Two-Round Threshold Lattice-Based Signatures from Threshold Homomorphic Encryption*

Kamil Doruk Gur[1], Jonathan Katz[2**], and Tjerand Silde[3***]

[1] University of Maryland, dgur1@cs.umd.edu
[2] Google, jkatz2@gmail.com
[3] Norwegian University of Science and Technology
tjerand.silde@ntnu.no

Abstract. Much recent work has developed efficient protocols for *threshold signatures*, where n parties share a signing key and some threshold t of those parties must interact to produce a signature. Yet efficient threshold signatures with post-quantum security have been elusive, with the state-of-the-art being a two-round scheme by Damgård et al. (PKC'21) based on lattices that supports only the full threshold case (i.e., $t = n$).

We show here a two-round threshold signature scheme based on standard lattice assumptions that supports arbitrary thresholds $t \leq n$. Estimates of our scheme's performance at the 128-bit security level show that in the 3-out-of-5 case, we obtain signatures of size 46.6 KB and public keys of size 13.6 KB. We achieve $\approx 5\times$ improved parameters if only a small number of signatures are ever issued with the same key.

As an essential building block and independent contribution, we construct an actively secure threshold (linearly) homomorphic encryption scheme that supports arbitrary thresholds $t \leq n$.

Keywords: Lattices · Threshold Signatures · Threshold Encryption

1 Introduction

In a *t-out-of-n threshold signature scheme*, a signing key is shared among n parties such that any t of those parties can jointly issue a signature. In contrast, an adversary corrupting strictly fewer than t of those parties cannot forge a signature. The past few years have witnessed remarkable progress in developing efficient protocols for threshold signatures. These efforts have been motivated largely by applications to cryptocurrency, with most attention being focused on threshold versions of ECDSA [GG18,LN18,DKLs19,CGG+20,CCL+20,DJN+20] and Schnorr-like schemes [KG20, Lin24, CGRS23]. Based in part on this level of interest, NIST has announced their intention [BP23] to standardize threshold cryptosystems.

* This is the full version of [GKS24] published at PQCrypto 2024.
** Work done in part at the University of Maryland and Dfns.
*** Work done in part while visiting the University of Maryland.

Efficient threshold signatures based on *post-quantum* hardness assumptions—and specifically lattice assumptions—have been elusive. While generic constructions are possible, they have drawbacks and/or are not particularly efficient. We survey other existing constructions in Section 1.3. The state-of-the-art is a recent construction by Damgård et al. [DOTT21] based on standard lattice assumptions that has a two-round signing protocol. Unfortunately, their solution only works for the full-threshold (i.e., $t = n$) setting and does not extend to the case of general thresholds $t \leq n$. (We discuss the challenges in adapting their technique to the case of general thresholds in Section 1.2.)

1.1 Our Contributions

We show a t-out-of-n threshold signature scheme based on standard lattice assumptions (Ring-LWE/SIS) that supports arbitrary thresholds $t \leq n$. Our scheme features a two-round signing protocol and allows for efficient distributed key generation. Estimates of our scheme's performance at the 128-bit security level show that in the 3-out-of-5 case, we obtain signatures of size 46.6 KB and public keys of size 13.6 KB. We can also reduce the signature size by up to a factor of 5× in settings where the number of signatures generated using a single key is bounded in advance; we refer to Section 7 for further details.

Our scheme is based on a general framework for constructing threshold signatures from a (linearly) homomorphic encryption scheme with threshold decryption. Although the particular instantiation we propose is based on a variant of the Dilithium signature scheme, our framework is general enough to be instantiated using other schemes in the future.

As an essential building block of independent interest, we show a new and actively secure t-out-of-n threshold (linearly) homomorphic encryption scheme. Our construction is based on the BGV encryption scheme [BGV12] combined with (verifiable) Shamir secret sharing and lattice-based zero-knowledge proofs. In contrast to prior work of Boneh et al. [BGG+18], we make the standard assumption that the set of users involved in decryption is known in advance; this allows us to achieve better parameters than in their case.

1.2 Technical Overview

We begin with a high-level overview of the approach used by Damgård et al. [DOTT21] to construct n-out-of-n lattice-based threshold signatures, and explain why their scheme does not generalize easily to the t-out-of-n case. We then describe the key ideas underlying our scheme. Several technical details are omitted since this is intended only to provide intuition.

We first describe a three-message identification scheme based on lattices inspired by the Schnorr identification scheme in the discrete-logarithm setting. Fix a ring R_q. The prover's private key is a short vector $\mathbf{s} \in R_q^{\ell+k}$, and its public key consists of a matrix $\bar{\mathbf{A}} := [\mathbf{A} \,|\, \mathbf{I}] \in R_q^{k \times (\ell+k)}$ (where $\mathbf{A}$ is uniform) and the vector $\boldsymbol{y} := \bar{\mathbf{A}}\mathbf{s}$. Execution of the protocol proceeds as follows:

1. The prover samples a short vector $\boldsymbol{r} \in R_q^{\ell+k}$ and sends $\boldsymbol{w} := \bar{\mathbf{A}}\boldsymbol{r}$.
2. The verifier responds with a short challenge $c \in R_q$.
3. The prover responds with short vector $\boldsymbol{z} := c \cdot \mathbf{s} + \boldsymbol{r}$.
4. The verifier accepts iff $\boldsymbol{z}$ is short and $\bar{\mathbf{A}}\boldsymbol{z} = c \cdot \boldsymbol{y} + \boldsymbol{w}$.

Although this protocol can be shown to be *sound*, in general it is not *honest-verifier zero knowledge* (HVZK). One way to address this is by allowing the prover to *abort* [Lyu12]. Specifically, step 3 is modified so the prover only responds if a certain condition holds, but aborts (and returns to step 1) otherwise. It can be shown that an execution of such a modified protocol is HVZK *conditioned on the event that an abort does not occur*. While this is insufficient to prove security of the above as an interactive protocol (since information may be leaked in executions where the prover aborts), it suffices when the Fiat-Shamir transform is applied to the above protocol[1] to derive a signature scheme (since the prover/signer will then never release transcripts from aborted executions). We refer to the latter approach as *Fiat-Shamir with aborts* (FSwA).

