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, Aarhus, Denmark
{ivan,orlandi,takahashi}@cs.au.dk
[2] NTT Corporation, Tokyo, Japan
mehdi.tibouchi.br@hco.ntt.co.jp

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 or 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-signatureschemes 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 only come with incomplete security proofs.

In this paper, we construct several lattice-based distributed signing protocols with low round complexity followingthe Fiat–Shamir with Aborts (FSwA) paradigm of Lyubashevsky (Asiacrypt 2009). Our protocols can be seen as distributed variants of the fast Dilithium-G signature scheme and the full security proof can be made assuming the hardness of module SIS and LWE problems. A key step to achieving security (unexplained 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 homomorphic commitments.

Exploiting the similarities between FSwA and Schnorr-style signatures, our approach makes the most of observations from recent advancements in the discrete log setting, such as Drijvers et al.'s seminal work on two-round multi-signatures (S&P 2019). In particular, we observe that the use of commitment not only resolves the subtle issue with aborts, but also makes it possible to realize secure *two-round n-out-of-n distributed signing* and *multi-signature in the plain public key model*, by equipping the commitment with a trapdoor feature. The construction of suitable trapdoor commitment from lattices is a side contribution of this paper.

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

J. A. Garay (Ed.): PKC 2021, LNCS 12710, pp. 99–130, 2021.
https://doi.org/10.1007/978-3-030-75245-3_5

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 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-out-of-2 variant [19,32,59] and in the more general t-out-of-n case [17,20,27,29,33,44,46–48,60]. 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. [10] proposed a threshold protocol to generate Gentry–Peikert–Vaikuntanathan signature [50]; Boneh et al. [15] 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.

Fiat–Shamir with Aborts. Neither of the above previous papers investigated signatures following the *Fiat–Shamir with Aborts (FSwA)* paradigm due to Lyubashevsky [62,63], which was specifically designed for lattice-based signatures and is nowadays one of the most efficient and popular approaches 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.

Examples of single-user schemes based on FSwA include Dilithium [65] and qTESLA [14], which are round-3 and round-2 candidates of the NIST Post-Quantum Cryptography Standardization process. Cozzo and Smart [26] 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 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* [9]), (2) the group of signing parties is usually not fixed, and each party can initiate the signing protocol with a dynamically chosen set of parties associated with $L = \{pk_1, \ldots, pk_n\}$, 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 [69] is supported).

Schnorr vs FSwA. There is a long line of research that starts from Schnorr's signature scheme [80] and follows the standard Fiat–Shamir paradigm to build distributed signatures [2,49,57,75,81] and multi-signatures [3,9,68–70,73,74,82]. In particular, Drijvers et al. [35] recently discovered a flaw of the existing *two-round* Schnorr-based multi-signatures, with a novel concurrent attack relying on the generalized birthday algorithm of Wagner [86]. They accordingly proposed mBCJ scheme, a provably secure variant of Bagherzhandi et al.'s BCJ scheme [3].

Unlike distributed n-out-of-n signatures, several *three* or *four-round* multi-signatures based on FSwA are already present in the literature. Bansarkhani and Sturm [4] extended Güneysu–Lyubashevsky–Pöppelmann (GLP) [53] signature and proposed the first multi-signature following the FSwA paradigm, which was recently followed by multiple similar variants [42,43,67,83,85]. Relying on the syntactic similarities between Schnorr and FSwA-style signatures, these protocols essentially borrow the ideas of Schnorr-based counterparts; for instance, [4] can be considered as a direct adaptation of Bellare and Neven's three-round Schnorr-like multi-signature [9]. However, as explained below, the security proofs of all these protocols are either incomplete or relying on a non-standard hardness assumption, 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, while making the most of useful observations from the discrete log setting.

Issue with "aborts". We first observe that there is an inherent issue when constructing distributed FSwA signatures. Just like earlier constructions in the

discrete log setting [9,75] previous FSwA multi-signatures 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. An important issue with the FSwA approach is that, currently there is no known general way to prove the underlying Σ-protocol zero-knowledge in case of aborts [11, §3.2],[41, §4],[5,6],[64, p. 26]. As a result, the signer should not reveal any of the aborted executions since otherwise the scheme cannot be proved secure. This issue is not serious in a single-user scheme, since the Σ-protocol is made non-interactive in the random oracle model anyway and there is no reason why the signer would reveal aborted executions.

In an interactive setting, the standard approach to circumvent the issue is to send a commitment to the first Σ-protocol message and only reveal it if the rejection sampling is successful. However, the previous FSwA multi-signatures skipped this subtle step. Thus the concurrent work by Fukumitsu and Hasegawa [43] (who constructed a FSwA-style multi-signature proven secure in QROM) had to rely on an additional non-standard assumption (which they call "rejected LWE"), while publicly available security proofs of other similar constructions [4,42,67,83,85] do not explain how to simulate the aborted executions. 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, it is paramount to strive to find protocols which can be proven secure relying on well-established computational hardness assumptions like LWE and SIS.

1.1 Contributions

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. Our constructions can be seen as distributed variants of the fast Dilithium-G signature scheme [37]. As a first step, we circumvent the aborts issue mentioned above by utilizing Baum et al.'s *additively homomorphic* commitment scheme [7], which is currently the most efficient construction based on lattices and relies on the hardness of Module-LWE and Module-SIS problems. This results in a provably secure (in the classical random oracle model), three-round n-out-of-n signature protocol DS_3[1].

[1] It is still an open question whether the aborts issue can instead be resolved by careful parameter choice, allowing to simulate the rejected transcripts without any additional assumptions. But we are aware of on-going work in this direction. If the question is answered in the affirmative our three-round protocol could be proven secure even without a commitment. However, the use of homomorphic commitment is crucial for constructing our new two-round protocols, which is our main contribution.

First Two-Round Protocols. Previous FSwA-based multi-signatures required at least three rounds of interaction. On the other hand, as most recent discrete log-based solutions indicate [35,57,73,74], two rounds is a natural goal because this clearly seems to be minimal for a distributed signature protocol based on the Fiat-Shamir paradigm: we first need to determine what should be hashed in order to get the challenge in the underlying Σ-protocol. This must (for security) include randomness from several players and hence requires at least one round of interaction. After this we need to determine the prover's answer in the Σ-protocol. This cannot be computed until the challenge is known and must (for security) require contributions from several players, and we therefore need at least one more round.

In this paper, we show that the application of homomorphic commitment not only resolves the issue with aborts, but also makes it possible to reduce the round complexity to two rounds. We do this by adding a *trapdoor* feature to the commitment scheme (a separate contribution that we discuss in more detail below). This results in a two-round, n-out-of-n signature protocol DS_2 presented in Sect. 3. With a slight modification this n-out-of-n protocol can be also turned into a two-round multi-signature scheme in the plain public key model. We describe a multi-signature variant MS_2 in the full version of this paper [30].

Our main two-round result highlights several important similarities and differences which emerge when translating a discrete log-based protocol to lattice-based one. The approaches taken in our two-round protocols are highly inspired by mBCJ discrete log-based multi-signature by Drijvers et al. [35] In particular, we observe that it is crucial for two-round protocols to use message-dependent commitment keys (as in mBCJ) instead of a single fixed key for all signing attempts (as in the original BCJ [3]), because otherwise the proof doesn't go through. Drijvers et al. only presented a full security proof for the 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 mentioned above, we turn Baum et al.'s scheme into a trapdoor commitment in Sect. 4, 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 [71] 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 [18,38,51,52,58] our technique may be of independent interest.

