$. Otherwise sample y from H_i's image uniformly at random and return $\mathrm{HT}_i[x] := y$.

Honest Party Oracle simulation. In this game $\mathcal{B}$ behaves exactly like a single honest party in DS$_3$ to simulate OSign and OGen. Moreover, once the key generation phase is done and the adversary initiated SIGN with some message μ it gets included in the list $\mathcal{M}$.

Forgery. When $\mathcal{A}$ outputs a forgery $(\mu^*, com, \mathbf{z}, r)$ at the end $\mathcal{B}$ first generates a commitment key $ck \leftarrow \mathsf{H}_3(\mu^*, \mathbf{t})$, derives a challenge $c \leftarrow \mathsf{H}_0(com, \mu^*, \mathbf{t})$ and reconstructs $\mathbf{w} = \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$. Then $\mathcal{B}$ checks $\mu^* \notin \mathcal{M}$ and the verification condition

$$\|\mathbf{z}\|_2 \leq B_n \quad \text{and} \quad \mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1.$$

If the forgery is verified then $\mathcal{B}$ outputs 1. Otherwise $\mathcal{B}$ outputs 0. Let $\Pr[\mathbf{G}_i]$ denote a probability that $\mathcal{B}$ returns 1 at the game $\mathbf{G}_i$. Then we have

$$\Pr[\mathbf{G}_0] = \mathbf{Adv}_{\mathsf{DS}_3}^{\mathsf{DS\text{-}UF\text{-}CMA}}(\mathcal{A}).$$

$\mathbf{G}_1$ In this game we modify $\mathcal{B}$ from the prior game so that it first picks a random challenge $c \leftarrow_{\$} C$ and computes its own signature share $\mathbf{z}_n$ without interacting with adversary. Then the oracle proceeds as in the previous game and sends out hash commitment h_n. Upon receiving $h_1, \dots, h_{n-1}$, the oracle searches the hash table HT_0 to check if there exists the corresponding preimages $com_1, \dots, com_{n-1}$. If it's successful, then let $com = \sum_j com_j$ and program the random oracle so that $\mathrm{HT}_0[com, \mu, \mathbf{t}] := c$. Otherwise simulation fails. Since $\mathbf{G}_1$ is identical to $\mathbf{G}_0$ from adversary $\mathcal{A}$'s point of view except at the *bad* events marked in Fig. 10, we have

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_0]| \le \Pr[bad_1] + \Pr[bad_2] + \Pr[bad_3] \le \frac{(Q_h + nQ_s + 1)^2}{2^{l_4+1}} + Q_s\left(\frac{Q_h + nQ_s}{2^{\xi}} + \frac{Q_h + Q_s}{2^{\xi}} + \frac{n}{2^{l_4}}\right)$$

where $\Pr[bad_1]$ corresponds to the probability that at least one collision occurs during at most $Q_h + nQ_s$ queries to H_4 made by $\mathcal{A}$ or $\mathcal{B}$; $\Pr[bad_2]$ is the probability that programming the random oracle H_0 fails at least once during Q_s trials due to either of two cases: 1) $\mathsf{H}_4(com_n)$ has been asked by $\mathcal{A}$ during at most $Q_h + nQ_s$ queries to H_4 (and therefore $\mathcal{A}$ knows com and could query $\mathsf{H}_0(com, \mu, \mathbf{t})$ deliberately), which could succeed with probability at most $1/2^{\xi}$ for each query, or 2) $\mathrm{HT}_0[com, \mu, \mathbf{t}]$ has been set by $\mathcal{A}$ or $\mathcal{B}$ by chance during at most $Q_h + Q_s$ prior queries to H_0, which could happen with probability at most $(Q_h + Q_s)/2^{\xi}$; $\Pr[bad_3]$ is the probability that $\mathcal{A}$ has predicted one of the $n-1$ outputs of random oracle H_4 without making a query to it, which could only happen with probability at most $n/2^{l_4}$ for each sign query. We remark that the above probability bound is essentially a special case of the one given by [BN06].

$\mathbf{G}_2$ In this game we modify $\mathcal{B}$ from the prior game so that if $\mathbf{z}_n$ gets rejected then it commits to some uniformly random vector $\mathbf{w}_n \in R_q^k$ and sends out hash of corresponding commitment $h_n = \mathsf{H}_4(com_n)$, where $com_n \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_n; r_n)$ and $r_n \leftarrow_{\$} D(S_r)$. Note that the adversary cannot distinguish this simulated com_n from the real one due to the hiding property of commitment. In other words, we have

$$|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1]| \le Q_s \cdot \epsilon_{\mathrm{hide}}.$$

$\mathbf{G}_3$ In this game $\mathcal{B}$ doesn't honestly generate $\mathbf{z}_n$ anymore and instead simulates the rejection sampling as follows. With probability $1 - 1/M$ (i.e., simulation of rejection), it generates commitment com_n to $\mathbf{w}_n \leftarrow_{\$} R_q^k$ as before. Otherwise it samples $\mathbf{z}_n$ from $D_s^{\ell+k}$ and computes $\mathbf{w}_n = \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n$. The signature share $\mathbf{z}_n$ generated this way is indistinguishable from the real one because of the special HVZK property of the underlying identification scheme. In other words, we can directly apply the result of Lemmas 3 and 4. Hence we have

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2]| \le Q_s \cdot \frac{e^{-t^2/2}}{M}.$$

At this point $\mathcal{B}$ simulates the honest party's behavior during signature generation by following $\mathsf{SimOSign}$ in Fig. 10.

$\mathbf{G}_4$ Now notice that signing phase doesn't rely on the actual secret key share $\mathbf{s}_n$ anymore. So the next step is to simulate the generation of $\mathbf{t}_n$ without using $\mathbf{s}_n$. In this game the public key share $\mathbf{t}_n$ is picked randomly from R_q^k during the key generation phase. Due to the hardness of $\mathsf{MLWE}_{q,k,\ell,\eta}$ the adversary cannot distinguish simulated $\mathbf{t}_n$ from the real one, and hence we have

$$|\Pr[\mathbf{G}_4] - \Pr[\mathbf{G}_3]| \le \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}}(\mathcal{A}).$$

$\mathbf{G}_5$ In this game $\mathcal{B}$ first picks the resulting public key $\mathbf{t}$ randomly from R_q^k and defines its own share $\mathbf{t}_n$ a posteriori, after extracting adversary's committed shares $\mathbf{t}_1, \dots, \mathbf{t}_{n-1}$. This can be done by searching the recorded random oracle queries in HT_2. Note that the distributions of $\mathbf{t}$ and $\mathbf{t}_n$ haven't changed from the previous game. Since $\mathbf{G}_5$ is identical to $\mathbf{G}_4$ from adversary $\mathcal{A}$'s point of view except at the *bad'* events marked in Fig. 11, we have

$$|\Pr[\mathbf{G}_5] - \Pr[\mathbf{G}_4]| \le \Pr[bad'_4] + \Pr[bad'_5] + \Pr[bad'_6] \le \frac{(Q_h + n + 1)^2}{2^{l_2+1}} + \frac{Q_h}{q^{kN}} + \frac{n}{2^{l_2}}$$

where $\Pr[bad'_4]$ corresponds to the probability that at least one collision occurs during at most $Q_h + n$ queries to H_2 made by $\mathcal{A}$ or $\mathcal{B}$; $\Pr[bad'_5]$ is the probability that programming the random oracle H_2 fails, which happens only if $\mathsf{H}_2(\mathbf{t}_n, n)$ has been previously asked by $\mathcal{A}$ during at most Q_h queries to H_2, and the probability that guessing a uniformly random $\mathbf{t}_n$ by chance is at most $1/q^{kN}$ for each query; $\Pr[bad'_6]$ is the probability that $\mathcal{A}$ has predicted one of the $n-1$ outputs of random oracle H_2 without making a query to it, which could only happen with probability at most $n/2^{l_2}$.