Another way to make the protocol HVZK, without introducing aborts, is to use *noise drowning* [GKPV10]. We use Rényi divergence [ASY22] to analyze the extra noise needed for security based on the number of allowed signing queries. When coupled with the Fiat-Shamir transform, this results in larger signatures than the FSwA approach but can lead to better computational efficiency. It can also benefit the threshold setting, where interaction is inherent.

Damgård et al. propose a way to distribute the FSwA version of the above scheme among n signers based on the following idea: the ith signer holds short vector $\mathbf{s}_i$ and $\mathbf{s} = \sum_{i\in[n]} \mathbf{s}_i$ is the private key. Then, the n signers can run a distributed, two-round signing protocol as follows:

1. The ith signer chooses a short vector $\boldsymbol{r}_i \in R_q^{\ell+k}$ and sends $\boldsymbol{w}_i := \bar{\mathbf{A}}\boldsymbol{r}_i$.
2. Each signer computes $\boldsymbol{w} := \sum_{i\in[n]} \boldsymbol{w}_i$ followed by $c := H(\boldsymbol{w})$. The ith signer then sends $\boldsymbol{z}_i := c \cdot \mathbf{s}_i + \boldsymbol{r}_i$.
3. Each signer then computes $\boldsymbol{z} := \sum_{i\in[n]} \boldsymbol{z}_i$ and outputs the signature $(c, \boldsymbol{z})$.

We stress that the above does *not* work directly since it does not consider the possibility that one or more of the honest signers will need to abort. Moreover, incorporating aborts in the trivial way (namely, by restarting the protocol if any of the signers abort) may not be secure since the initial message $\boldsymbol{w}_i$ of the ith signer is revealed even if that signer later aborts and, as we have noted above, aborted executions of the underlying identification protocol are not HVZK. To address this, Damgård et al. modify the above so that each signer sends a (trapdoor) homomorphic *commitment* to $\boldsymbol{w}_i$ in the first round; thus, $\boldsymbol{w}_i$ is not revealed if the ith signer aborts. We omit further details, as they are not necessary to understand the difficulties in extending this approach to the t-out-of-n case.

A natural way to try to extend the approach of Damgård et al. to the case of general thresholds is to share the master secret $\mathbf{s}$ among the n parties in a

[1] Applying the Fiat-Shamir transform means that the challenge c is computed as a hash of the initial message $\boldsymbol{w}$ and possibly other information.

t-out-of-n fashion using, e.g., Shamir secret sharing. The problem with this idea, however, is that we need both the master secret $\mathbf{s}$ and each party's share $\mathbf{s}_i$ to be short, and it is not clear whether this can be achieved when using t-out-of-n secret sharing. (In contrast, this is easy to achieve in the n-out-of-n case since the sum of n short vectors is still short.) Note that the $\{\mathbf{s}_i\}$ need to be short regardless of whether one uses the FSwA approach or noise flooding (without aborts): in the former case, if any $\mathbf{s}_i$ is too large, the corresponding signer will abort too often; in the latter case, achieving HVZK with a large $\mathbf{s}_i$ would require parameters that are too large to be secure.

Here, we adopt an approach that relies on a threshold (linearly) homomorphic encryption scheme with non-interactive decryption. We describe the idea based on a generic scheme and show in Section 3 an instantiation based on standard lattice assumptions. We build on a version of the identification protocol described above that uses larger parameters and does not require aborts. The signing key $\mathbf{s}$ and verification key $(\bar{\mathbf{A}}, \boldsymbol{y} := \bar{\mathbf{A}}\mathbf{s})$ of the signature scheme will be as before. Now, however, instead of sharing $\mathbf{s}$ itself, the signers each hold an encryption $\mathsf{ctx}_{\mathbf{s}} = \mathsf{Enc}(\mathbf{s})$ of $\mathbf{s}$ with respect to a known public encryption key pk, and share the corresponding decryption key sk in a t-out-of-n fashion. Any set $\mathcal{U} \subseteq [n]$ of t parties can generate a signature (in the semi-honest setting) as follows:

1. For $i \in \mathcal{U}$, the ith signer chooses a bounded vector $\boldsymbol{r}_i \in R_q^{\ell+k}$ and sends $\boldsymbol{w}_i := \bar{\mathbf{A}}\boldsymbol{r}_i$. It also sends $\mathsf{ctx}_{\boldsymbol{r}_i}$, an encryption of $\boldsymbol{r}_i$.
2. Each signer in $\mathcal{U}$ locally computes $\boldsymbol{w} := \sum_{i \in \mathcal{U}} \boldsymbol{w}_i$, $c := H(\boldsymbol{w})$, and an "encrypted (partial) signature" $\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{i \in \mathcal{U}} \mathsf{ctx}_{\boldsymbol{r}_i}$. The ith signer then sends its threshold decryption share of $\mathsf{ctx}_{\boldsymbol{z}}$.
3. Given decryption shares from all parties in $\mathcal{U}$, each signer can decrypt $\mathsf{ctx}_{\boldsymbol{z}}$ to obtain $\boldsymbol{z}$, and output the signature $(c, \boldsymbol{z})$.

The key insight is that while we cannot use t-out-of-n secret sharing for the signing key due to the required size bounds, we *can* use it for the threshold decryption key since decryption shares can be large.

While the above is secure for semi-honest adversaries, additional work is needed to handle malicious adversaries while achieving a two-round signing protocol. We refer to Section 5 for further details.