1.2 Technical Overview

Our protocols are based on Dilithium signature scheme, which works over rings $R = \mathbb{Z}[X]/(f(X))$ and $R_q = \mathbb{Z}_q[X]/(f(X))$ defined with an appropriate irreducible polynomial $f(X)$ (see preliminaries for more formal details). Here we go over the core ideas of our construction by considering simple 2-out-of-2 signing protocols. The protocols below can be generalized to an n-party setting in a straightforward manner. We assume that each party P_j for $j = 1, 2$ owns a secret signing key share $\mathbf{s}_j \in R^{\ell+k}$ which has small coefficients, and a public random matrix $\bar{\mathbf{A}} = [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$. The joint public verification key is defined as $pk = (\mathbf{A}, \mathbf{t})$, where $\mathbf{t} = \bar{\mathbf{A}}(\mathbf{s}_1 + \mathbf{s}_2) \bmod q$. In the actual protocols pk also needs to be generated in a distributed way, but here we omit the key generation phase for brevity's sake.

Naive Approach. We first present a naive (insecure) way to construct a 2-party signing protocol from FSwA. If the reader is familiar with CoSi Schnorr multi-signature [82] this construction is essentially its lattice-based, 2-out-of-2 variant. In this protocol the parties P_j for $j = 1, 2$ involved in signing the message μ work as follows.

1. P_j samples a randomness $\mathbf{y}_j$ from some distribution $D^{\ell+k}$ defined over $R^{\ell+k}$ (which is typically the uniform distribution over a small range or discrete Gaussian), and then sends out the first message of FSwA $\mathbf{w}_j = \bar{\mathbf{A}}\mathbf{y}_j \bmod q$.
2. P_j locally derives a joint challenge $c \leftarrow \mathsf{H}(\mathbf{w}_1 + \mathbf{w}_2, \mu, pk)$ and performs the rejection sampling $\mathsf{RejSamp}(c\mathbf{s}_j, \mathbf{z}_j)$ with $\mathbf{z}_j = c\mathbf{s}_j + \mathbf{y}_j$; if the result of $\mathsf{RejSamp}(c\mathbf{s}_j, \mathbf{z}_j)$ is "reject" then P_j sets $\mathbf{z}_j := \perp$. After exchanging $\mathbf{z}_j$'s if $\mathbf{z}_1 = \perp$ or $\mathbf{z}_2 = \perp$ (i.e., either of the parties aborts), then the protocol restarts from the step 1.
3. Each party outputs $(\mathbf{w}, \mathbf{z}) := (\mathbf{w}_1 + \mathbf{w}_2, \mathbf{z}_1 + \mathbf{z}_2)$ as a signature on μ.

Note that the rejection sampling step is needed to make the distribution of $\mathbf{z}_j$ independent of a secret $\mathbf{s}_j$. The verification algorithm checks that the norm of $\mathbf{z}$ is small, and that $\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} = \mathbf{w} \pmod q$ holds, where the challenge is recomputed as $c \leftarrow \mathsf{H}(\mathbf{w}, \mu, pk)$. One can easily check that the signature generated as above satisfies correctness, thanks to the linearity of the SIS function $f_{\bar{\mathbf{A}}}(\mathbf{x}) = \bar{\mathbf{A}}\mathbf{x} \bmod q$. However, we observe that an attempt to give a security proof fails due to two problems. Suppose the first party $\widetilde{P}_1$ is corrupt and let us try to simulate the values returned by honest P_2, whenever queried by the adversary.

First, since the protocol reveals $\mathbf{w}_2$ whether P_2 aborts or not, the joint distribution of rejected transcript $(\mathbf{w}_2, c, \perp)$ has to be simulated. As mentioned earlier, there is no known way to simulate it; in fact, the honest verifier zero knowledge (HVZK) of FSwA is only proven for "non-aborting" cases in the original work by Lyubashevsky [62–64] and its successors. Note that the obvious fix where players hash 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 obtaining the challenge c 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.

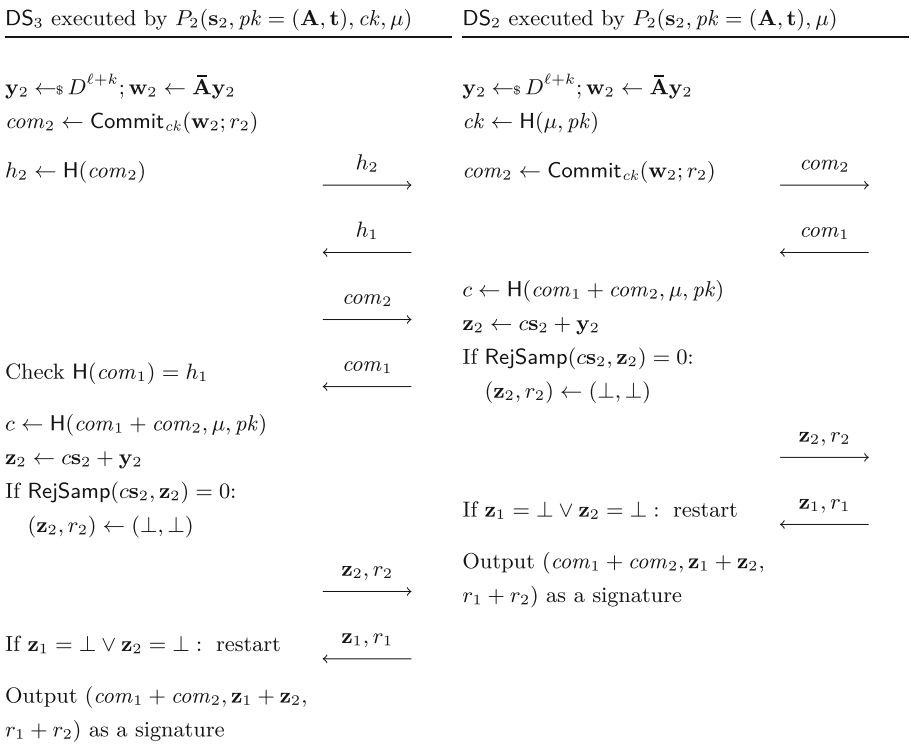

Fig. 1. Comparison of different instantiations of FSwA-based 2-party signing protocols

The second problem is more generic and could also occur in the standard Fiat–Shamir-style two party signing: if P_2 sends out $\mathbf{w}_2$ first, then the simulator does not know $\mathbf{w}_1$. In FS-style constructions, the usual strategy for signing oracle query simulation is to first sample a challenge c by itself, generate a simulated transcript $(\mathbf{w}_2, c, \mathbf{z}_2)$ by invoking a special HVZK simulator on c, and then program the random oracle H such that its output is fixed to a predefined challenge c. In the two-party setting, however, derivation of the *joint* challenge $c = \mathsf{H}(\mathbf{w}_1 + \mathbf{w}_2, \mu, pk)$ requires contribution from $\widetilde{P}_1$ and thus there is no way for the simulator to program H in advance. Not only the proof doesn't go through, but also this naive construction is amenable to a concrete attack, which allows malicious $\widetilde{P}_1$ to create a valid forgery by adaptively choosing $\mathbf{w}_1$ after seeing $\mathbf{w}_2$. In [30] we describe this attack relying on a variant of Wagner's generalized birthday problem [54,86].

Homomorphic Commitment to Simulate Aborts. We now present an intermediate provably secure protocol that circumvents the above issues. See DS_3 in Fig. 1. To address the first issue with aborts, each player P_j now commits to an initial Σ-protocol message $\mathbf{w}_j$ using an *additively homomorphic* commitment com_j. Thanks to the hiding property, each party leaks no useful information

about $\mathbf{w}_j$ until the rejection sampling is successful, and thus it is now possible to simulate a rejected transcript $(com_j, c, \perp)$. Then P_j broadcasts a hash based commitment to com_j, to deal with the second issue. Once all parties have done this, com_j's are revealed in the next round and checked against the hashes. 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. The verification now receives a signature consisting of three elements $(com, \mathbf{z}, r)$ and simply checks that $\bar{\mathbf{A}}\mathbf{z} - c\mathbf{t} \pmod q$ and r form a correct opening to com, where the challenge is recomputed as $c \leftarrow \mathsf{H}(com, \mu, pk)$.