$\mathbf{G}_6$ In this game $\mathcal{B}$ first picks the resulting random matrix $\mathbf{A} \in R_q^{k\times\ell}$ and defines its own share $\mathbf{A}_n$ a posteriori, after extracting adversary's committed shares $\mathbf{A}_1, \ldots, \mathbf{A}_{n-1}$. This can be done by searching the recorded random oracle queries in HT_3. Note that the distributions of $\mathbf{A}$ and $\mathbf{A}_n$ haven't changed from the previous game. Since $\mathbf{G}_6$ is identical to $\mathbf{G}_5$ from adversary $\mathcal{A}$'s point of view except at the *bad* events marked in Fig. 11, we have

$$|\Pr[\mathbf{G}_6] - \Pr[\mathbf{G}_5]| \leq \Pr[bad_4] + \Pr[bad_5] + \Pr[bad_6] \leq \frac{(Q_h + n + 1)^2}{2^{l_1+1}} + \frac{Q_h}{q^{k\ell N}} + \frac{n}{2^{l_1}}$$

where the bounds are calculated just as in $\mathbf{G}_5$.

Now $\mathcal{B}$ entirely simulates the behaviors of honest party by invoking $\mathsf{SimOGen}$ and $\mathsf{SimOSign}$, which don't rely on the secret key share $\mathbf{s}_n$. We would like to evaluate the upper bound of $\Pr[\mathbf{G}_6]$. We first argue that the following probability is negligible.

$$\Pr_{\mathbf{A} \leftarrow\$ R_q^{k\times\ell}, \mathbf{t} \leftarrow\$ R_q^k, ck \leftarrow\$ S_{ck}}\left[\exists(com, c, \mathbf{z}, r, c', \mathbf{z}', r') : \begin{array}{r} c \neq c' \wedge \|\mathbf{z}\|_2 \leq B_n \wedge \|\mathbf{z}'\|_2 \leq B_n \\ \wedge\ \mathsf{Open}_{ck}(com, r, \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}) \\ = \mathsf{Open}_{ck}(com, r', \bar{\mathbf{A}}\mathbf{z}' - c'\mathbf{t}) = 1 \end{array}\right] \quad (1)$$

Case 1: $\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} \neq \bar{\mathbf{A}}\mathbf{z}' - c'\mathbf{t}$. In this case, (1) is bounded by the probability that there exists some commitment having two openings over random choice of ck, which should be bounded by negligible $\epsilon_{\mathrm{ubind}}$ if the commitment key is uniform (and hence $ck \leftarrow\$ S_{ck}$ can be regarded as if it was generated from CGen) and if unconditionally binding holds (see Definition 4).

Case 2: $\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} = \bar{\mathbf{A}}\mathbf{z}' - c'\mathbf{t}$. Let $\mathbf{z}_1 \in R_q^\ell, \mathbf{z}_2 \in R_q^k, \mathbf{z}'_1 \in R_q^\ell, \mathbf{z}'_2 \in R_q^k$ be such that $\mathbf{z} = \begin{bmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{bmatrix}$ and $\mathbf{z}' = \begin{bmatrix}\mathbf{z}'_1\\ \mathbf{z}'_2\end{bmatrix}$, we have

$$\mathbf{A}(\mathbf{z}_1 - \mathbf{z}'_1) + \mathbf{z}_2 - \mathbf{z}'_2 = (c - c')\mathbf{t}$$

where we used the fact that $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}]$. Hence, the probability (1) in this case is bounded by

$$2 \cdot |\bar{C}| \cdot \frac{(4B_n + 1)^{(\ell+k)N}}{q^{kN}}$$

by applying Lemma 5 with $\bar{\mathbf{z}} = \mathbf{z} - \mathbf{z}'$, $\bar{c} = c - c'$, $\beta = 2B_n$, and

$$\bar{C} = \{\bar{c} \in R : \bar{c} = c - c' \wedge c \in C \wedge c' \in C \wedge c \neq c'\}.$$

If the event for (1) doesn't occur, then it means that for given $com \in S_{com}$ there exists at most one transcript that verifies. In that case $\mathcal{A}$ has at most a $1/|C| =$ chance of obtaining the correct challenge for each query to H_0 with input $(com, \mu^*, \mathbf{t})$ if $\mu^* \notin \mathcal{M}$.

Since $\mathcal{A}$ makes at most Q_h queries to H_0 and H_3 in total and $\mathcal{B}$ makes a single query to H_0 and H_3 at the forgery phase, we have

$$\Pr[\mathbf{G}_6] \leq (Q_h + 1)\left(\epsilon_{\mathrm{ubind}} + 2 \cdot |\bar{C}| \cdot \frac{(4B_n + 1)^{(\ell+k)N}}{q^{kN}} + \frac{1}{|C|}\right).$$

The following lemma is a slightly modified version of Lemma 4.6 of [KLS18]. The main difference is that we use the Euclidean norm instead of ∞-norm.

Lemma 5. *Let β be a positive integer less than $\sqrt{q/2}$ and $\bar{C}$ be a set of elements in $R \setminus \{\mathbf{0}\}$ with coefficients less than $\sqrt{q/2}$. If $q = 5 \mod 8$ then*

$$\Pr_{\mathbf{A} \leftarrow\$ R_q^{k\times\ell}, \mathbf{t} \leftarrow\$ R_q^k}\left[\exists(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2, \bar{c}) \in R_q^\ell \times R_q^k \times \bar{C} : \mathbf{A}\bar{\mathbf{z}}_1 + \bar{\mathbf{z}}_2 = \bar{c}\mathbf{t} \wedge \|\bar{\mathbf{z}}\|_2 \leq \beta\right] \leq 2 \cdot |\bar{C}| \cdot \frac{(2\beta + 1)^{(\ell+k)N}}{q^{kN}}$$

where $\bar{\mathbf{z}} = \begin{bmatrix}\bar{\mathbf{z}}_1\\ \bar{\mathbf{z}}_2\end{bmatrix}$.

Proof. **Case $\bar{\mathbf{z}}_1 = \mathbf{0}$.** Since $0 \le \|\bar{c}\|_\infty \le \sqrt{q/2}$ and $q = 5 \mod 8$, Lemma 2.2 by Lyubashevsky and Seiler [LS18] guarantees that $\bar{c}$ is invertible in R_q. In this case the probability is upper-bounded by

$$\begin{aligned}
&\Pr_{\mathbf{t} \leftarrow\$ R_q^k}[\exists(\bar{\mathbf{z}}_2, \bar{c}) \in R_q^k \times \bar{C} : \bar{\mathbf{z}}_2 = \bar{c}\mathbf{t} \wedge \|\bar{\mathbf{z}}_2\|_2 \le \beta] \\
=& \Pr_{\mathbf{t} \leftarrow\$ R_q^k}[\exists(\bar{\mathbf{z}}_2, \bar{c}) \in R_q^k \times \bar{C} : \bar{c}^{-1}\bar{\mathbf{z}}_2 = \mathbf{t} \wedge \|\bar{\mathbf{z}}_2\|_2 \le \beta] \\
\le& \sum_{\bar{\mathbf{z}}_2 \in R_q^k, \bar{c} \in \bar{C}} \Pr_{\mathbf{t} \leftarrow\$ R_q^k}[\bar{c}^{-1}\bar{\mathbf{z}}_2 = \mathbf{t} \wedge \|\bar{\mathbf{z}}_2\|_2 \le \beta] \\
\le& \sum_{\bar{\mathbf{z}}_2 \in R_q^k, \bar{c} \in \bar{C}} \Pr_{\mathbf{t} \leftarrow\$ R_q^k}[\bar{c}^{-1}\bar{\mathbf{z}}_2 = \mathbf{t} \wedge \|\bar{\mathbf{z}}_2\|_\infty \le \beta] \\
=& |\bar{C}| \cdot \left(\frac{2\beta+1}{q}\right)^{kN}
\end{aligned}$$

Case $\bar{\mathbf{z}}_1 \neq \mathbf{0}$. Let $\mathbf{a} \in R_q^k, \mathbf{A}' \in R_q^{k\times(\ell-1)}$ be such that $[\mathbf{a}|\mathbf{A}'] = \mathbf{A}$ and $\bar{z} \in R_q, \bar{\mathbf{z}}_1' \in R_q^{\ell-1}$ be such that $\begin{bmatrix} \bar{z} \\ \bar{\mathbf{z}}_1' \end{bmatrix} = \bar{\mathbf{z}}_1$. Assuming wlog that $\bar{z}$ is non-zero, it is guaranteed that $\bar{z}$ is invertible in R_q since $\|\bar{z}\|_\infty \le \|\bar{z}\|_2 \le \beta \le \sqrt{q/2}$. Hence we obtain the following upper-bound.