1.3 Related Work

Lattice-based threshold signature schemes. Bendlin et al. [BKP13] show a threshold version of the (hash-and-sign based) GPV scheme [GPV08]. Their protocol uses generic secure multiparty computation to distributively compute the most expensive part of the scheme (namely, Gaussian sampling [Pei10]), and seems unlikely to yield a practical solution; moreover, their scheme requires and honest majority. Cozzo and Smart [CS19] and Tang et al. [TPCZ23] explored the use of generic secure multiparty computation to construct threshold versions of several signature schemes submitted to the NIST post-quantum standardization process but concluded that this approach is unlikely to yield practical protocols.

Boneh et al. [BGG+18] show a "universal thresholdizer" that can be used to create a threshold version of any signature scheme. The basic idea behind their framework is to encrypt the master private key of the underlying signature scheme using a threshold fully homomorphic encryption (FHE) scheme, evaluate the underlying scheme homomorphically, and then use threshold decryption to recover the signature. Agrawal et al. [ASY22] adapted this approach to the specific signature scheme Dilithium-G [DKL+18] and showed how to tolerate adaptive corruptions. Our approach is similar in spirit to these approaches, but by moving as many signature steps as possible outside the homomorphic evaluation we can base our protocol on threshold *linearly* homomorphic—rather than *fully* homomorphic—encryption; besides the efficiency advantages this confers, this also allows us to distribute key generation (something not achieved in [ASY22, BGG+18]).

We have already mentioned the work of Damgård et al. [DOTT21] showing an efficient n-out-of-n threshold scheme based on lattices, and explained why it does not readily extend to give a t-out-of-n scheme. For completeness, we remark that there has recently been extensive work on lattice-based multisignatures [FH20, DOTT21, BTT22, FSZ22, Che23] which are related to—but distinct from—n-out-of-n threshold signatures. The schemes of Boschini et al. [BTT22] and Chen [Che23] can be turned into n-out-of-n threshold schemes. Unfortunately, as with the scheme by Damgård et al., it seems difficult to adapt their schemes to support arbitrary thresholds.

Threshold homomorphic encryption. Bendlin and Damgård [BD10] show a threshold homomorphic encryption scheme with semi-honest security. The scheme we construct is based on the work of Aranha et al. [ABGS23] and Hough et al. [HSS23], which achieve malicious security for the Bendlin-Damgård scheme. Although the Bendlin-Damgård scheme supports arbitrary thresholds, the subsequent works only support the full threshold case.

The threshold FHE scheme of Boneh et al. [BGG+18] lacks an efficient mechanism for proving correctness of partial decryptions, which is needed for handling malicious behavior. Recent work by Boudgoust and Scholl [BS23] is similar to our threshold encryption scheme but has the same drawback; moreover, their scheme only achieves one-way security.

Chowdhury et al. [CSS+22] show a threshold fully homomorphic encryption scheme with trusted setup and semi-honest security; the size of their key shares is exponential in the number of parties n. Devevey et al. [DLN+21] also construct a lattice-based threshold encryption scheme, but this scheme also relies on a trusted dealer. Rotaru et al. [RST+22] give an actively secure distributed key-generation protocol for lattice-based threshold encryption, but only for the full threshold case. Dahl et al. [DDEK+23] present an actively secure threshold FHE scheme in the honest-majority setting. They rely on generic MPC in an offline phase.

Concurrent work. Concurrent with or subsequent to our own work, several other lattice-based threshold signature schemes supporting general thresholds

have been proposed [dPKM+24,EKT24,CATZ24]. In contrast to our work, none of these support distributed key generation.

2 Background

Let N be a power of 2, and let $q = 1 \bmod 2N$ be prime. Define the rings $R = \mathbb{Z}[X]/\langle X^N + 1\rangle$ and $R_q = \mathbb{Z}_q[X]/\langle X^N + 1\rangle$. For $f(X) = \sum_{i=0}^{N-1} \alpha_i X^i \in R_q$, we compute norms of f by viewing each $\alpha_i \in \mathbb{Z}_q$ as an integer in the range $\left\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\right\}$ and then viewing f as a vector over $\mathbb{Z}$; thus,

$$\|f\|_1 = \sum_{i=0}^{N-1} |\alpha_i|, \qquad \|f\|_2 = \left(\sum_{i=0}^{N-1} \alpha_i^2\right)^{1/2}, \qquad \|f\|_\infty = \max_{i\in\{0,\ldots,N-1\}} \{|\alpha_i|\},$$

with $\alpha_i \in \left\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\right\}$. We define the norm of a vector $\boldsymbol{f} \in R_q^k$ to be the largest norm of any of its elements, e.g., $\|\boldsymbol{f}\|_1 = \max_{i\in\{1,\ldots,k\}} \{\|f_i\|_1\}$. All vectors are column vectors by default; thus, a row vector is written as the transpose of a column vector. We use the standard definition of the discrete Gaussian distribution $D_{\boldsymbol{v},\bar{\sigma}}$ over the integer lattice $\Lambda = \mathbb{Z}^k$, with center $\boldsymbol{v} \in \mathbb{R}^k$ and standard deviation $\bar{\sigma}$. If $\boldsymbol{v} = \boldsymbol{0}$, we omit the first subscript.

Shamir secret sharing [Sha79] of elements in R_q^ℓ can be done by independently applying standard Shamir secret sharing (over the field $\mathbb{Z}_q$) to each of the $\ell \cdot q$ coefficients of the polynomials. We use $\{x_i\}_{i\in[n]} \leftarrow \mathsf{Share}_{t,n}(x)$ to denote t-out-of-n Shamir secret sharing of coefficients of an element x, and write $x := \mathsf{Rec}_{t,n}(\{x_i\}_{i\in\mathcal{U}})$ for reconstruction (where $\mathcal{U} \subseteq [n]$ has size t).