We note that the extra round for hash commitment is a standard technique, previously used in multiple three-round protocols, such as Nicolosi et al. [75] and Bellare and Neven [9] in the discrete log setting, and Bansarkhani and Sturm [4] in their FSwA-based instantiation. This way, the simulator for honest P_2 can successfully extract corrupt $\widetilde{P}_1$'s share com_1 by keeping track of incoming queries to H (when modeled as a random oracle), and program H such that $\mathsf{H}(com_1 + com_2, \mu, pk) := c$ before revealing com_2. For the completeness, in [30] we provide a formal security proof for DS_3 by showing a reduction to Module-LWE *without* relying on the forking lemma [9,79]. This is made possible by instantiating the construction with unconditionally binding commitment, which allows us to avoid rewinding the adversary and apply the *lossy identification* technique by Abdalla et al. [1].

One efficiency issue is, that the protocol has to be restarted until *all* parties pass the rejection sampling step simultaneously. All previous FSwA-based multi-signatures also had the same issues, but we can mitigate by running sufficiently many parallel executions of the protocol at once, or by carefully choosing the parameters for rejection sampling. To further reduce the number of aborts, we chose to instantiate the protocol with Dilithium-"G" [37] instead of the one submitted to NIST competition [65].

Trapdoor Commitment to Avoid the Extra Round. Although DS_3 is secure, the first round of interaction may seem redundant, since the parties are essentially "committing to a commitment". We show that the extra hash commitment round can be indeed dropped by adding a trapdoor feature to the commitment scheme, which allows the so-called *straight-line simulation* technique by Damgård [28]. We present our main two-round protocol DS_2 in Fig. 1. This way, the simulation of honest P_2 does not require the knowledge of corrupt $\widetilde{P}_1$'s commitment share; instead, the simulator can now simply send a commitment com_2 (to some random value) and then later equivocate to an arbitrary value using the known trapdoor td associated with a trapdoored commitment key tck. Concretely, the simulator need not program the random oracle this time, and instead derives a challenge $c \leftarrow \mathsf{H}(com_1 + com_2, \mu, pk)$ as the real honest party would do. Now the simulator invokes a (special) HVZK simulator with c as input, to obtain a transcript $(\mathbf{w}_2, c, \mathbf{z}_2)$. With some constant probability it equivocates com_2 to $\mathbf{w}_2$, or otherwise sends out $\perp$ to simulate aborts. We also stress that the *per-message commitment key* $ck \leftarrow \mathsf{H}(\mu, pk)$ is crucial in the two-round protocol; if a single ck is used across all signing attempts, then a concurrent attack similar

to the one against the naive construction becomes applicable. Unlike the three-round protocol, we present a security proof 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 be able to reduce security to the binding property. We leave for future work a tighter security proof, as well as a proof in the quantum random oracle model. We can now convert to a two-round multi-signature scheme in the plain public key model: following Bellare–Neven [9] the protocol now generates *per-user challenges* $c_j = \mathsf{H}(\mathbf{t}_j, \sum_j com_j, \mu, L)$ for each user's public key $\mathbf{t}_j \in L$, instead of interactively generating the fixed joint public key $\mathbf{t}$ in advance. The verification algorithm is adjusted accordingly: given $(com, \mathbf{z}, r)$ and a list of public keys L, the verifier checks that $\bar{\mathbf{A}}\mathbf{z} - \sum_j c_j \mathbf{t}_j \pmod q$ and r form a correct opening to com, where c_j's are recomputed as in the signing protocol.

1.3 Related Work

The FSwA paradigm was first proposed by Lyubashevsky [62,63] and many efficient signature schemes following this framework have been devised, such as GLP [53], BLISS [36], Dilithium [65] and qTESLA [14]. Bansarkhani and Sturm [4] 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 [67], three-round protocol with tight security proof [42] and proof in QROM [43], ID-based blind multi-signature [85] and ID-based proxy multi-signature [83]. However, as mentioned earlier the security proofs for all these multi-signatures are either incomplete or rely on a non-standard heuristic assumption. Choi and Kim [22] proposed a linearly homomorphic multi-signature from lattices trapdoors. Kansal and Dutta [55] constructed a single-round multi-signature scheme relying on the hardness of SIS, which was soon after broken by Liu et al. [61]. Several lattice-based threshold *ring signatures* exist in the literature, such as Cayrel et al. [21], Bettaieb and Schrek [13], and Torres et al. [84]. Döroz et al. [34] devised lattice-based *aggregate signature schemes* relying on rejection sampling. Very recently, Esgin et al. [39] developed FSwA-based *adaptor signatures* with application to blockchains.

Our two-round protocols rely on trapdoor commitment to enable the straight-line simulation of ZK. The trick is originated in a concurrent ZK proof by Damgård [28] and similar ideas have been extensively used in the ZK literature [12,23–25], to turn honest verifier ZK proof into full-fledged ZK. Moreover, recent efficient lattice-based ZK proofs [16,31,40,41,87] 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. [5] 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 [51], homomorphic trapdoor functions [52], Chameleon hash [18,38] or mercurial commitment [58].

Comparison with Bendlin et al. [10] An entirely different approach to constructing threshold signatures based on lattices relies not on the Fiat–Shamir with aborts paradigm, but on GPV hash-and-sign signatures [50]. This approach was introduced by Bendlin et al. in [10], who described how to implement Peikert's hash-and-sign signatures [78] in a multiparty setting. Compared to the approach in this paper, it has the advantage of realizing the same distributed signature scheme (e.g., with the same size bound for verification) independently of the number of parties, and in particular, signature size does not grow with the number of parties. Moreover, it supports more general access structure than the full threshold considered in this paper (although their protocol does not withstand dishonest majority for the sake of information-theoretic security, while our protocol does tolerate up to $n-1$ corrupt parties). Its main downside, however, is that the most expensive part of Peikert's signing algorithm, namely the offline lattice Gaussian sampling phase, is carried out using *generic multiparty computation* (this is the first step of the protocol π_{Perturb} described in [10, Fig. 23]). This makes it difficult to estimate the concrete efficiency of Bendlin et al.'s protocol, but since the Peikert signature scheme is fairly costly even in a single-user setting, the protocol is unlikely to be practical.

In contrast, while our protocols do use discrete Gaussian sampling, it is only carried out *locally by each party*, and it is Gaussian sampling over $\mathbb{Z}$ rather than a lattice, which is considerably less costly. Furthermore, while we also use lattice trapdoors as a proof technique in the trapdoor commitment scheme of our two-round protocol, trapdoor Gaussian sampling is never carried out in the actual protocol, only in the simulation (the actual protocol has no trapdoor). Thus, our protocols entirely avoid the expensive machinery present in Bendlin et al.'s scheme, and have a fully concrete instantiation (at the cost of signatures increasing in size with the number of parties).

2 Preliminaries

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.