$$\begin{aligned}
&\Pr_{\mathbf{A} \leftarrow\$ R_q^{k\times\ell}, \mathbf{t} \leftarrow\$ R_q^k}[\exists(\bar{\mathbf{z}}_1, \bar{\mathbf{z}}_2, \bar{c}) \in R_q^\ell \times R_q^k \times \bar{C} : \mathbf{A}\bar{\mathbf{z}}_1 + \bar{\mathbf{z}}_2 = \bar{c}\mathbf{t} \wedge \|\bar{\mathbf{z}}\|_2 \le \beta] \\
=& \Pr_{\mathbf{a} \leftarrow\$ R_q, \mathbf{A}' \leftarrow\$ R_q^{k\times(\ell-1)}, \mathbf{t} \leftarrow\$ R_q^k}[\exists(\bar{z}, \bar{\mathbf{z}}_1', \bar{\mathbf{z}}_2, \bar{c}) \in R_q \times R_q^{\ell-1} \times R_q^k \times \bar{C} : \bar{z}\mathbf{a} + \mathbf{A}'\bar{\mathbf{z}}_1' + \bar{\mathbf{z}}_2 = \bar{c}\mathbf{t} \wedge \|\bar{\mathbf{z}}\|_2 \le \beta] \\
=& \Pr_{\mathbf{a} \leftarrow\$ R_q}[\exists(\bar{z}, \bar{\mathbf{z}}_1', \bar{\mathbf{z}}_2, \bar{c}) \in R_q \times R_q^{\ell-1} \times R_q^k \times \bar{C} : \mathbf{a} = \bar{z}^{-1}(\bar{c}\mathbf{t} - \mathbf{A}'\bar{\mathbf{z}}_1' - \bar{\mathbf{z}}_2) \wedge \|\bar{\mathbf{z}}\|_2 \le \beta] \\
\le& \sum_{\bar{\mathbf{z}}_1 \in R_q^\ell \setminus \{\mathbf{0}\}, \bar{\mathbf{z}}_2 \in R_q^k, \bar{c} \in \bar{C}} \Pr[\mathbf{a} = \bar{z}^{-1}(\bar{c}\mathbf{t} - \mathbf{A}'\bar{\mathbf{z}}_1' - \bar{\mathbf{z}}_2) \wedge \|\bar{\mathbf{z}}\|_2 \le \beta] \\
\le& \sum_{\bar{\mathbf{z}}_1 \in R_q^\ell \setminus \{\mathbf{0}\}, \bar{\mathbf{z}}_2 \in R_q^k, \bar{c} \in \bar{C}} \Pr[\mathbf{a} = \bar{z}^{-1}(\bar{c}\mathbf{t} - \mathbf{A}'\bar{\mathbf{z}}_1' - \bar{\mathbf{z}}_2) \wedge \|\bar{\mathbf{z}}\|_\infty \le \beta] \\
\le& \sum_{\bar{\mathbf{z}}_1 \in R_q^\ell \setminus \{\mathbf{0}\}, \bar{\mathbf{z}}_2 \in R_q^k, \bar{c} \in \bar{C}} \Pr[\mathbf{a} = \bar{z}^{-1}(\bar{c}\mathbf{t} - \mathbf{A}'\bar{\mathbf{z}}_1' - \bar{\mathbf{z}}_2) \wedge \|\bar{\mathbf{z}}\|_\infty \le \beta] \\
\le& |\bar{C}| \cdot \frac{(2\beta+1)^{(\ell+k)N}}{q^{kN}}
\end{aligned}$$

Putting the two cases together we obtain the result.

Algorithm SearchHashTable

```
Upon receiving the hash table HT together with hash values $(h_1, \dots, h_{n-1})$:
    1. If for some $j \in [n-1]$ the preimage of $h_j$ doesn't exist in HT then set the flag *alert*.
    2. If for some $j \in [n-1]$ more than one preimages of $h_j$ exist in HT then set the flag *bad*.
    3. Return $(alert, bad, m_1, \dots, m_{n-1})$ where $h_j = \mathrm{HT}[m_j]$ for $j \in [n-1]$.
```

Fig. 9. Routine for searching hash tables.

Oracle SimOSign

Initialization Upon receiving any query from the adversary, if MGEN and KEYGEN have not been completed, then return $\perp$. Otherwise, if the query is of the form $(\textsc{sign}, \mu, \mathit{sid})$, then update $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$.

Signature Generation

1. Upon receiving $(\textsc{sign}, \mu, \mathit{sid})$, get $\mathit{ck} \leftarrow \mathsf{H}_3(\mu, \mathbf{t})$, sample $c \leftarrow\!\!\$\, C$ and simulate the rejection sampling as follows.
   a. With probability $1 - 1/M$, sample $\mathbf{w}_n \leftarrow\!\!\$\, R_q^k$; otherwise sample $\mathbf{z}_n \leftarrow\!\!\$\, D_\sigma^{\ell+k}$, define $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n$ and set the flag *accept*. Then generate a commitment $\mathit{com}_n \leftarrow \mathsf{Commit}_{\mathit{ck}}(\mathbf{w}_n; r_n)$ with $r_n \leftarrow\!\!\$\, D(S_r)$, and its hash $h_n \leftarrow \mathsf{H}_4(\mathit{com}_n)$.
   b. Send out $(\textsc{exchange}, h_n, \mathit{sid}, n)$.
2. Upon receiving $(\textsc{exchange}, h_j, \mathit{sid}, j)$ for all $j \in [n-1]$:
   a. Invoke SearchHashTable in Fig. 9 on input HT_4 and $(h_1, \ldots, h_{n-1})$ to obtain $(\mathit{alert}, \mathit{bad}_1, \mathit{com}_1, \ldots, \mathit{com}_{n-1})$.
   b. If the flag bad_1 is set then simulation fails.
   c. If the flag *alert* is set then send out $(\textsc{open}, \mathit{com}_n, \mathit{sid}, n)$.
   d. Otherwise define $\mathit{com} := \sum_{j\in[n]} \mathit{com}_j$ and
      – If $\mathrm{HT}_0[\mathit{com}, \mu, \mathbf{t}]$ has been already set then simulation fails and set the flag bad_2.
      – Otherwise program the random oracle $\mathrm{HT}_0[\mathit{com}, \mu, \mathbf{t}] := c$ and send out $(\textsc{open}, \mathit{com}_n, \mathit{sid}, n)$
3. Upon receiving $(\textsc{open}, \mathit{com}_j, \mathit{sid}, j)$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_4(\mathit{com}_j) \neq h_j$ for some j, then send out ABORT.
   b. If *alert* is set and $\mathsf{H}_4(\mathit{com}_j) = h_j$ holds for all j then simulation fails and set the flag bad_3.
   c. Otherwise send out $(\textsc{proceed}, \mathbf{z}_n, r_n, \mathit{sid}, n)$ if the flag *accept* is set; otherwise send out $(\textsc{restart}, \perp, \perp, \mathit{sid}, n)$.
4. Upon receiving $(\textsc{restart}, \perp, \perp, \mathit{sid}, j)$ from some party go to 1. Otherwise upon receiving $(\textsc{proceed}, \mathbf{z}_j, r_j, \mathit{sid}, j)$ for all $j \in [n-1]$ compute the combined signature as follows.
   a. For each $j \in [n-1]$ reconstruct $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c\mathbf{t}_j$ and validate the signature share:
   $$\|\mathbf{z}_j\|_2 \leq B \quad \text{and} \quad \mathsf{Open}_{\mathit{ck}}(\mathit{com}_j, r_j, \mathbf{w}_j) = 1.$$
   If the check fails for some j then send out ABORT.
   b. Compute $\mathbf{z} := \sum_{j\in[n]} \mathbf{z}_j$ and $r := \sum_{j\in[n]} r_j$.
   c. Output $(\mathit{com}, \mathbf{z}, r)$ as a signature.

Fig. 10. Oracle simulator for the honest party P_n during the signature generation phase.