2.1 Cryptographic Assumptions

We define the *Ring Short Integer Solution* [Mic02] and *Ring Learning With Errors* [LPR10] problems as follows:

Definition 1 (R-SIS)**.** *The Ring Short Integer Solution problem* $\mathsf{R\text{-}SIS}_{k,N,q,\beta}$ *is* (t,ϵ)-hard *if for any adversary* $\mathcal{A}$ *running in time at most* t*:*

$$\Pr\left[\begin{array}{c}\{a_i\}_{i\in[k]} \leftarrow R_q; \\ \{y_i\}_{i\in[k]} \leftarrow \mathcal{A}(\{a_i\})\end{array} : \begin{array}{c}0 < \|\boldsymbol{y}\|_2 \le \beta \wedge \\ \sum_{i\in[k]} a_i y_i = 0 \bmod q\end{array}\right] \le \epsilon.$$

Definition 2 (R-LWE)**.** *Let* χ *be a bounded distribution over* R_q *outputting element of maximum absolute norm* β*. The Ring Learning with Errors problem* $\mathsf{R\text{-}LWE}_{k,N,q,\beta}$ *is* (t,ϵ)-hard *if for any adversary* $\mathcal{A}$ *running in time at most* t*:*

$$\begin{aligned}&\big|\Pr[\{a_i\}_{i\in[k]} \leftarrow R_q; s, \{e_i\}_{i\in[k]} \leftarrow \chi : \mathcal{A}(\{a_i\}, \{a_i s + e_i\}) = 1] \\ &\quad - \Pr[\{a_i\}_{i\in[k]}, \{u_i\}_{i\in[k]} \leftarrow R_q : \mathcal{A}(\{a_i\}, \{u_i\}) = 1]\big| \le \epsilon.\end{aligned}$$

The above assumption also implies that distinguishing uniform u from $a_i s + p e_i$ is also (t,ϵ)-hard if public $p \in \mathbb{Z}_q$ is relatively prime to q [BGV12].

2.2 Homomorphic Trapdoor Commitment Schemes

Let cpp denote fixed parameters implicit input to all algorithms. Following the definition in [DOTT21], a trapdoor commitment scheme is a tuple of probabilistic polynomial-time algorithms $(\mathsf{CGen}, \mathsf{Com}, \mathsf{Open}, \mathsf{TCGen}, \mathsf{TCom}, \mathsf{Eqv})$ where

- CGen outputs a commitment key ck.
- Com takes as input a message $\mu \in R_q$, samples randomness $\rho \in S_\rho$ according to distribution D_ρ, and outputs a commitment com.
- Open takes as input a commitment com, message $\mu \in R_q$, and randomness $\rho \in S_\rho$, and outputs 1 iff the opening is valid.
- TCGen outputs a trapdoor commitment key ck along with a trapdoor td.
- TCom takes as input a trapdoor td and outputs a commitment com.
- Eqv takes as input a trapdoor td, a commitment com, and a message μ, and outputs randomness $\rho \in S_\rho$.

The trapdoor commitment scheme is secure if it satisfies the following:

Correctness: The scheme is *correct* if for any $\mu \in R_q$

$$\Pr\left[\begin{array}{c}\mathsf{ck} \leftarrow \mathsf{CGen};\\ \rho \leftarrow D_\rho; \mathsf{com} \leftarrow \mathsf{Com}(\mu;\rho)\end{array} : \mathsf{Open}(\mathsf{com}, \mu, \rho) = 1\right] = 1.$$

Hiding: The scheme is (t, ϵ)-*hiding* if for any adversary $\mathcal{A}$ running in time at most t:

$$\Pr\left[\begin{array}{c}\mathsf{ck} \leftarrow \mathsf{CGen}; b \leftarrow \{0,1\}; \rho \leftarrow D_\rho;\\ (\mu_0, \mu_1) \leftarrow \mathcal{A}(\mathsf{cpp}, \mathsf{ck}); \mathsf{com} \leftarrow \mathsf{Com}(\mu_b;\rho)\end{array} : \mathcal{A}(\mathsf{com}) = b\right] \leq \frac{1}{2} + \epsilon.$$

Binding: The scheme is (t, ϵ)-*binding* if for any adversary $\mathcal{A}$ running in time at most t:

$$\Pr\left[\begin{array}{c}\mathsf{ck} \leftarrow \mathsf{CGen};\\ (\mathsf{com}, \mu_0, \rho_0, \mu_1, \rho_1) \leftarrow \mathcal{A}(\mathsf{ck})\end{array} : \begin{array}{c}\mu_0 \neq \mu_1\\ \wedge\ \mathsf{Open}(\mathsf{com}, \mu_0, \rho_0) = 1\\ \wedge\ \mathsf{Open}(\mathsf{com}, \mu_1, \rho_1) = 1\end{array}\right] \leq \epsilon.$$

Equivocality: The scheme is ϵ-*equivocal* if for any $\mu \in R_q$, the statistical difference between the following distributions is at most ϵ:

$$\left\{\begin{array}{c}\mathsf{ck} \leftarrow \mathsf{CGen};\\ \rho \leftarrow D_\rho; \mathsf{com} \leftarrow \mathsf{Com}(\mu;\rho)\end{array} : (\mathsf{ck}, \mu, \mathsf{com}, \rho)\right\}$$

$$\left\{\begin{array}{c}(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TCGen};\\ \mathsf{com} \leftarrow \mathsf{TCom}(\mathsf{td}); \rho \leftarrow \mathsf{Eqv}(\mathsf{td}, \mathsf{com}, \mu)\end{array} : (\mathsf{ck}, \mu, \mathsf{com}, \rho)\right\}.$$

A trapdoor commitment scheme over polynomial rings can be constructed based on $\mathsf{R\text{-}SIS}$ and $\mathsf{R\text{-}LWE}$ [DOTT21, GPV08, MP12].