2.1 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 [37], 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 $v' = v \bmod q$. 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 $\rho_{\mathbf{v},s}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{v}\|_2^2 / s^2)$ 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 [72, Theorem 3.3][40, 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 $D^m_{s\sqrt{n}}$.*

2.2 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 $\mathbf{x} \in R_q^{\ell+k} \setminus \{\mathbf{0}\}$ 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 $\mathbf{t} := [\mathbf{A}|\mathbf{I}] \cdot \mathbf{s}$.*

2.3 Fiat–Shamir with Aborts Framework and Dilithium-G

Algorithm 1. Key generation

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

Algorithm 2. Signature verification

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

Algorithm 3. Signature generation

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

Algorithm 4. $\mathsf{Trans}(sk, c)$

1: $\mathbf{y} \leftarrow\!\!\$\ D_s^{\ell+k}$
2: $\mathbf{w} \coloneqq \bar{\mathbf{A}}\mathbf{y}$
3: $\mathbf{z} \coloneqq c\mathbf{s} + \mathbf{y}$
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** $(\bot, c, \bot)$

Algorithm 5. $\mathsf{SimTrans}(pk, c)$

1: $\mathbf{z} \leftarrow\!\!\$\ D_s^{\ell+k}$
2: $\mathbf{w} \coloneqq \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$
3: With prob. $1/M$:
4: **return** $(\mathbf{w}, c, \mathbf{z})$
5: Otherwise:
6: **return** $(\bot, c, \bot)$

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 [63, Lemma 4.4] we restate 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. *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 claim by Lyubashevsky (adapted from [63, Lemma 4.5]) 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 $\epsilon < 2^{-100}$, thereby ensuring > 100 bits of security).

Lemma 3. *For* $V \subseteq R^m$ *let* $T = \max_{\mathbf{v}\in V} \|\mathbf{v}\|_2$. *Fix some* t *such that* $t = \omega(\sqrt{\log(mN)})$ *and* $t = o(\log(mN))$. *If* $\sigma = \alpha T$ *for any positive* α, *then*

$$\Pr[M \ge D_s^m(\mathbf{z})/D_{\mathbf{v},s}^m(\mathbf{z}) : \mathbf{z} \leftarrow\!\!\$\ D_s^m(\mathbf{z})] \ge 1 - \epsilon$$

where $M = e^{t/\alpha+1/(2\alpha^2)}$ *and* $\epsilon = 2e^{-t^2/2}$.

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 [56, Lemma 4.3] to prove the security of Dilithium signature instantiated with the uniform distribution. We remark that the simulator in Algorithm 5 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 $(\mathbf{w}, c, \perp)$ [11,64] (without assuming some ad-hoc assumptions like rejection-DCK [5] or rejected-LWE [43]). We give a proof in the full version of this paper [30].

Lemma 4 (Non-abort Special Honest Verifier Zero Knowledge). *Let* $m = \ell+k$ *and* $T = \max_{c\in C,\mathbf{s}\in S_\eta^m} \|c\cdot\mathbf{s}\|_2$. *Fix some* t *such that* $t = \omega(\sqrt{\log(mN)})$ *and* $t = o(\log(mN))$. *If* $\sigma = \alpha T$ *for any positive* α, *then for any* $c \in C$ *and* $\mathbf{s} \in S_\eta^m$, *the output distribution of* $\mathsf{Trans}(sk, c)$ *(Algorithm 4) is within statistical distance* ϵ/M *of the output distribution of* $\mathsf{SimTrans}(pk, c)$ *(Algorithm 5), where* $M = e^{t/\alpha+1/(2\alpha^2)}$ *and* $\epsilon = 2e^{-t^2/2}$. *Moreover,* $\Pr[\mathsf{Trans}(\mathbf{s}, c) \neq (\perp, c, \perp)] \geq (1-\epsilon)/M$.

2.4 Trapdoor Homomorphic Commitment Scheme

Below we formally define a trapdoor commitment scheme. In [30] we also define standard security requirements like hiding and binding, as well as the two additional properties required by our protocols: additive homomorphism and uniform key. The lattice-based commitments described in Sect. 4 indeed satisfy all of them. 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 [77] trivially satisfy this property. The additive homomorphism is also needed to preserve the algebraic structure of the first "commit" message of FSwA.

Definition 4 (Trapdoor Commitment Scheme). *A trapdoor commitment scheme* TCOM *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* S_{td} *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.*

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

A trapdoor commitment is said to be secure if it is unconditionally hiding, computationally binding, and for any $msg \in S_{msg}$*, the statistical distance* ϵ_{td} *between* (ck, msg, com, r) *and* (tck, msg, com', r') *is negligible in* λ*, where* $cpp \leftarrow \mathsf{CSetup}(1^\lambda); ck \leftarrow \mathsf{CGen}(cpp); r \leftarrow_\$ D(S_r); com \leftarrow \mathsf{Commit}_{ck}(msg; r)$ *and* $(tck, td) \leftarrow \mathsf{TCGen}(cpp); com' \leftarrow \mathsf{TCommit}_{tck}(td); r' \leftarrow \mathsf{Eqv}_{tck}(td, com', msg)$*.*

2.5 Security Notions for n-out-of-n Signature and Multi-signature

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 [59] for two-party signing protocol. Our definition can be regarded as its generalization to n-party setting. Following Lindell, we assume that the key generation can be invoked only once, while many signing sessions can be executed concurrently. The main difference is that, in our protocols all players have the same role, and therefore we fix wlog the index of honest party and challenger to n, who has to send out the message first in each round of interaction. This way, we assume that the adversary $\mathcal{A}$ who corrupts $P_1, \ldots, P_{n-1}$ is *rushing* by default (i.e., $\mathcal{A}$ is allowed to choose their own messages based on P_n's message).

Definition 5 (Distributed Signature Protocol). *A distributed signature protocol* DS *consists of the following algorithms.*

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

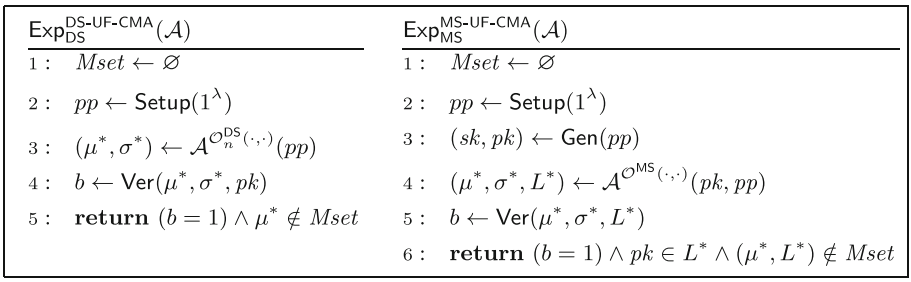

Fig. 2. DS-UF-CMA and MS-UF-CMA experiments. The oracles $\mathcal{O}^{\mathsf{DS}}_n$ and $\mathcal{O}^{\mathsf{MS}}$ are described in Figs. 3 and 4. In the left (resp. right) experiment, *Mset* is the set of all inputs μ (resp. (μ, L)) such that (sid, μ) (resp. $(sid, (\mu, L))$) was queried by $\mathcal{A}$ to its oracle as the first query with identifier $sid \neq 0$ (resp with any identifier sid). Note that pk in the left experiment is the public verification key output by P_n when it completes $\mathsf{Gen}_n(pp)$.

Oracle $\mathcal{O}^{\mathsf{DS}}_n(sid, m)$

The oracle is initialized with public parameters pp generated by Setup algorithm. The variable *flag* is initially set to false.

Key Generation Upon receiving $(0, m)$, if *flag* = true then return $\perp$. Otherwise do the following:

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

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

- If the oracle is queried with sid for the first time then parse the incoming message m as μ. It initializes a machine $\mathcal{M}_{sid}$ running the instructions of party P_n in the distributed signing protocol $\mathsf{Sign}_n(sid, sk_n, pk, \mu)$. The machine $\mathcal{M}_{sid}$ is initialized with the key share and any state information stored by $\mathcal{M}_0$ at the end of the key generation phase. The message μ to be signed is included in *Mset*. If P_n sends the first message in the signing protocol, then this message is the oracle reply.
- If $\mathcal{M}_{sid}$ has been already initialized then the oracle hands the machine $\mathcal{M}_{sid}$ the next incoming message m and returns the next message sent by $\mathcal{M}_{sid}$. If $\mathcal{M}_{sid}$ concludes with local output σ, then the output obtained by $\mathcal{M}_{sid}$ is returned.