Oracle SimOGen

Matrix Generation :

1. Upon receiving $(\textsc{mgen}, pp, 0)$, sample $g_n \leftarrow_\$ \{0,1\}^{l_1}$ and send out $(\textsc{exchange}, g_n, 0, n)$.
2. Upon receiving $(\textsc{exchange}, g_j, 0, j)$ for all $j \in [n-1]$ proceed as follows:
   a. Invoke SearchHashTable in Fig. 9 on input HT_1 and $(g_1, \dots, g_{n-1})$ to obtain $(alert, bad_4, (\mathbf{A}_1, 1), \dots, (\mathbf{A}_{n-1}, n-1))$.
   b. If the flag bad_4 is set then simulation fails.
   c. If the flag $alert$ is set then pick $\mathbf{A}_n \leftarrow_\$ R_q^{k\times\ell}$. Otherwise pick $\mathbf{A} \leftarrow_\$ R_q^{k\times\ell}$ and define $\mathbf{A}_n \coloneqq \mathbf{A} - \sum_{j=1}^{n-1} \mathbf{A}_j$.
      - If $\mathrm{HT}_1[\mathbf{A}_n, n]$ has been already set then simulation fails and set bad_5.
      - Otherwise program the random oracle $\mathrm{HT}_1[\mathbf{A}_n, n] \coloneqq g_n$ and send out $(\textsc{open}, \mathbf{A}_n, 0, n)$.
3. Upon receiving $(\textsc{open}, \mathbf{A}_j, 0, j)$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_1(\mathbf{A}_j, j) \neq g_j$ for some j then send out $\textsc{abort}$.
   b. If $alert$ is set and $\mathsf{H}_1(\mathbf{A}_j, j) = g_j$ for all j then simulation fails and set bad_6.
   c. Otherwise define $\bar{\mathbf{A}} \coloneqq [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$ and ignore the future invocation of $\textsc{mgen}$.

Key Pair Generation :

1. Upon receiving $(\textsc{keygen}, pp, 1)$, sample $g'_n \leftarrow_\$ \{0,1\}^{l_2}$ and send out $(\textsc{exchange}, g'_n, 1, n)$.
2. Upon receiving $(\textsc{exchange}, g'_j, 1, j)$ for all $j \in [n-1]$ proceed as follows:
   a. Invoke SearchHashTable in Fig. 9 on input HT_2 and $(g'_1, \dots, g'_{n-1})$ to obtain $(alert', bad'_4, (\mathbf{t}_1, 1), \dots, (\mathbf{t}_{n-1}, n-1))$.
   b. If the flag bad'_4 is set then simulation fails.
   c. If the flag $alert'$ is set then pick $\mathbf{t}_n \leftarrow_\$ R_q^k$. Otherwise pick $\mathbf{t} \leftarrow_\$ R_q^k$ and define $\mathbf{t}_n \coloneqq \mathbf{t} - \sum_{j=1}^{n-1} \mathbf{t}_j$.
      - If $\mathrm{HT}_2[\mathbf{t}_n, n]$ has been already set then simulation fails and set bad'_5.
      - Otherwise program the random oracle $\mathrm{HT}_2[\mathbf{t}_n, n] \coloneqq g'_n$ and send out $(\textsc{open}, \mathbf{t}_n, 1, n)$.
3. Upon receiving $(\textsc{open}, \mathbf{t}_j, 1, j)$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_2(\mathbf{t}_j, j) \neq g'_j$ for some j then send out $\textsc{abort}$.
   b. If $alert'$ is set and $\mathsf{H}_2(\mathbf{t}_j, j) = g'_j$ for all j then simulation fails and set bad'_6.
   c. Otherwise set public key shares $L \coloneqq \{\mathbf{t}_1, \dots, \mathbf{t}_n\}$, complete $\textsc{keygen}$ and ignore the future invocation of $\textsc{keygen}$.

Fig. 11. Oracle simulator for the honest party P_n during the key generation phase.

C Security Proof for DS_2

Here we give a full security proof for our two-round protocol DS_2 (see Fig. 6 for the protocol specification). The setup, key generation, and verification are identical to DS_3.

Theorem 2. *Suppose the trapdoor commitment scheme* TCOM *is* ϵ_{td}*-secure, additively homomorphic and has uniform keys. For any probabilistic polynomial-time adversary* $\mathcal{A}$ *that makes a single query to* OGen*,* Q_s *queries to* OSign *and* Q_h *queries to the random oracle* $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$*, the protocol* $\mathsf{DS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ *is* $\mathsf{DS\text{-}UF\text{-}CMA}$ *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions, where* $\beta = 2\sqrt{B_n^2 + \kappa}$.

Proof. Given $\mathcal{A}$ against DS_2 we show that its advantage $\mathbf{Adv}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}_2}(\mathcal{A})$ is negligible by constructing a reduction. Without loss of generality we assume that P_n is an honest party. Our first goal is to construct an algorithm $\mathcal{B}$ around $\mathcal{A}$ that simulate the behaviors of P_n without using honestly generated key pairs. Then we apply the forking lemma to $\mathcal{B}$ and obtain two forgeries with distinct challenges, which allows to construct a solution to SIS. Below we discuss how to do this via several intermediate hybrids.

$\mathbf{G}_0$ **Random Oracle simulation.** We assume that $\mathcal{B}$ receives random samples $h_i \leftarrow_\$ C$ for $i \in [Q_h + Q_s]$ as input. The random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$, $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{l_1}$, $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{l_2}$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{ck}$ are simulated as follows. The table HT_i is initially empty.

$\mathsf{H}_0(x)$ The $\mathcal{B}$ maintains ctr which is initially set to 0. If $\mathrm{HT}_0[x]$ is already set then return $\mathrm{HT}_0[x]$. Otherwise parse x as $(com, \mu, \mathbf{t}')$. If $\mathbf{t}' \neq \mathbf{t}$ set $\mathrm{HT}_0[com, \mu, \mathbf{t}'] \leftarrow_\$ C$. If $\mathbf{t}' = \mathbf{t}$ then proceeds as follows: 1) first make a query $\mathsf{H}_3(\mu, \mathbf{t})$, so that the $\mathsf{H}_3[\mu, \mathbf{t}]$ is immediately set, 2) increment ctr and then set $\mathrm{HT}_0[com, \mu, \mathbf{t}] := h_{ctr}$. Finally, return $\mathrm{HT}_0[com, \mu, \mathbf{t}']$.

$\mathsf{H}_1(x)$ If $\mathrm{HT}_1[x]$ is already set then return $\mathrm{HT}_1[x]$. Otherwise return $\mathrm{HT}_1[x] \leftarrow_\$ \{0,1\}^{l_1}$.

$\mathsf{H}_2(x)$ If $\mathrm{HT}_2[x]$ is already set then return $\mathrm{HT}_2[x]$. Otherwise return $\mathrm{HT}_2[x] \leftarrow_\$ \{0,1\}^{l_2}$.

$\mathsf{H}_3(\mu, \mathbf{t})$ If $\mathrm{HT}_3[\mu, \mathbf{t}]$ is already set then return $\mathrm{HT}_3[\mu, \mathbf{t}]$. Otherwise return $\mathrm{HT}_3[\mu, \mathbf{t}] \leftarrow_\$ S_{ck}$.

Honest Party Oracle simulation. In this game $\mathcal{B}$ behaves exactly like a single honest party in DS_2 to simulate OSign and OGen. Moreover, once the key generation phase is done and the adversary initiated SIGN with some message μ it gets included in $\mathcal{M}$.

Forgery. When $\mathcal{A}$ outputs a forgery $(\mu^*, com, \mathbf{z}, r)$ at the end $\mathcal{B}$ first generates a commitment key $ck \leftarrow \mathsf{H}_3(\mu^*, \mathbf{t})$, derives a challenge $c \leftarrow \mathsf{H}_0(com, \mu^*, \mathbf{t})$ and reconstructs $\mathbf{w} = \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$. Then $\mathcal{B}$ checks $\mu^* \notin \mathcal{M}$ and the verification condition

$$\|\mathbf{z}\|_2 \leq B_n \quad \text{and} \quad \mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1.$$

If $\mathcal{A}$ outputs a successful forgery then $\mathcal{B}$ outputs (i_f, out) where i_f is such that $h_{i_f} = c$ and $out = (\mu^*, com, c, \mathbf{z}, r, ck, \mathbf{t})$. Otherwise $\mathcal{B}$ outputs $(0, \perp)$. Let $\Pr[\mathbf{G}_i]$ denote a probability that $\mathcal{B}$ doesn't output $(0, \perp)$ at the game $\mathbf{G}_i$. Then we have

$$\Pr[\mathbf{G}_0] = \mathbf{Adv}^{\mathsf{DS\text{-}UF\text{-}CMA}}_{\mathsf{DS}_2}(\mathcal{A}).$$

$\mathbf{G}_1$ This game is identical to $\mathbf{G}_0$ except at the following points.