2.3 Non-Interactive Zero-Knowledge Proofs of Knowledge

We use standard definitions of non-interactive zero-knowledge proofs of knowledge (NIZKPoKs). An NIZKPoK for an NP language $\mathcal{L}$ with associated relation $\mathcal{R}_{\mathcal{L}}$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{SetupTD}, \mathsf{Prove}, \mathsf{Vrfy}, \mathsf{Extract}, \mathsf{Sim})$, where:

- Setup outputs a common reference string crs.
- $\mathsf{SetupTD}$ outputs a common reference string crs and a trapdoor td.
- Prove, on input crs and $(x, w) \in \mathcal{R}_{\mathcal{L}}$, outputs a proof π.
- Vrfy, on input crs, x and π, outputs a result $b \in \{0, 1\}$.
- $\mathsf{Extract}$, on input $\mathsf{crs}, \mathsf{td}, x$ and π^*, outputs a witness w^*.
- Sim, on input $\mathsf{crs}, \mathsf{td}$ and x, outputs a proof π^*.

A NIZKPoK is *secure* if the following properties hold:

Correctness: The scheme is *correct* if for all $(x, w) \in R_{\mathcal{L}}$ we have

$$\Pr\left[\begin{array}{c}\mathsf{crs} \leftarrow \mathsf{Setup} \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{array} : \mathsf{Vrfy}(\mathsf{crs}, x, \pi) = 1\right] = 1.$$

Knowledge extraction: The scheme is (t, ϵ)-*knowledge extractable* if for all malicious provers P^* we have an extractor $\mathcal{E}$ that in time t can, except with probability ϵ, extract a valid witness by running $\mathsf{Extract}$:

$$\Pr\left[\begin{array}{c}(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupTD}; \\ \perp \neq \pi^* \leftarrow \mathsf{P}^*(\mathsf{crs}, x, w); \\ w^* \leftarrow \mathsf{Extract}(\mathsf{crs}, \mathsf{td}, x, \pi^*);\end{array} : \begin{array}{c}\mathsf{Vrfy}(\mathsf{crs}, x, \pi^*) = 1 \\ \wedge\ (x, w^*) \notin \mathcal{R}_{\mathcal{L}}\end{array}\right] \leq \epsilon.$$

Zero knowledge: The scheme is (t, ϵ)-*(computational) zero-knowledge* if there exists a simulator Sim that in t time can simulate a proof by running Sim:

$$\begin{aligned}&|\Pr[\mathsf{crs} \leftarrow \mathsf{Setup}; \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) : \mathcal{A}(\mathsf{crs}, x, \pi) = 1] \\ &- \Pr[(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{SetupTD}; \pi \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathsf{td}, x) : \mathcal{A}(\mathsf{crs}, x, \pi) = 1]| \leq \epsilon.\end{aligned}$$

We will make use of the lattice-based zero-knowledge profs of linear relations and shortness by Lyubashevsky et al. [BDL+18, BLNS21, LNP22] in this paper, and, for concurrent security, we need zero-knowledge proofs with straight-line extractability; this can be achieved using, e.g., the transform by Katsumata [Kat21].

2.4 Threshold Signatures

A threshold signature scheme $\mathcal{TS}$ (adapting [Lin17, Section 4] and [DOTT21, Definition 5]) consists of the following algorithms:

- $\mathsf{KGen}_{\mathcal{TS}}$ is an interactive protocol run by n users that takes as input n and a threshold t. Each user either aborts or outputs a (common) public key pk, (common) auxiliary data aux, and a secret key share sk_i.

– $\mathsf{Sign}_{\mathcal{TS}}$ is an interactive protocol run by a set $\mathcal{U}$ of t users. Each party begins holding their secret key share, auxiliary data aux, and a message μ. Each user either aborts or outputs a signature σ.
– $\mathsf{Vrfy}_{\mathcal{TS}}$ takes as input the public key pk, a message μ, and a signature σ, and outputs 1 iff the signature is valid.

Correctness is defined in the natural way. We can consider unforgeability against either a passive (aka semi-honest) or an active (aka malicious) adversary. In either case, we consider a static corruption model in which the adversary $\mathcal{A}$ starts by corrupting a set $\mathcal{C} \subset [n]$ of up to $t-1$ users. Let $\mathcal{H} = [n] \setminus \mathcal{C}$ denote the honest users. In the *passive* setting, the attacker is given the view of the corrupted parties from the execution of the key-generation protocol; in the *malicious* setting, the attacker runs an execution of the key-generation protocol with the honest parties in which the corrupted parties can behave arbitrarily. (Here and below, we assume a *rushing* adversary who can wait to receive honest parties' messages in any given round before sending any of the corrupted parties' messages.) Following key generation, $\mathcal{A}$ can repeatedly make *signing queries* in which it specifies a message μ and a set of t users $\mathcal{U}$, and thereby initiate an execution of the signing protocol with those users holding message μ. Let $\mathcal{C}_{\mathcal{U}} = \mathcal{U} \cap \mathcal{C}$, and $\mathcal{H}_{\mathcal{U}} = \mathcal{U} \cap \mathcal{H}$. In the passive case, $\mathcal{A}$ is given the view of the parties in $\mathcal{C}_{\mathcal{U}}$; in the malicious case, $\mathcal{A}$ runs an execution of the signing protocol with the parties in $\mathcal{H}_{\mathcal{U}}$ in which parties in $\mathcal{C}_{\mathcal{U}}$ can behave arbitrarily.

At the end of the experiment, $\mathcal{A}$ outputs a message/signature pair (μ^*, σ^*). The adversary succeeds if μ^* was never used in one of $\mathcal{A}$'s signing queries, and σ^* is a valid signature on μ^* with respect to the common public key output by [2] the honest parties in the key-generation protocol. We let $\mathsf{Adv}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{TS}}(\mathcal{A})$ denote the probability with which adversary $\mathcal{A}$ succeeds when attacking $\mathcal{TS}$.

3 Threshold Homomorphic Encryption

Our threshold signature scheme relies on an underlying threshold linearly homomorphic encryption scheme. We define the required security properties formally and show how to instantiate a scheme based on BGV [BGV12]. We specialize our definition for schemes with non-interactive decryption.