Fig. 3. Honest party oracle for the distributed signing protocol.

Definition 6. (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 polynomial time adversary* $\mathcal{A}$*, its advantage*

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

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

Oracle $\mathcal{O}^{\mathsf{MS}}(sid, m)$

The oracle is initialized with public parameters pp generated by Setup algorithm.

Signature Generation Upon receiving (sid, m) do the following:

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

Fig. 4. Honest party oracle for the multi-signature protocol.

Next we define the standard security notion of multi-signature protocol in the plain public-key model. The following definitions are adapted from [9], but the syntax is made consistent with n-out-of-n signing. The main difference from the distributed signature is, that there is no interactive key generation protocol 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 keys, instead of a single combined public key as in the prior case. We also note that n is now the number of *maximum* number of parties involved in a single execution of signing protocol, since the size of L may vary depending on a protocol instance.

Definition 7 (Multi-signature Protocol). *A multisignature protocol* MS *consists of the following 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 non-interactive key generation algorithm that outputs a key pair on a public parameter pp as input.*
- $\mathsf{Sign}(sid, sk, pk, \mu, L) \to \sigma$*: The interactive signing algorithm that is run by a party* P *holding a key pair* (sk, pk)*. Each* P *runs the protocol on session ID sid, its signing key sk, public key pk, message to be signed* μ*, and a set of co-signers' public keys* L *as input. At the end of the protocol* P *obtains a signature* σ *as output.*
- $\mathsf{Ver}(\sigma, \mu, L) \to b$*: 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 8 (MS-UF-CMA Security). *A multisignature protocol* MS *is said to be* MS-UF-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 DS_2: Two-Round n-out-of-n Signing from Module-LWE and Module-SIS

3.1 Protocol Specification and Overview

This section presents our main construction: provably secure two-round n-out-of-n protocol $\mathsf{DS}_2 = (\mathsf{Setup}, (\mathsf{Gen}_j)_{j\in[n]}, (\mathsf{Sign}_j)_{j\in[n]}, \mathsf{Ver})$, formally specified in Fig. 5. As mentioned in Sect. 2.5 all players have the same role and hence we only present n-th player's behavior. The protocol is built on top of additively homomorphic trapdoor commitment scheme TCOM with uniform keys (see Definition 4 and [30] for the formal definitions), and we will describe concrete instances of TCOM later in Sect. 4. We go over high-level ideas for each step below.

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-bound of Lemma 2
$B = \gamma\sigma\sqrt{N(\ell+k)}$	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
$C = \{c \in R : \|c\|_\infty = 1 \wedge \|c\|_1 = \kappa\}$	Challenge space where $\lvert C\rvert = \binom{N}{\kappa}2^\kappa$
$S_\eta = \{x \in R : \|x\|_\infty \leq \eta\}$	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 can proceed
$M_n = M^n$	The expected number of restarts until all n parties proceed simultaneously
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

Parameter Setup. We assume that a trusted party invokes $\mathsf{DS}_2.\mathsf{Setup}(1^\lambda)$ 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 $\mathsf{TCOM.CSetup}(1^\lambda)$). Most parameters commonly appear in the literature about the Fiat–Shamir with aborts paradigm (e.g. [37,63]) and we therefore omit the details here. The bit length l_1 and l_2 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 in Sect. 3.2.

Key Generation. The key generation $\mathsf{DS}_2.\mathsf{Gen}_n$ essentially follows the approach by Nicolosi et al. [75] for two-party Schnorr signing. 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[2]; as long as there is at least one honest party sampling a matrix share correctly, the resulting combined matrix is guaranteed to follow the uniform distribution. For the exact same reason, the exchange of $\mathbf{t}_j$'s is also carried with the 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* [70]). Furthermore, the party's index j is concatenated with the values to be hashed for the sake of "domain separation" [8]. 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 crucial step of $\mathsf{DS}_2.\mathsf{Sign}_n$ in Fig. 5 is commitment key generation at **Inputs** 3; in fact, if instead 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 [30] we sketch a variant of the concurrent attack due to Drijvers et al. [35]. The original attack was against two-round Schnorr multi-signatures including BCJ scheme [3], 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 message-dependent commitment key, following the mBCJ scheme of Drijvers et al.

Then the signing protocol starts by exchanging the first "commit" messages of Σ-protocol, from which all parties derive a single joint challenge $c \in C$ via a random oracle. 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. Finally, the com_j's and r_j's are added together in a meaningful way, thanks to the homomorphic property of commitment scheme.

[2] We remark that the "commitments" generated by H_1 and H_2 in Fig. 5 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 [76].

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

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. 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 g_n.
2. Upon receiving g_j for all $j \in [n-1]$ send out $\mathbf{A}_n$.
3. Upon receiving $\mathbf{A}_j$ for all $j \in [n-1]$:
   a. If $\mathsf{H}_1(\mathbf{A}_j, j) \neq g_j$ for some j then send out ABORT.
   b. Otherwise set public random matrix $\bar{\mathbf{A}} := [\mathbf{A}|\mathbf{I}] \in R_q^{k\times(\ell+k)}$, where $\mathbf{A} := \sum_{j\in[n]} \mathbf{A}_j$.

Key Pair Generation

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

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

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

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 $\mathsf{DS}_2.\mathsf{Gen}_n(pp)$ has been previously invoked. If a party halts with ABORT at any point, then all $\mathsf{Sign}_n(sid, sk_n, pk, \mu)$ executions are aborted.

Inputs

1. P_n receives a unique session ID sid, $sk_n = \mathbf{s}_n$, $pk = (\mathbf{A}, \mathbf{t})$ and message $\mu \in \{0,1\}^*$ as input.
2. P_n verifies that sid has not been used before (if it has been, the protocol is not executed).
3. P_n locally computes a per-message commitment key $ck \leftarrow \mathsf{H}_3(\mu, pk)$.

Signature Generation P_n works as follows:

1. Compute the first message as follows.
   a. Sample $\mathbf{y}_n \leftarrow\!\!\$\ D_s^{\ell+k}$ and compute $\mathbf{w}_n := \bar{\mathbf{A}}\mathbf{y}_n$.
   b. Compute $com_n \leftarrow \mathsf{Commit}_{ck}(\mathbf{w}_n; r_n)$ with $r_n \leftarrow\!\!\$\ D(S_r)$.
   c. Send out com_n.
2. Upon receiving com_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 $c \leftarrow \mathsf{H}_0(com, \mu, pk)$.
   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
   $$\min\left(1, D_s^{\ell+k}(\mathbf{z}_n)/(M \cdot D_{c\mathbf{s}_n,s}^{\ell+k}(\mathbf{z}_n))\right)$$
   send out $(\mathbf{z}_n, r_n)$; otherwise send out RESTART and go to 1.
3. Upon receiving RESTART from some party go to 1. Otherwise upon receiving $(\mathbf{z}_j, r_j)$ 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 $r := \sum_{j\in[n]} r_j$.

If the protocol does not abort, P_n obtains a signature $(com, \mathbf{z}, r)$ as local output.

Algorithm $\mathsf{DS}_2.\mathsf{Ver}((com, \mathbf{z}, r), \mu, pk)$

Upon receiving a message μ, signature $(com, \mathbf{z}, r)$, and combined public key $pk = (\mathbf{A}, \mathbf{t})$, generate a commitment key $ck \leftarrow \mathsf{H}_3(\mu, pk)$, derive a challenge $c \leftarrow \mathsf{H}_0(com, \mu, pk)$ and reconstruct $\mathbf{w} := \bar{\mathbf{A}}\mathbf{z} - c\mathbf{t}$. Then accept if $\|\mathbf{z}\|_2 \leq B_n$ and $\mathsf{Open}_{ck}(com, r, \mathbf{w}) = 1$.

Fig. 5. Distributed n-out-of-n signature scheme.