Random Oracle simulation. The simulation of the random oracle H_3 is done as in [DEF+19]. The core idea is to make sure that all sign queries can be responded with trapdoor commitments, which can be equivocated to an arbitrary plaintext later, and that the forgery submitted by $\mathcal{A}$ involves the actual commitment key output by CGen.

$\mathsf{H}_3(\mu, \mathbf{t})$ If $\mathrm{HT}_3[\mu, \mathbf{t}]$ is set then return $\mathrm{HT}_3[\mu, \mathbf{t}]$. Otherwise with probability ϖ compute $(ck, td) \leftarrow \mathsf{TCGen}(cpp)$, store the trapdoor $\mathrm{TDT}[\mu, \mathbf{t}] := td$ and then return $\mathrm{HT}_3[\mu, \mathbf{t}] := ck$. With probability $1 - \varpi$ return $\mathrm{HT}_3[\mu, \mathbf{t}] \leftarrow_\$ S_{ck}$.

Honest Party Oracle simulation. The $\mathcal{B}$ differs from the prior one at the following steps in the signature generation.

1.a. Call $ck \leftarrow \mathsf{H}_3(\mu, \mathbf{t})$. If $\mathrm{TDT}[\mu, \mathbf{t}]$ is not set (i.e., TCGen was not called) then $\mathcal{B}$ aborts by outputting $(0, \perp)$.

1.c. Call $(com_n, st) \leftarrow \mathsf{TCommit}_{ck}(td)$ instead of committing to $\mathbf{w}_n$.

2.c. After computing $\mathbf{z}_n := c\mathbf{s}_n + \mathbf{y}_n$ derive randomness $r_n \leftarrow \mathsf{Eqv}_{ck}(td, st, com_n, \mathbf{w}_n = \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n)$.

Forgery. When $\mathcal{A}$ outputs a successful forgery $(\mu^*, com, \mathbf{z}, r)$ at the end *and if* $\mathrm{TDT}[\mu^*, \mathbf{t}]$ *is not set (i.e.,* CGen *was called)* then $\mathcal{B}$ outputs (i_f, out). Otherwise $\mathcal{B}$ outputs $(0, \perp)$. Recall that TCOM is ϵ_{td}-secure we have

$$\Pr[\mathbf{G}_1] \geq \varpi^{Q_s}(1-\varpi) \cdot (\Pr[\mathbf{G}_0] - Q_s \cdot \epsilon_{\mathrm{td}})$$

because the simulation is successful only if the random oracle H_3 internally uses TCGen for all Q_s signing queries *and* if it uses CGen when μ^* (the message used for forgery) is queried. Note that by setting $\varpi = Q_s/(Q_s+1)$ since $(1/(1+1/Q_s))^{Q_s} \geq 1/e$ for $Q_s \geq 0$ we obtain

$$\Pr[\mathbf{G}_1] \geq \frac{\Pr[\mathbf{G}_0] - Q_s \cdot \epsilon_{\mathrm{td}}}{e(Q_s+1)}.$$

$\mathbf{G}_2$ This game is identical to $\mathbf{G}_1$ except at the following points.

Honest Party Oracle simulation. The $\mathcal{B}$ doesn't honestly generate $\mathbf{z}_n$ anymore and instead simulates the rejection sampling as follows.

2.c. Sample $\mathbf{z}_n \leftarrow\!\!\$\, D_s^{\ell+k}$ and derive randomness $r_n \leftarrow \mathsf{Eqv}_{ck}(td, st, com_n, \mathbf{w}_n = \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n)$.

2.d. With probability $1/M$ announce $(\text{PROCEED}, \mathbf{z}_n, r_n, sid, n)$. Otherwise send out $(\text{RESTART}, \perp, \perp, sid, n)$.

The signature share $\mathbf{z}_n$ simulated this way is indistinguishable from the real one because of special HVZK property of the underlying identification scheme. In other words, we can directly apply the result of Lemma 4. Hence we have

$$|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1]| \leq Q_s \cdot \frac{e^{-t^2/2}}{M}.$$

$\mathbf{G}_3$ Now notice that signing phase doesn't rely on the actual secret key share $\mathbf{s}_n$ anymore. So the next step is to simulate the generation phase without using $\mathbf{s}_n$. This can be done just as in Fig. 11 used for the security proof of three-round protocol, and hence

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2]| \leq \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}}(\mathcal{A}) + \frac{(Q_h+n+1)^2}{2^{l_1+1}} + \frac{Q_h}{q^{k\ell N}} + \frac{n}{2^{l_1}} + \frac{(Q_h+n+1)^2}{2^{l_2+1}} + \frac{Q_h}{q^{kN}} + \frac{n}{2^{l_2}}.$$

Our goal is to embed a challenge commitment key $ck \leftarrow \mathsf{GenCGen}(cpp)$ and an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, which is denoted as $[\mathbf{A}'|\mathbf{I}]$ with $\mathbf{A}' \leftarrow\!\!\$\, R_q^{k\times(\ell+1)}$. As in $\mathbf{G}_3$ the combined public key $(\mathbf{A}, \mathbf{t})$ 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 adversary at all, if $\mathbf{A}'$ is regarded as $\mathbf{A}' = [\mathbf{A}|\mathbf{t}]$. Hence we define the input generator IGen of forking lemma (Lemma 6) so that it outputs the instance $(ck, \mathbf{A}, \mathbf{t})$.

Now we prove the theorem by constructing $\mathcal{B}'$ around $\mathcal{B}$ that either (1) breaks binding of commitment wrt ck, or (2) finds a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ on input $\mathbf{A}' = [\mathbf{A}|\mathbf{t}]$. The $\mathcal{B}'$ invokes the forking algorithm $\mathcal{F}_{\mathcal{B}}$ on input $(ck, \mathbf{A}, \mathbf{t})$ from Lemma 6. Then with probability frk we immediately get two forgeries $out = (\mu^*, com, c, \mathbf{z}, r, ck, \mathbf{t})$ and $out' = (\mu^*, com', c', \mathbf{z}', r', ck', \mathbf{t})$, where frk satisfies

$$\Pr[\mathbf{G}_3] = \mathrm{acc} \leq \frac{Q_h + Q_s}{|C|} + \sqrt{(Q_h+Q_s)\cdot \mathrm{frk}}.$$

By construction of $\mathcal{F}_{\mathcal{B}}$ we have $com = com'$ and $c \neq c'$. Due to the simulation of H_0 we guarantee that $\mathsf{H}_3(\mu^*, \mathbf{t}) = ck = ck'$ since it should have been invoked right before the fork. Since both forgeries are verified we have

$$\|\mathbf{z}\|_2 \leq B_n \wedge \|\mathbf{z}'\|_2 \leq B_n \wedge \mathsf{Open}_{ck}(com, r, \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}) = \mathsf{Open}_{ck}(com, r', \bar{\mathbf{A}}\mathbf{z}' - c'\mathbf{t}) = 1$$

where $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}]$. Thanks to the simulation of H_3 it is guaranteed that ck follows the uniform distribution over S_{ck} which is perfectly indistinguishable from honestly generated $ck \leftarrow \mathsf{CGen}(cpp)$ (since the keys are uniform). Hence the probability that $\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} \neq \bar{\mathbf{A}}\mathbf{z}' - c'\mathbf{t}$ is at most ϵ_{bind}. Now we assume $\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} = \bar{\mathbf{A}}\mathbf{z}' - c'\mathbf{t}$. Rearranging it leads to

$$[\mathbf{A}|\mathbf{I}|\mathbf{t}]\begin{bmatrix}\mathbf{z}-\mathbf{z}' \\ c'-c\end{bmatrix} = \mathbf{0}.$$

Hence we have found a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, where $\beta = \sqrt{(2B_n)^2 + 4\kappa}$, since $\|\mathbf{z}-\mathbf{z}'\|_2 \leq 2B_n$ and $\|c-c'\|_2 \leq \sqrt{4\kappa}$. Putting two cases together, we get

$$\mathrm{frk} \leq \epsilon_{\mathrm{bind}} + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}$$

D Two-round Multi-signature in the Plain Public Key Model

In this section we describe our two-round multi-signature scheme $\mathsf{MS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$. The Setup works just like the one for DS_2 and DS_3, but it additionally outputs a matrix $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}]$ (if the generation of $\bar{\mathbf{A}}$ has to be done in a distributed way then one can invoke a matrix generation protocol in Fig. 3 instead). Then Gen algorithm is the same as Algorithm 1, except that it takes $\bar{\mathbf{A}}$ as input and outputs $sk = \mathbf{s}$ and $pk = \mathbf{t}$. The signing protocol and verification are described in Figs. 12 and 13. The protocol $\mathsf{MS}_2.\mathsf{Sign}$ closely resembles $\mathsf{DS}_2.\mathsf{Sign}$. The main difference is that signature shares are constructed from per-user challenges, instead of a single common challenge for all co-signers (just as Bellare–Neven [BN06] or Bagherzandi et al. [BCJ08] did). Therefore, the random oracle simulation below is slightly more involved than in Theorem 2.