3.1 Definitions

A homomorphic encryption scheme $\mathcal{E}$, given a set of possible circuits $\mathcal{F}$, consists of the following algorithms:

– $\mathsf{KGen}_{\mathcal{E}}$ is a probabilistic algorithm that outputs a public encryption key $\mathsf{pk}_{\mathcal{E}}$ and a decryption key $\mathsf{sk}_{\mathcal{E}}$.

[2] In particular, if all honest parties abort the key-generation protocol (in the malicious setting) then there is no public key and, by definition, $\mathcal{A}$ cannot succeed.

- Enc is a probabilistic algorithm that takes as input a public key $\mathsf{pk}_{\mathcal{E}}$ and a plaintext ptx, and outputs a ciphertext ctx.
- Eval is a deterministic algorithm that takes as input a circuit $F \in \mathcal{F}$ and a list of ciphertexts $\mathsf{ctx}_1, \ldots, \mathsf{ctx}_k$, and outputs a ciphertext ctx^*.
- Dec is a deterministic algorithm that takes as input a decryption key $\mathsf{sk}_{\mathcal{E}}$ and a ciphertext ctx^*, and outputs a plaintext ptx.

Notationally, we allow algorithms to take as inputs a list of plaintexts, ciphertexts, or keys to denote that they are applied on each input individually. Correctness and ciphertext indistinguishability (i.e., IND-CPA security) are standard, and thus we omit the definitions here.

We now extend the above to accommodate $\mathcal{E}$ with distributed key generation and (non-interactive) threshold decryption:

- DKGen is an interactive protocol run by n users, on common input a threshold t. Either all honest users output $\perp$, or they each output a decryption key share sk_i and a (common) public key $\mathsf{pk}_{\mathcal{E}}$.
- TDec is a probabilistic algorithm that takes as input a set $\mathcal{U}$ consisting of t users, a decryption key share sk_i, and a ciphertext ctx^*, and outputs a partial decryption share ds_i.
- Comb is a deterministic algorithm that takes as input a set $\mathcal{U}$ consisting of t users, a ciphertext ctx^*, and a set of decryption shares $\{\mathsf{ds}_i\}_{i \in \mathcal{U}}$, and outputs a plaintext ptx.

We define security in the threshold setting as follows:

Indistinguishability. $\mathcal{E}$ is (t, ϵ)-threshold indistinguishable if the probability of any adversary $\mathcal{A}$ running in time t succeeding in experiment $\mathsf{Exp}_{\mathcal{E}}^{\text{IND-CPA}}(\mathcal{A})$ as depicted in Figure 1 is at most $1/2 + \epsilon$.

$\mathsf{Exp}_{\mathcal{E}}^{\text{IND-CPA}}(\mathcal{A})$

1. $\mathcal{A}$ outputs a set of corrupted users $\mathcal{C}$ of size at most $t-1$. Let $\mathcal{H} := [n] \backslash \mathcal{C}$.
2. $\mathcal{A}$ (controlling parties in $\mathcal{C}$) runs DKGen with the parties in $\mathcal{H}$. The view of the honest parties defines a public key $\mathsf{pk}_{\mathcal{E}}$ as well as keys $\{\mathsf{sk}_i\}_{i \in \mathcal{H}}$.
3. At any time during the experiment, $\mathcal{A}$ may make a "threshold decryption query" of the form $(\mathcal{U}, \{\mathsf{ptx}_j, \rho_j\}, F \in \mathcal{F})$, answered as follows:
   - Compute ciphertexts $\mathsf{ctx}_j := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathsf{ptx}_j; \rho_j)$ for all j.
   - Evaluate the function F on $\{\mathsf{ctx}_j\}$ to get ciphertext ctx^*.
   - Let $\mathcal{H}_{\mathcal{U}} := \mathcal{H} \cap \mathcal{U}$. For $i \in \mathcal{H}_{\mathcal{U}}$, compute $\mathsf{ds}_i \leftarrow \mathsf{TDec}(\mathcal{U}, \mathsf{sk}_i, \mathsf{ctx}^*)$.
   - Return decryption shares $\{\mathsf{ds}_i\}_{i \in \mathcal{H}_{\mathcal{U}}}$ to $\mathcal{A}$.
4. At some point, $\mathcal{A}$ outputs plaintexts ptx_0 and ptx_1. Sample $b \leftarrow \{0, 1\}$, compute $\mathsf{ctx}_b := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathsf{ptx}_b)$, and give ctx_b to $\mathcal{A}$.
5. $\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$.

Fig. 1. Experiment $\mathsf{Exp}_{\mathcal{E}}^{\text{IND-CPA}}(\mathcal{A})$ implicitly parameterized by t and n.

3.2 The BGV Encryption Scheme

Our threshold scheme is based on the BGV encryption scheme [BGV12], which we review now. Let $p \ll q$ be prime, let R_q and R_p be as in Section 2 (for the same dimension N), and let D_{KGen} and D_{Enc} be distributions over R_q such that elements in their support have ℓ_∞-norm bounded by B_{KGen} and B_{Enc}, respectively. The BGV encryption scheme consists of the following algorithms:

- $\mathsf{KGen}_{\mathsf{BGV}}$: Sample a uniform element $a_{\mathcal{E}} \in R_q$ along with $s, e \leftarrow D_{\mathsf{KGen}}$, and output public key $\mathsf{pk}_{\mathcal{E}} := (a_{\mathcal{E}}, b_{\mathcal{E}}) = (a_{\mathcal{E}}, a_{\mathcal{E}} s + pe)$ and secret key $\mathsf{sk}_{\mathcal{E}} := s$.
- $\mathsf{Enc}_{\mathsf{BGV}}$: On input $\mathsf{pk}_{\mathcal{E}}$ and message $\mathsf{ptx} := m \in R_p$, sample $r, e', e'' \leftarrow D_{\mathsf{Enc}}$ and output ciphertext $\mathsf{ctx} := (u, v) = (a_{\mathcal{E}} r + pe', b_{\mathcal{E}} r + pe'' + m)$.
- $\mathsf{Dec}_{\mathsf{BGV}}$: On input the secret key $\mathsf{sk}_{\mathcal{E}}$ and ciphertext ctx, output the plaintext message $\mathsf{ptx} := m = (v - su \bmod q) \bmod p$.