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 plain Dilithium as a base scheme then the bound would grow almost linearly). Hence together with the tail-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 perfect correctness, the bound check can be also carried out 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).

3.2 Asymptotic Efficiency Analysis

Number of Aborts and Signature Size. As indicated in Table 1 the probability that all participants simultaneously 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 $\alpha \geq 11n$ [37], 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 translates to a signature size increase of a factor of roughly[3] $O(\log n)$, so the scaling in terms of the number of parties is reasonable. In addition, when using the trapdoor commitment scheme of Sect. 4, one can substantially reduce the signature size by using the common Fiat–Shamir trick of expressing the signature as $(c, \mathbf{z}, r)$ instead of $(com, \mathbf{z}, r)$, and carrying out the verification by first recomputing the commitment using the randomness r, and then checking the consistency of the challenge: $c \stackrel{?}{=} \mathsf{H}_0(com, \mu, pk)$. This keeps signature size close to the original Dilithium-G, despite the relatively large size of commitments.

We expect that a number of further optimizations are possible to improve the efficiency of this protocol in both asymptotic and concrete terms (e.g., by relying on stronger assumptions like (Mod-)NTRU), although this is left for further work. Accordingly, we also leave for further work the question of providing concrete parameters for the protocol, since the methodology for setting parameters is currently a moving target (e.g., the original parameters for Dilithium-G

[3] To be more precise, since the verification bound scales as $n^{3/2}$, one should also increase q by the same bound to avoid arithmetic overflow. This makes the MSIS problem harder, but the MLWE easier if the dimension is kept unchanged. To keep the same security level, one should therefore also increase N by a factor of $1 + O(\frac{\log n}{\log q_0})$ where q_0 is the value of q in the single-user setting. Therefore, one could in principle argue that signature size actually scales as $O(\log^2 n)$. However, one typically chooses $q_0 > 2^{20}$, and therefore even in settings with billions of parties, $\frac{\log n}{\log q_0} < 2$. Thus, one can effectively regard N as independent of n.

are not considered up-to-date), there is arguably no good point of comparison in the literature (in particular, no previous lattice-based two-round protocol), and again, a concrete instantiation would likely rely on stronger assumptions to achieve better efficiency anyway.

Round Complexity. If this protocol is used as it is, it only outputs a signature after the three rounds with probability $1/M_n$ (which is $1/3$ with the parameters above). As a result, to effectively compute a signature, it has to be repeated M_n times on average, and so the expected number of rounds is in fact larger than 2 ($2M_n = 6$ in this case). One can of course adjust the parameters to reduce M_n to any constant greater than 1, or even to $1 + o(1)$ by picking e.g. $\alpha = \Theta(n^{1+\epsilon})$; this results in an expected number of rounds arbitrarily close to 2. Alternatively, one can keep a 2-round protocol while ensuring that the parties output a signature with overwhelming probability, simply by running sufficiently many parallel executions of the protocol at once: $\lambda / \left(\log \frac{M_n}{M_n - 1}\right)$ parallel executions suffice if λ is the security parameter.

3.3 Security

Theorem 1. *Suppose the trapdoor commitment scheme* TCOM *is secure, additively homomorphic and has uniform keys. For any probabilistic polynomial-time adversary* $\mathcal{A}$ *that initiates a single key generation protocol by querying* $\mathcal{O}_n^{\mathsf{DS}_2}$ *with* $sid = 0$*, initiates* Q_s *signature generation protocols by querying* $\mathcal{O}_n^{\mathsf{DS}_2}$ *with* $sid \neq 0$*, and makes* Q_h *queries to the random oracle* $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$*, the protocol* DS_2 *of Fig. 5 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* $\beta = 2\sqrt{B_n^2 + \kappa}$.

We give a sketch of the security proof. The full proof with concrete security bound is given in [30]. We also remark that its multi-signature variant MS_2 [30] can be proven secure relying on essentially the same idea. We show that given any efficient adversary $\mathcal{A}$ that creates a valid forgery with non-negligible probability, one can break either $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ assumption or computational binding of TCOM.

Key Generation Simulation. For the key generation phase, 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 $\mathsf{MLWE}_{q,k,\ell,\eta}$ assumption, the honest party oracle simulator can replace $\mathbf{t}_n$ with such a vector. Therefore, the distribution of combined public key $\mathbf{t} = \sum_{j \in [n]} \mathbf{t}_j$ is also indistinguishable from the uniform distribution. Thanks to the random oracle commitment, after the adversary has submitted g'_j for each $j \in [n-1]$ one can extract the adversary's public key share $\mathbf{t}_j$, with which the simulator sets its share a posteriori $\mathbf{t}_n := \mathbf{t} - \sum_{j \in [n-1]} \mathbf{t}_j$ and programs the random oracle accordingly $\mathsf{H}_2(\mathbf{t}_n, n) := g'_n$. Using the same argument, one can set a random matrix share $\mathbf{A}_n := \mathbf{A} - \sum_{j \in [n-1]} \mathbf{A}_j$ given a resulting random matrix $\mathbf{A} \leftarrow\!\!\$\ R_q^{k \times \ell}$. 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 $(\mathbf{A}, \mathbf{t})$ is uniformly distributed in $R_q^{k\times\ell} \times R_q^k$, so replacing it with a $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ instance doesn't change the view of adversary at all, if $\mathbf{A}'$ is regarded as $\mathbf{A}' = [\mathbf{A}|\mathbf{t}]$.

Signature Generation Simulation. The oracle simulation closely follows the one for mBCJ [35]. Concretely, the oracle simulator programs H_3 so that for each signing query it returns tck generated via $(tck, td) \leftarrow \mathsf{TCGen}(cpp)$, and for the specific crucial query that is used to create a forgery it returns an actual commitment key $ck \leftarrow \mathsf{CGen}(cpp)$, which has been received by the reduction algorithm as a problem instance of the binding game. This way, upon receiving signing queries the oracle simulator can send out a "fake" commitment $com_n \leftarrow \mathsf{TCommit}_{tck}(td)$ at the first round, and then the known trapdoor td allows to later equivocate to a simulated first message of the Σ-protocol *after* the joint random challenge $c \in C$ has been derived; formally, it samples a simulated signature share $\mathbf{z}_n \leftarrow\!\!\$\ D_s^{\ell+k}$ and then derives randomness as $r_n \leftarrow \mathsf{Eqv}_{tck}(td, com_n, \mathbf{w}_n := \bar{\mathbf{A}}\mathbf{z}_n - c\mathbf{t}_n)$. On the other hand, when the reduction obtains two openings after applying the forking lemma it can indeed break the binding property with respect to a real commitment key ck.

Forking Lemma. Our proof is relying on the forking lemma [9,79]. 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 under the same key, which seems to require some kind of rewinding technique. After applying the forking lemma, the adversary submits two forgeries with distinct challenges $c^* \neq \hat{c}^*$, with which we can indeed find a solution to $\mathsf{MSIS}_{q,k,\ell+1,\beta}$, or otherwise break computational binding with respect to ck. Concretely, after invoking the forking lemma, we obtain two forgeries $(com^*, \mathbf{z}^*, r^*, \mu^*)$ and $(\hat{com}^*, \hat{\mathbf{z}}^*, \hat{r}^*, \hat{\mu}^*)$ such that $c^* = \mathsf{H}(com^*, \mu, pk) \neq \mathsf{H}(\hat{com}^*, \hat{\mu}^*, pk) = \hat{c}^*$, $com^* = \hat{com}^*$, $\mu^* = \hat{\mu}^*$, and $\mathsf{H}(\mu^*, pk) = \mathsf{H}(\hat{\mu}^*, pk) = ck$. Since both forgeries are verified, we have $\|\mathbf{z}^*\|_2 \leq B_n \wedge \|\hat{\mathbf{z}}^*\|_2 \leq B_n$, and

$$\mathsf{Open}_{ck}(com^*, r^*, \bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t}) = \mathsf{Open}_{ck}(com^*, \hat{r}^*, \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}) = 1.$$