Theorem 3. *Suppose the trapdoor commitment scheme is ϵ_{td}-secure and additively homomorphic. For any probabilistic polynomial-time adversary $\mathcal{A}$ that makes Q_s queries to* OSign *and Q_h queries to the random oracle, the protocol* $\mathsf{MS}_2 = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{Vf})$ *is* $\mathsf{MS\text{-}UF\text{-}CMA}$ *secure under* $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ *and* $\mathsf{MLWE}_{q,k,\ell,\eta}$ *assumptions, where $\beta = 2\sqrt{B_n^2 + \kappa}$.*

Proof. Given $\mathcal{A}$ against MS_2 we show that its advantage $\mathbf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}_2}(\mathcal{A})$ is negligible by constructing a reduction. Without loss of generality we assume that P_n is an honest party. Our first goal is to construct an algorithm $\mathcal{B}$ around $\mathcal{A}$ that simulate the behaviors of P_n without using honestly generated key pairs. Then we apply the forking lemma to $\mathcal{B}$ and obtain two forgeries with distinct challenges with respect to $\mathbf{t}_n$, which allows to construct a solution to SIS. Below we discuss how to do this via several intermediate hybrids.

$\mathbf{G}_0$ **Random Oracle simulation.** We assume that $\mathcal{B}$ receives randomly picked $h_i \in C$ for $i \in [Q_h + Q_s]$ as input. The random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{ck}$ are simulated as follows. The table HT_i is initially empty.

$\mathsf{H}_0(x)$ The $\mathcal{B}$ maintains ctr which is initially set to 0. If $\mathrm{HT}_0[x]$ is already set then return $\mathrm{HT}_0[x]$. Otherwise parse x as $(com, \mu, \mathbf{t}', L)$. If either $\mathbf{t}_n$ or $\mathbf{t}'$ is not present in set L then set $\mathrm{HT}_0[com, \mu, \mathbf{t}', L] \leftarrow\!\!\$\, C$. If both $\mathbf{t}_n$ and $\mathbf{t}'$ are in set L then proceeds as follows: 1) first make a query $\mathsf{H}_3(\mu, L)$, so that the $\mathsf{H}_3[\mu, L]$ is immediately set, 2) for each $\mathbf{t}_j \in L$ such that $\mathbf{t}_j \neq \mathbf{t}_n$, set $\mathrm{HT}_0[com, \mu, \mathbf{t}_j, L] \leftarrow\!\!\$\, C$, and 3) increment ctr and set $\mathrm{HT}_0[com, \mu, \mathbf{t}_n, L] := h_{ctr}$. Finally, return $\mathrm{HT}_0[com, \mu, \mathbf{t}', L]$.

$\mathsf{H}_3(\mu, L)$ If $\mathrm{HT}_3[\mu, L]$ is already set then return $\mathrm{HT}_3[\mu, L]$. Otherwise return $\mathrm{HT}_3[\mu, L] \leftarrow\!\!\$\, S_{ck}$.

Honest Party Oracle simulation. In this game $\mathcal{B}$ behaves exactly like a single honest party in MS_2 to simulate OSign. Once the key generation is done and the adversary initiated SIGN with some message μ and set L, if $\mathbf{t}_n \notin L$ then $\mathcal{B}$ returns $\perp$. Otherwise μ gets included in $\mathcal{M}$ and $\mathcal{B}$ invokes instructions of $\mathsf{Sign}(\mathbf{s}_n, \mu, L)$.

Forgery. When $\mathcal{A}$ outputs a forgery $(\mu^*, com, \mathbf{z}, r, L^*)$ at the end $\mathcal{B}$ first generates a commitment key $ck \leftarrow \mathsf{H}_3(\mu^*, L^*)$. Suppose the index set $J = \{j : \mathbf{t}_j \in L^*\}$, then $\mathcal{B}$ derives $c_j = \mathsf{H}_0(com, \mu, \mathbf{t}_j, L)$ for each $j \in J$ and reconstructs $\mathbf{w} = \bar{\mathbf{A}}\mathbf{z} - \sum_j c_j \mathbf{t}_j$. Then $\mathcal{B}$ checks $(\mu^*, L^*) \notin \mathcal{M}, \mathbf{t}_n \in L^*$ and the verification condition

$$\|\mathbf{z}\|_2 \leq B_n \quad \text{and} \quad \mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1.$$

If $\mathcal{A}$ outputs a successful forgery then $\mathcal{B}$ outputs (i_f, out), where i_f is such that $h_{i_f} = c_n$ and $out = (\mu^*, com, \{c_j\}_{j \in J}, \mathbf{z}, r, ck, \{\mathbf{t}_j\}_{j \in J})$. Otherwise $\mathcal{B}$ outputs $(0, \perp)$. Let $\Pr[\mathbf{G}_i]$ denote a probability that $\mathcal{B}$ doesn't output $(0, \perp)$ at the game $\mathbf{G}_i$. Then we have

$$\Pr[\mathbf{G}_0] = \mathbf{Adv}^{\mathsf{MS\text{-}UF\text{-}CMA}}_{\mathsf{MS}_2}(\mathcal{A}).$$

$\mathbf{G}_1$ This game is identical to $\mathbf{G}_0$ except at the following points.

Random Oracle simulation. The simulation of the random oracle H_3 is done as follows.

$\mathsf{H}_3(\mu, L)$ If $\mathrm{HT}_3[\mu, L]$ is set then return $\mathrm{HT}_3[\mu, L]$. Otherwise with probability ϖ compute $(ck, td) \leftarrow \mathsf{TCGen}(cpp)$, store the trapdoor $\mathrm{TDT}[\mu, L] := td$ and then return $\mathrm{HT}_3[\mu, L] := ck$. With probability $1 - \varpi$ return $\mathrm{HT}_3[\mu, L] \leftarrow\!\!\$\, S_{ck}$.

Honest Party Oracle simulation. The $\mathcal{B}$ differs from the prior one at the following steps in the signature generation.

Protocol $\mathsf{MS}_2.\mathsf{Sign}$

The protocol is parameterized by public parameters described in Table 1 and relies on the random oracles $\mathsf{H}_0 : \{0,1\}^* \to C$ and $\mathsf{H}_3 : \{0,1\}^* \to S_{ck}$. The protocol assumes that $\mathsf{MS}_2.\mathsf{Gen}$ has been previously invoked and works on $sk_n = \mathbf{s}_n$ and $pk_n = \mathbf{t}_n$ as input.