Correctness and IND-CPA security follows from [BGV12, LPR13]:

Lemma 1 (Security). *Let $B_{\mathsf{Dec}} = 2pB_{\mathsf{KGen}}B_{\mathsf{Enc}} + pB_{\mathsf{Enc}}$ be such that $B_{\mathsf{Dec}} < \lfloor q/2 \rfloor$. Let $(\mathsf{pk}_{\mathcal{E}}, \mathsf{sk}_{\mathcal{E}})$ be any key pair output from $\mathsf{KGen}_{\mathsf{BGV}}$ and let (u, v) be any ciphertext output from $\mathsf{Enc}_{\mathsf{BGV}}$ on input $\mathsf{pk}_{\mathcal{E}}$ and any message m. Then the BGV encryption scheme is perfectly correct, that is, $m = \mathsf{Dec}_{\mathsf{BGV}}(\mathsf{sk}_{\mathcal{E}}, (u, v))$. Furthermore, it is* IND-CPA *secure if $\mathsf{R\text{-}LWE}_{N,q,B_{\mathsf{KGen}}}$ and $\mathsf{R\text{-}LWE}_{N,q,B_{\mathsf{Dec}}}$ are hard.*

3.3 Distributed Key Generation for BGV

We propose a t-out-of-n distributed key-generation protocol for the BGV encryption scheme. For simplicity, we first describe a semi-honest version of the protocol, and then discuss how to add appropriate zero-knowledge proofs to ensure security against active adversaries.

In what follows, let χ_{tern} be the distribution over R_q where each coefficient a is sampled independently from $\{-1, 0, 1\}$ with $\Pr[a = 0] = 1/2$ and $\Pr[a = 1] = \Pr[a = -1] = 1/4$. We assume all parties $\mathcal{P}_i$ agree on a uniform ring element $a_{\mathcal{E}}$ which could be taken to be the output of a random oracle on some session ID or nonce. The semi-honest key-generation protocol proceeds as follows:

1. $\mathcal{P}_i$ samples $s_i, e_i \leftarrow \chi_{\mathsf{tern}}$, and computes $b_i := a_{\mathcal{E}} s_i + pe_i$. It then broadcasts b_i to all other parties.
2. $\mathcal{P}_i$ generates t-out-of-n Shamir secret shares $\{s_{i,j}\}_{j \in [n]}$ of s_i, and sends $s_{i,j}$ to $\mathcal{P}_j$ over a private channel as $[s_{i,j}]$.
3. $\mathcal{P}_i$ computes $b_{\mathcal{E}} := \sum b_j$ and outputs $\mathsf{pk} = (a_{\mathcal{E}}, b_{\mathcal{E}})$ and $\mathsf{sk}_i = \sum_j s_{j,i}$.

If we let D_{KGen} be the distribution over R_q obtained by summing n independent samples from χ_{tern}, it is clear that the above generates BGV encryption keys according to the distribution D_{KGen}.

For security against a malicious adversary, we use a commit-and-open approach so corrupted parties cannot choose their contributions based on honest parties' contributions, and add ZK proofs of correctness. Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a hash function. The modified protocol proceeds as follows:

1. $\mathcal{P}_i$ samples $s_i, e_i \leftarrow \chi_{\mathsf{tern}}$, computes $b_i := a_{\mathcal{E}} s_i + p e_i$, and then computes $h_i := H(i, b_i)$. It broadcasts h_i.
2. $\mathcal{P}_i$ broadcasts b_i, generates t-out-of-n Shamir secret shares $\{s_{i,j}\}_{j\in[n]}$ (resp., $\{e_{i,j}\}_{j\in[n]}$) of s_i (resp., e_i), and sets $b_{i,j} := a_{\mathcal{E}} s_{i,j} + p e_{i,j}$ for $j \in [n]$.
3. $\mathcal{P}_i$ samples a commitment randomness $\rho_{i,j} \in S_\rho$ and commits to $s_{i,j}$ as $\mathsf{com}_{i,j} := \mathsf{Com}(s_{i,j}; \rho_{i,j})$. It broadcasts $\{b_{i,j}, \mathsf{com}_{i,j}\}_{j\in[n]}$ and sends to $\mathcal{P}_j$ (over a private channel) the values $[s_{i,j}, e_{i,j}, \rho_{i,j}]$. $\mathcal{P}_i$ also gives an NIZK proof of knowledge of values $s_i, e_i, \{s_{i,j}, e_{i,j}\}_{j\in[n]}$ such that (1) s_i, e_i are in the support of χ_{tern}, (2) the $\{s_{i,j}\}_{j\in[n]}$ (resp., $\{e_{i,j}\}_{j\in[n]}$) are a correct t-out-of-n secret sharing of s_i (resp., e_i), and (3) the broadcasted values are consistent with a correct execution using $s_i, e_i, \{s_{i,j}, e_{i,j}\}_{j\in[n]}$ (see Section 6 details on the proofs).
4. $\mathcal{P}_i$ checks that the following hold for all $j \neq i$ (and aborts if not): (1) $h_j = H(j, b_j)$; (2) $\mathsf{com}_{j,i}$ has a correct opening with respect to $s_{j,i}$ and $\rho_{i,j}$ (3) the NIZK proof given by $\mathcal{P}_j$ verifies; and (4) the $\{b_{j,k}\}_{k\in[n]}$ are a correct secret sharing of a value b_j, and locally check the same for the sharings of s_j and e_j. If not, $\mathcal{P}_i$ broadcasts abort and aborts. (All parties abort if any party broadcasts abort.) Assuming that no party aborted, then $\mathcal{P}_i$ computes $b_{\mathcal{E}} := \sum_j b_j$, $\mathsf{sk}_i := \sum_j s_{j,i}$, and $\rho_i = \sum_j \rho_{j,i}$. $\mathcal{P}_i$ then computes the public commitments $\mathsf{com}_j := \sum_{k\in[n]} \mathsf{com}_{j,k}$.[3] The final public key is $\mathsf{pk}_{\mathcal{E}} := (a_{\mathcal{E}}, b_{\mathcal{E}}, \{\mathsf{com}_j\}_{j\in n})$ and secret key is then $\mathsf{sk}'_i = (\mathsf{sk}_i, \rho_i)$.