If $\bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t} \neq \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}$ then it means that computational binding is broken with respect to a commitment key ck. Suppose $\bar{\mathbf{A}}\mathbf{z}^* - c^*\mathbf{t} = \bar{\mathbf{A}}\hat{\mathbf{z}}^* - \hat{c}^*\mathbf{t}$. Rearranging it leads to

$$[\mathbf{A}|\mathbf{I}|\mathbf{t}] \begin{bmatrix} \mathbf{z}^* - \hat{\mathbf{z}}^* \\ \hat{c}^* - c^* \end{bmatrix} = \mathbf{0}.$$

Recalling that $[\mathbf{A}'|\mathbf{I}] = [\mathbf{A}|\mathbf{t}|\mathbf{I}]$ is an instance of $\mathsf{MSIS}_{q,k,\ell+1,\beta}$ problem, we have found a valid solution if $\beta = \sqrt{(2B_n)^2 + 4\kappa}$, since $\|\mathbf{z}^* - \hat{\mathbf{z}}^*\|_2 \leq 2B_n$ and $0 < \|\hat{c}^* - c^*\|_2 \leq \sqrt{4\kappa}$.

4 Lattice-Based Commitments

In this section, we describe possible constructions for the lattice-based commitment schemes used in our protocols. The three-round protocol DS_3 requires a

statistically binding, computationally hiding homomorphic commitment scheme, whereas the two-round protocol DS_2 of Sect. 3 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. [7]. More precisely, the first type of commitment scheme is a simple variant of the scheme of [7], 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 [7], 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 [7] with Micciancio–Peikert style trapdoors [71]. To prevent statistical learning attacks on the trapdoor sampling, however, it is important to sample the randomness in the commitment according to a discrete Gaussian distribution, in contrast with Baum et al.'s original scheme.

4.1 Statistically Binding Commitment Scheme

We first describe a statistically binding commitment scheme from lattices. The scheme, described in [30], is a simple variant of the scheme from [7], 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 [71, 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 $\big|B_{m'}(0, 2B)\big|/q^{mN} = 2^{-\Omega(N)}$. Now $\big|B_{m'}(0, 2B)\big| = o\big((2\pi e/m'N)^{m'N/2} \cdot (2B)^{m'N}\big)$. Hence, picking for example $m' = 2m$, we get:

$$\frac{\big|B_{m'}(0, 2B)\big|}{q^{mN}} \ll \Big(\frac{4\pi e \cdot B^2}{mNq}\Big)^{mN},$$

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 $\big[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{s}\big]^T$ 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:

$$\mathbf{f}_1 = \mathbf{r}_1 + \hat{\mathbf{A}}_1' \cdot \begin{bmatrix} \mathbf{r}_2 \\ \mathbf{s} \end{bmatrix} \qquad \mathbf{f}_2 = \mathbf{r}_2 + \hat{\mathbf{A}}_2' \cdot \mathbf{s} + \mathbf{x},$$

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.

4.2 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. 6. 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 for verification should be large enough to accommodate 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 [71, 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 $\hat{\mathbf{A}}_1$ (or equivalently, to the $\mathsf{MSIS}_{q,1,\ell+2w-1,2B}$ problem), and finding such a solution is hard.

Note that since the first entry of $\hat{\mathbf{A}}_1$ is invertible, one can put it in the form $[\mathbf{A}|\mathbf{I}]$ without loss of generality to express it directly as an MSIS problem in the sense of Definition 2. It also 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 [66, 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 [66, 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 [71, Sect. 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 [71, Lemma 2.9]. Thus, one can pick $s = \Theta(N^{3/2} \log^2 N)$.

From the previous paragraphs, we can in particular see that the trapdoor commitment satisfies the security requirements of Definition 4. Thus, to summarize, we have proved the following theorem.

Theorem 2. *The trapdoor commitment scheme of Fig. 6, with the following choice of parameters:*

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

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

Note that we did not strive for optimality in the parameter selection; a finer analysis is likely to lead to 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.

Protocol Lattice-based Commitment Scheme

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

$\mathsf{CGen}(cpp)$ 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.

$\mathsf{Commit}_{ck}(x)$ 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 $\|\mathbf{r}\|_2 \leq B$.

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

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

$\mathsf{TCGen}(cpp)$ 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 tck, where $\mathbf{G}$ is given by:

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

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

$\mathsf{Eqv}_{tck}(\mathbf{R}, \mathbf{f}, x)$ uses the trapdoor discrete Gaussian sampling algorithm of Micciancio–Peikert [71, Algorithm 3] (or faster variants such as the one described in [45]) 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. 6. Equivocable variant of the commitment from [7].

Acknowledgment. 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). We thank Cecilia Boschini for her insightful comments on an early form of our two-round protocol. We are grateful for helpful suggestions by anonymous reviewers.

References

1. Abdalla, M., Fouque, P.A., Lyubashevsky, V., Tibouchi, M.: Tightly secure signatures from Lossy identification schemes. J. Cryptol. **29**(3), 597–631
2. Abe, M., Fehr, S.: Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 317–334. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_20
3. Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: ACM CCS 2008, pp. 449–458. ACM Press (2008)
4. El Bansarkhani, R., Sturm, J.: An efficient lattice-based multisignature scheme with applications to bitcoins. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 140–155. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_9
5. Barthe, G., et al.: Masking the GLP lattice-based signature scheme at any order. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 354–384. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_12
6. Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Rossi, M., Tibouchi, M.: GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In: ACM CCS 2019, pp. 2147–2164. ACM Press (2019)
7. Baum, C., Damgård, I., Lyubashevsky, V., Oechsner, S., Peikert, C.: More efficient commitments from structured lattice assumptions. In: Catalano, D., De Prisco, R. (eds.) SCN 2018. LNCS, vol. 11035, pp. 368–385. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98113-0_20
8. Bellare, M., Davis, H., Günther, F.: Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 3–32. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_1
9. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: ACM CCS 2006, pp. 390–399. ACM Press (2006)
10. Bendlin, R., Krehbiel, S., Peikert, C.: How to share a lattice trapdoor: threshold protocols for signatures and (H)IBE. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 218–236. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38980-1_14
11. Benhamouda, F., Camenisch, J., Krenn, S., Lyubashevsky, V., Neven, G.: Better zero-knowledge proofs for lattice encryption and their application to group signatures. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 551–572. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_29
12. Benhamouda, F., Krenn, S., Lyubashevsky, V., Pietrzak, K.: Efficient zero-knowledge proofs for commitments from learning with errors over rings. In: ESORICS 2015, Part I. LNCS, vol. 9326, pp. 305–325. Springer, Heidelberg. https://doi.org/10.1007/978-3-319-24174-6_16
13. Bettaieb, S., Schrek, J.: Improved lattice-based threshold ring signature scheme. In: Gaborit, P. (ed.) PQCrypto 2013. LNCS, vol. 7932, pp. 34–51. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38616-9_3
14. Bindel, N., et al.: qTESLA. Technical report, National Institute of Standards and Technology