Signature Generation:

1. Upon receiving $(\textsc{sign}, \mu, sid, L)$, if $\mathbf{t}_n \in L$ then compute the first message as follows.
   a. Derive a per-message commitment key $ck \leftarrow \mathsf{H}_3(\mu, L)$.
   b. Sample $\mathbf{y}_n \leftarrow\!\!\$\ D_s^{\ell+k}$ and compute $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{y}_n$.
   c. Compute $com_n \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_n; r_n)$ with $r_n \leftarrow\!\!\$\ D(S_r)$.
   d. Send out $(\textsc{open}, com_n, sid, n)$.
2. Upon receiving $(\textsc{open}, com_j, sid, j)$ for all P_j such that $\mathbf{t}_j \in L$ compute the signature share as follows.
   a. Set $com := \sum_j com_j$.
   b. Derive a per-user challenge $c_n \leftarrow \mathsf{H}_0(com, \mu, \mathbf{t}_n, L)$.
   c. Computes a signature share $\mathbf{z}_n := c_n\mathbf{s}_n + \mathbf{y}_n$.
   d. Run the rejection sampling on input $(c_n\mathbf{s}_n, \mathbf{z}_n)$, i.e., with probability
   $$\min\left(1, D_s^{\ell+k}(\mathbf{z}_n)/(M \cdot D_{c_n\mathbf{s}_n,s}^{\ell+k}(\mathbf{z}_n))\right)$$
   announce $(\textsc{proceed}, \mathbf{z}_n, r_n, sid, n)$; otherwise send out $(\textsc{restart}, \perp, \perp, sid, n)$.
3. Upon receiving $(\textsc{restart}, \perp, \perp, sid, j)$ from some party go to 1. Otherwise upon receiving $(\textsc{proceed}, \mathbf{z}_j, r_j, sid, j)$ for all j such that $\mathbf{t}_j \in L$ compute the combined signature as follows
   a. For each j derive a per-user challenge $c_j \leftarrow \mathsf{H}_0(com, \mu, \mathbf{t}_j, L)$, reconstruct $\mathbf{w}_j := \bar{\mathbf{A}}\mathbf{z}_j - c_j\mathbf{t}_j$ and validate the signature share:
   $$\|\mathbf{z}_j\|_2 \le B \quad \text{and} \quad \mathsf{Open}_{ck}(com_j, r_j, \mathbf{w}_j) = 1.$$
   If the check fails for some j then send out $\textsc{abort}$.
   b. Compute $\mathbf{z} := \sum_j \mathbf{z}_j$ and $r := \sum_j r_j$.
   c. Output $(com, \mathbf{z}, r)$ as a signature.

Fig. 12. Two-round multi-signature protocol secure in the plain public key model.

Algorithm $\mathsf{MS}_2.\mathsf{Vf}$

Upon receiving a message μ, signature $(com, \mathbf{z}, r)$, and a set of public keys L, generate a commitment key $ck \leftarrow \mathsf{H}_3(\mu, L)$, for each j such that $\mathbf{t}_j \in L$ derive a per-user challenge $c_j \leftarrow \mathsf{H}_0(com, \mu, \mathbf{t}_j, L)$ and reconstruct $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - \sum_j c_j\mathbf{t}_j$. Then accept if the following holds:

$$\|\mathbf{z}\|_2 \le B_n \quad \text{and} \quad \mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1.$$

Fig. 13. Verification algorithm for multi-signature protocol.

1.a. Call $ck \leftarrow \mathsf{H}_3(\mu, L)$. If $\mathrm{TDT}[\mu, L]$ is not set (i.e., TCGen was not called) then $\mathcal{B}$ aborts by outputting $(0, \perp)$.

1.c. Call $(com_n, st) \leftarrow \mathsf{TCommit}_{ck}(td)$ instead of committing to $\mathbf{w}_n$.

2.c. After computing $\mathbf{z}_n = c_n\mathbf{s}_n + \mathbf{y}_n$ derive randomness $r_n \leftarrow \mathsf{Eqv}_{ck}(td, st, com_n, \mathbf{w}_n = \bar{\mathbf{A}}\mathbf{z}_n - c_n\mathbf{t}_n)$.

Forgery. When $\mathcal{A}$ outputs a successful forgery $(\mu^*, com, \mathbf{z}, r, L^*)$ at the end *and if* $\mathrm{TDT}[\mu^*, L^*]$ *is not set (i.e.,* CGen *was called)* then $\mathcal{B}$ outputs (i_{f}, out). Otherwise $\mathcal{B}$ outputs $(0, \perp)$. Recall that TCOM is ϵ_{td}-secure we have

$$\Pr[\mathbf{G}_1] \geq \varpi^{Q_s}(1 - \varpi) \cdot (\Pr[\mathbf{G}_0] - Q_s \cdot \epsilon_{td})$$

because the simulation is successful only if the random oracle H_3 internally uses TCGen for all Q_s signing queries *and* if it uses CGen when μ^* (the message used for forgery) is queried. Note that by setting $\varpi = Q_s/(Q_s + 1)$ since $(1/(1 + 1/Q_s))^{Q_s} \geq 1/e$ for $Q_s \geq 0$ we obtain

$$\Pr[\mathbf{G}_1] \geq \frac{\Pr[\mathbf{G}_0] - Q_s \cdot \epsilon_{\mathrm{td}}}{e(Q_s + 1)}.$$

$\mathbf{G}_2$ This game is identical to $\mathbf{G}_1$ except at the following points.

Honest Party Oracle simulation. The $\mathcal{B}$ doesn't honestly generate $\mathbf{z}_n$ anymore and instead simulates the rejection sampling as follows.

2.c. Sample $\mathbf{z}_n \leftarrow\!\!\$\, D_s^{\ell+k}$ and derive randomness $r_n \leftarrow \mathsf{Eqv}_{ck}(td, st, com_n, \mathbf{w}_n = \bar{\mathbf{A}}\mathbf{z}_n - c_n\mathbf{t}_n)$.

2.d. With probability $1/M$ announce $(\textsc{proceed}, \mathbf{z}_n, r_n, sid, n)$. Otherwise send out $(\textsc{restart}, \perp, \perp, sid, n)$.

The signature share $\mathbf{z}_n$ simulated this way is indistinguishable from the real one because of special HVZK property of the underlying identification scheme. In other words, we can directly apply the result of Lemma 4. Hence we have

$$|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1]| \leq Q_s \cdot \frac{e^{-t^2/2}}{M}.$$

$\mathbf{G}_3$ Now notice that signing phase doesn't rely on the actual secret key share $\mathbf{s}_n$ anymore. So the next step is to simulate the generation of $\mathbf{t}_n$ without using $\mathbf{s}_n$. In this game $\mathcal{A}$ receives the challenge public key $\mathbf{t}_n$ which is sampled from R_q^k uniformly at random. Due to the hardness of $\mathsf{MLWE}_{q,k,\ell,\eta}$ the adversary cannot distinguish simulated $\mathbf{t}_n$ from the real one, and hence we have

$$|\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2]| \leq \mathbf{Adv}_{\mathsf{MLWE}_{q,k,\ell,\eta}}(\mathcal{A}).$$

Our goal is to embed a challenge commitment key $ck \leftarrow \mathsf{GenCGen}(cpp)$ and an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, which is denoted as $[\mathbf{A}'|\mathbf{I}]$ with $\mathbf{A}' \leftarrow\!\!\$\, R_q^{k\times(\ell+1)}$. As in $\mathbf{G}_3$ the matrix and public key $(\mathbf{A}, \mathbf{t}_n)$ 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 adversary at all, if $\mathbf{A}'$ is regarded as $\mathbf{A}' = [\mathbf{A}|\mathbf{t}_n]$. Hence we define the input generator IGen of forking lemma (Lemma 6) so that it outputs the instance $(ck, \mathbf{A}, \mathbf{t}_n)$.

Now we prove the theorem by constructing $\mathcal{B}'$ around $\mathcal{B}$ that either (1) breaks binding of commitment wrt ck, or (2) finds a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ on input $\mathbf{A}' = [\mathbf{A}|\mathbf{t}_n]$. The $\mathcal{B}'$ invokes the forking algorithm $\mathcal{F}_{\mathcal{B}}$ on input $(ck, \mathbf{A}, \mathbf{t}_n)$ from Lemma 6. Then with probability frk we immediately get two forgeries $out = (\mu^*, com, \{c_j\}_{j\in J}, \mathbf{z}, r, ck, \{\mathbf{t}_j\}_{j\in J})$ and $out' = (\mu^*, com', \{c'_j\}_{j\in J'}, \mathbf{z}', r', ck', \{\mathbf{t}'_j\}_{j\in J'})$, where frk satisfies

$$\Pr[\mathbf{G}_3] = \mathrm{acc} \leq \frac{Q_h + Q_s}{|C|} + \sqrt{(Q_h + Q_s) \cdot \mathrm{frk}}.$$

Due to the way we simulated the random oracle H_0, we have $com = com'$, $\mathbf{t}_j = \mathbf{t}'_j$ for $j \in J = J'$, $c_j = c'_j$ for $j \in J\backslash\{n\}$, and $ck = ck'$ since all these values should have been recorded in HT_0 or HT_3 right before the fork. Moreover, by construction of $\mathcal{F}_{\mathcal{B}}$ it holds that $c_n \neq c'_n$. Since both forgeries are verified we have

$$\|\mathbf{z}\|_2 \leq B_n \wedge \|\mathbf{z}'\|_2 \leq B_n \wedge 1 = \mathsf{Open}_{ck}(com, r, \bar{\mathbf{A}}\mathbf{z} - \sum_j c_j\mathbf{t}_j) = \mathsf{Open}_{ck}(com, r', \bar{\mathbf{A}}\mathbf{z}' - \sum_j c'_j\mathbf{t}_j)$$

where $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}]$.

Thanks to the simulation of H_3 it is guaranteed that ck follows the uniform distribution over S_{ck} which is perfectly indistinguishable from honestly generated $ck \leftarrow \mathsf{CGen}(cpp)$ (since the keys are uniform). Hence the probability that $\bar{\mathbf{A}}\mathbf{z} - \sum_{j \in J} c_j \mathbf{t}_j \neq \bar{\mathbf{A}}\mathbf{z}' - \sum_{j \in J} c'_j \mathbf{t}_j$ is at most ϵ_{bind}. Now we assume $\bar{\mathbf{A}}\mathbf{z} - \sum_{j \in J} c_j \mathbf{t}_j = \bar{\mathbf{A}}\mathbf{z}' - \sum_{j \in J} c'_j \mathbf{t}_j$. Recall that $c_j = c'_j$ for $j \neq n$, we have $\bar{\mathbf{A}}\mathbf{z} - c_n \mathbf{t}_n = \bar{\mathbf{A}}\mathbf{z}' - c'_n \mathbf{t}_n$. Rearranging it leads to

$$[\mathbf{A}|\mathbf{I}|\mathbf{t}_n] \begin{bmatrix} \mathbf{z} - \mathbf{z}' \\ c'_n - c_n \end{bmatrix} = \mathbf{0}.$$

Hence we have found a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, where $\beta = \sqrt{(2B_n)^2 + 4\kappa}$, since $\|\mathbf{z} - \mathbf{z}'\|_2 \leq 2B_n$ and $\|c_n - c'_n\|_2 \leq \sqrt{4\kappa}$. Putting two cases together, we get

$$\text{frk} \leq \epsilon_{\text{bind}} + \mathbf{Adv}_{\mathsf{MSIS}_{q,k,\ell+1,\beta}}$$

E General Forking Lemma

We restate the general forking lemma from [BN06].

Lemma 6 (General Forking Lemma). *Let Q be a number of queries and C be a set of size $|C| > 2$. Let $\mathcal{B}$ be a randomized algorithm that on input x, $h_1, \ldots, h_Q$ returns an index $i \in [0, Q]$ and a side output out. Let* IGen *be a randomized algorithm that we call the input generator. Let $\mathcal{F}_\mathcal{B}$ be a forking algorithm that works as in Fig. 14 given x as input and given black-box access to $\mathcal{B}$. Suppose the following probabilities.*

$$\text{acc} := \Pr[i \neq 0 : x \leftarrow \mathsf{IGen}(1^\lambda); h_1, \ldots, h_Q \leftarrow_\$ C; (i, out) \leftarrow \mathcal{B}(x, h_1, \ldots, h_Q)]$$
$$\text{frk} := \Pr[(out, out') \neq (\bot, \bot) : x \leftarrow \mathsf{IGen}(1^\lambda); (out, out') \leftarrow \mathcal{F}_\mathcal{B}(x)]$$

Then

$$\text{frk} \geq \text{acc} \cdot \left(\frac{\text{acc}}{Q} - \frac{1}{|C|} \right).$$

Alternatively,

$$\text{acc} \leq \frac{Q}{|C|} + \sqrt{Q \cdot \text{frk}}.$$

Algorithm $\mathcal{F}_\mathcal{B}(x)$

```
Upon receiving $x$
  1. Pick a random coin $\rho$ for $\mathcal{B}$.
  2. Generate $h_1, \ldots, h_Q \leftarrow_\$ C$.
  3. $(i, out) \leftarrow \mathcal{B}(x, h_1, \ldots, h_Q; \rho)$.
  4. If $i = 0$ then return $(\bot, \bot)$.
  5. Regenerate $h'_i, \ldots, h'_Q \leftarrow_\$ C$.
  6. $(i', out') \leftarrow \mathcal{B}(x, h_1, \ldots, h_{i-1}, h'_i, \ldots, h'_Q; \rho)$.
  7. If $i = i'$ and $h_i \neq h'_i$ then return $(out, out')$
  8. Else return $(\bot, \bot)$.
```

Fig. 14. The forking algorithm $\mathcal{F}_\mathcal{B}$

F Potential Wagner-like Attack on a Variant with Fixed Commitment Key

Below we sketch a variant of the concurrent attack originally described by Drijvers et al. [DEF+19]. The original attack was against two-round Schnorr multi-signatures including BCJ scheme [BCJ08], but due to the very similar structure of FSwA-based lattice signatures an attack would become feasible against our two-round protocols (albeit with sub-exponential computational costs) if the commitment key was fixed during the setup phase. This motivates us to derive a per-message commitment key during the signing protocol as we presented in Figs. 6 and 12.

For simplicity we describe an attack in the two-party setting. Let $\mathbf{s}$ and $\mathbf{s}'$ be the key shares of honest party and adversary respectively and let $\mathbf{t} = \bar{\mathbf{A}}(\mathbf{s}+\mathbf{s}')$ be the combined public key. The adversary initiates k concurrent signing sessions on the same message μ. Then for each session $i \in [k]$, the honest party submits

$$com_i \leftarrow \mathsf{Commit}_{ck}(\bar{\mathbf{A}}\mathbf{y}_i; r_i).$$

Here the adversary does not immediately send back its own commitment share. Instead let $com^* = com_1 + \ldots + com_k$, by only interacting with the random oracle H_0 the adversary tries to find a message μ^* and $com'_1, \ldots, com'_k \in S_{com}$ such that the following holds.

$$\mathsf{H}_0(com^*, \mu^*, \mathbf{t}) = \mathsf{H}_0(com_1 + com'_1, \mu, \mathbf{t}) + \ldots + \mathsf{H}_0(com_k + com'_k, \mu, \mathbf{t})$$

Because the image C of the random oracle consists of small and sparse vectors in $\mathbb{Z}^N$, finding such inputs amounts to solving a vectorial variant of Wagner's generalized birthday problem (GBP) [Wag02, HJ10] for $k+1$ list sums, when k is chosen such that $k+1$ is a power of two. Then the adversary resumes the sessions by sending back such com'_i for $i \in [k]$. The honest signer for each session returns its signature share together with commitment opening r_i

$$\mathbf{z}_i = \mathbf{y}_i + c_i\mathbf{s}$$

where $c_i = \mathsf{H}_0(com_i + com'_i, \mu, \mathbf{t})$. Now let

$$\begin{aligned} com^* &= com_1 + \ldots + com_k \\ \mathbf{z}^* &= \mathbf{z}_1 + \ldots + \mathbf{z}_k + c^*\mathbf{s}' = \mathbf{y}_1 + \ldots + \mathbf{y}_k + c^*(\mathbf{s}+\mathbf{s}') \\ r^* &= r_1 + \ldots + r_k \end{aligned}$$

be a forgery on the μ^*. Thanks to the collision found by a GBP solver it holds that $\mathsf{H}_0(com^*, \mu^*, \mathbf{t}) = c^*$. Due to the homomorphism and correctness of the commitment opening it holds that $\mathsf{Open}_{ck}(com^*, r^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t}^*) = 1$. Hence the entire verification passes. The adversary would require extra care about the norm of $\mathbf{z}^*$ by bounding the number of sessions k, since the small $\mathbf{z}^*$ is part of the verification check. For this reason there should be some tradeoffs for k, since the larger the k is, the easier the GBP becomes.

If the protocol derives a per-message commitment key via random oracle $\mathsf{H}_3 : \{0,1\}^* \to S_{ck}$ as our protocols (as well as mBCJ) do, the attack becomes nontrivial; now the tuple $(com^*, r^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t})$ has to be verified with respect to the message-dependent key $ck^* \leftarrow \mathsf{H}_3(\mu^*, \mathbf{t})$, which of course shouldn't collide with $ck \leftarrow \mathsf{H}_3(\mu, \mathbf{t})$ thanks to the random oracle.