15. Boneh, D., et al.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 565–596. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_19
16. Bootle, J., Lyubashevsky, V., Seiler, G.: Algebraic techniques for short(er) exact lattice-based zero-knowledge proofs. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 176–202. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_7
17. Canetti, R., Makriyannis, N., Peled, U.: UC non-interactive, proactive, threshold ecdsa. Cryptology ePrint Archive, Report 2020/492
18. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
19. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Two-party ECDSA from Hash proof systems and efficient instantiations. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 191–221. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_7
20. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold EC-DSA. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 266–296. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_10
21. Cayrel, P.-L., Lindner, R., Rückert, M., Silva, R.: A Lattice-based threshold ring signature scheme. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 255–272. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14712-8_16
22. Choi, R., Kim, K.: Lattice-based multi-signature with linear homomorphism. In: 2016 Symposium on Cryptography and Information Security (SCIS 2016)
23. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 711–742. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_24
24. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Four-round concurrent non-malleable commitments from one-way functions. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 127–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_5
25. Ciampi, M., Persiano, G., Scafuro, A., Siniscalchi, L., Visconti, I.: Improved OR-composition of Sigma-protocols. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 112–141. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_5
26. Cozzo, D., Smart, N.P.: Sharing the LUOV: threshold post-quantum signatures. In: Albrecht, M. (ed.) IMACC 2019. LNCS, vol. 11929, pp. 128–153. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35199-1_7
27. Dalskov, A., Keller, M., Orlandi, C., Shrishak, K., Shulman, H.: Securing DNSSEC keys via threshold ECDSA from generic MPC. Cryptology ePrint Archive, Report 2019/889
28. Damgård, I.: Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 418–430. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_30
29. Damgård, I., Jakobsen, T.P., Nielsen, J.B., Pagter, J.I., Østergård, M.B.: Fast threshold ECDSA with honest majority. Cryptology ePrint Archive, Report 2020/501

30. Damgård, I., Orlandi, C., Takahashi, A., Tibouchi, M.: Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. Cryptology ePrint Archive, Report 2020/1110
31. del Pino, R., Lyubashevsky, V., Seiler, G.: Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In: ACM CCS 2018. pp. 574–591. ACM Press (2018)
32. Doerner, J., Kondi, Y., Lee, E., shelat, A.: Secure two-party threshold ECDSA from ECDSA assumptions. In: 2018 IEEE Symposium on Security and Privacy, pp. 980–997. IEEE Computer Society Press (2018)
33. Doerner, J., Kondi, Y., Lee, E., Shelat, A.: Threshold ECDSA from ECDSA assumptions: the multiparty case. In: 2019 IEEE Symposium on Security and Privacy, pp. 1051–1066. IEEE Computer Society Press (2019)
34. Doröz, Y., Hoffstein, J., Silverman, J.H., Sunar, B.: MMSAT: a scheme for multimessage multiuser signature aggregation. Cryptology ePrint Archive, Report 2020/520
35. Drijvers, M., et al.: On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy, pp. 1084–1101. IEEE Computer Society Press (2019)
36. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_3
37. Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: digital signatures from module lattices
38. Ducas, L., Micciancio, D.: Improved short lattice signatures in the standard model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 335–352. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_19
39. Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. Cryptology ePrint Archive, Report 2020/845
40. Esgin, M.F., Steinfeld, R., Liu, J.K., Liu, D.: Lattice-based zero-knowledge proofs: new techniques for shorter and faster constructions and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 115–146. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_5
41. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 67–88. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_4
42. Fukumitsu, M., Hasegawa, S.: A tightly-secure lattice-based multisignature. In: APKC@AsiaCCS 2019, pp. 3–11. ACM (2019)
43. Fukumitsu, M., Hasegawa, S.: A lattice-based provably secure multisignature scheme in quantum random oracle model. In: Nguyen, K., Wu, W., Lam, K.Y., Wang, H. (eds.) ProvSec 2020. LNCS, vol. 12505, pp. 45–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62576-4_3
44. Gagol, A., Kula, J., Straszak, D., Swietek, M.: Threshold ECDSA for decentralized asset custody. Cryptology ePrint Archive, Report 2020/498
45. Genise, N., Micciancio, D.: Faster Gaussian sampling for trapdoor lattices with arbitrary modulus. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 174–203. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_7
46. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: ACM CCS 2018, pp. 1179–1194. ACM Press (2018)

47. Gennaro, R., Goldfeder, S.: One round threshold ECDSA with identifiable abort. Cryptology ePrint Archive, Report 2020/540
48. Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 156–174. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39555-5_9
49. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83
50. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th ACM STOC, pp. 197–206. ACM Press
51. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
52. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: 47th ACM STOC, pp. 469–477. ACM Press
53. Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: a signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_31
54. Howgrave-Graham, N., Joux, A.: New generic algorithms for hard knapsacks. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 235–256. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_12
55. Kansal, M., Dutta, R.: Round optimal secure multisignature schemes from lattice with public key aggregation and signature compression. In: Nitaj, A., Youssef, A. (eds.) AFRICACRYPT 2020. LNCS, vol. 12174, pp. 281–300. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51938-4_14
56. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 552–586. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_18
57. Komlo, C., Goldberg, I.: FROST: flexible round-optimized Schnorr threshold signatures. Cryptology ePrint Archive, Report 2020/852
58. Libert, B., Nguyen, K., Tan, B.H.M., Wang, H.: Zero-knowledge elementary databases with more expressive queries. In: Lin, D., Sako, K. (eds.) PKC 2019. LNCS, vol. 11442, pp. 255–285. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17253-4_9
59. Lindell, Y.: Fast secure two-party ECDSA signing. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 613–644. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_21
60. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: ACM CCS 2018, pp. 1837–1854. ACM Press (2018)
61. Liu, Z.Y., Tseng, Y.F., Tso, R.: Cryptanalysis of a round optimal lattice-based multisignature scheme. Cryptology ePrint Archive, Report 2020/1172
62. Lyubashevsky, V.: Fiat-Shamir with aborts: applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_35

63. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
64. Lyubashevsky, V.: Lattice-based zero-knowledge and applications. CIS 2019
65. Lyubashevsky, V., et al.: CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology
66. Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-LWE cryptography. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 35–54. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_3
67. Ma, C., Jiang, M.: Practical lattice-based multisignature schemes for blockchains. IEEE Access **7**, 179765–179778
68. Ma, C., Weng, J., Li, Y., Deng, R.H.: Efficient discrete logarithm based multi-signature scheme in the plain public key model. Des. Codes Cryptogr. **54**(2), 121–133. https://doi.org/10.1007/s10623-009-9313-z
69. Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple Schnorr multi-signatures with applications to Bitcoin. Des. Codes Crypt. **87**(9), 2139–2164 (2019). https://doi.org/10.1007/s10623-019-00608-x
70. Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: extended abstract. In: ACM CCS 2001, pp. 245–254. ACM Press (2001)
71. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
72. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_2
73. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: simple two-round Schnorr multi-signatures. Cryptology ePrint Archive, Report 2020/1261
74. Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. Cryptology ePrint Archive, Report 2020/1057
75. Nicolosi, A., Krohn, M.N., Dodis, Y., Mazières, D.: Proactive two-party signatures for user authentication. In: NDSS 2003. The Internet Society (2003)
76. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_19
77. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
78. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_5
79. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptol. **13**(3), 361–396
80. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_22
81. Stinson, D.R., Strobl, R.: Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In: Varadharajan, V., Mu, Y. (eds.) ACISP 2001. LNCS, vol. 2119, pp. 417–434. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47719-5_33

82. Syta, E., et al.: 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)
83. Toluee, R., Eghlidos, T.: An efficient and secure ID-based multi-proxy multi-signature scheme based on lattice. Cryptology ePrint Archive, Report 2019/1031
84. Torres, W.A., Steinfeld, R., Sakzad, A., Kuchta, V.: Post-quantum linkable ring signature enabling distributed authorised ring confidential transactions in blockchain. Cryptology ePrint Archive, Report 2020/1121
85. Tso, R., Liu, Z., Tseng, Y.: Identity-based blind multisignature from lattices. IEEE Access **7**, 182916–182923
86. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–304. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_19
87. Yang, R., Au, M.H., Zhang, Z., Xu, Q., Yu, Z., Whyte, W.: Efficient lattice-based zero-knowledge arguments with standard soundness: construction and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 147–175. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_6