We remark that this protocol leads to the following equation for the distributed secret key: $\sum_{j\in\mathcal{U}} \lambda_j \mathsf{sk}_j = \mathsf{sk}$, for Lagrange coefficients λ_j.

Remark 1. We note that while the adversary can bias the distribution of the final key, the zero-knowledge proofs ensures that the noise values are properly bounded, and the honest parties ensures that the final key pair has appropriately entropy for a secure scheme.

3.4 Threshold Decryption for BGV

We now describe how threshold decryption is done. We begin by considering the semi-honest setting, and then define appropriate zero-knowledge proofs to defend against malicious behavior. For statistical security parameter sec and noise bound B_{Dec} described in Section 3.2, we have:

TDec: On input a set of users $\mathcal{U}$ of size t, decryption key share sk_i, and ciphertext $\mathsf{ctx} = (u, v)$, let λ_i be the Lagrange coefficient for party i with respect to $\mathcal{U}$. Sample noise $E_i \leftarrow D_{\mathsf{TDec}}$ such that $\|E_i\|_\infty \leq B_{\mathsf{TDec}} = 2^{\mathsf{sec}} B_{\mathsf{Dec}}/tp$ and output decryption share $\mathsf{ds}_i := \lambda_i \mathsf{sk}_i u + pE_i$.

Comb: On input a ciphertext $\mathsf{ctx} = (u, v)$ and partial decryption shares $\{\mathsf{ds}_j\}_{j\in\mathcal{U}}$, output plaintext message $\mathsf{ptx} := (v - \sum_{j\in\mathcal{U}} d_j) \bmod q \bmod p$.

[3] By the homomorphic properties of the underlying commitment scheme these are indeed commitments to sum of $s_{j,k}$ under the randomness of sum of $\rho_{j,k}$.

$\underline{\mathsf{DKGen}(a_{\mathcal{E}}, t, n)}$

$s_i, e_i \leftarrow \chi_{\mathsf{tern}}, \quad b_i := a_{\mathcal{E}} s_i + p e_i, \quad h_i := H(i, b_i)$

$$\xrightarrow{\quad h_i \quad}$$

$$\xleftarrow{\quad \{h_j\}_{j \neq i} \quad}$$

$\{s_{i,j}\}_{j\in[n]} \leftarrow \mathsf{Share}_{t,n}(s_i), \ \{e_{i,j}\}_{j\in[n]} \leftarrow \mathsf{Share}_{t,n}(e_i)$

$\forall j \in [n] : b_{i,j} := a_{\mathcal{E}} s_{i,j} + p e_{i,j}$

$\forall j \in [n] : \rho_{i,j} \leftarrow S_\rho, \ \mathsf{com}_{i,j} := \mathsf{Com}(s_{i,j}; \rho_{i,j})$

Compute NIZK π_i of shortness and linear relations

$$\xrightarrow{\quad b_i, \{b_{i,j}, \mathsf{com}_{i,j}\}_{j\in[n]}, \pi_i, \quad [s_{i,j}, e_{i,j}, \rho_{i,j}]_{j\in[n]} \quad}$$

$$\xleftarrow{\quad \{b_j, \{b_{j,k}, \mathsf{com}_{j,k}\}_{k\in[n]}, \pi_j, \quad [s_{j,i}, e_{j,i}, \rho_{j,i}]\}_{j\neq i} \quad}$$

if any $h_j \neq H(j, b_j)$: **abort** (j)

if any π_j **is invalid**: **abort** (j)

if any $0 = \mathsf{Open}(\mathsf{com}_{j,i}, s_{j,i}, \rho_{j,i})$: **abort** (j)

$b_{\mathcal{E}} := \sum_j b_j, \quad \mathsf{sk}_i := \sum_j s_{j,i}, \quad \rho_i = \sum_j \rho_{j,i}$

$\forall j \in [n] : \mathsf{com}_j := \sum_{k\in[n]} \mathsf{com}_{j,k}$

return $\mathsf{pk}_{\mathcal{E}} := (a_{\mathcal{E}}, b_{\mathcal{E}}, \{\mathsf{com}_j\}_{j\in[n]}), \quad \mathsf{sk}'_i := (\mathsf{sk}_i, \rho_i)$

Fig. 2. Actively secure key-generation protocol, from the point of view of $\mathcal{P}_i$. The elements in square brackets with subscript j are sent to $\mathcal{P}_j$ over a private channel.

Let $B_{\mathsf{Dec}} + tpB_{\mathsf{TDec}} < \lfloor q/2 \rfloor$. To show correctness, we note that

$$v - \sum_{j\in\mathcal{U}} d_j = v - \sum_{j\in\mathcal{U}} (\lambda_j \mathsf{sk}_j u + pE_j) = v - \mathsf{sk}\, u - p \sum_{j\in\mathcal{U}} E_j.$$

Then, from the definition of the encryption algorithm and Lemma 1 we know that $\|v - \mathsf{sk}u\|_\infty \leq B_{\mathsf{Dec}}$, and from the definition of the threshold decryption algorithm we have that $\|E_i\|_\infty \leq B_{\mathsf{TDec}}$. It follows that the total amount of noise is bounded by $\lfloor q/2 \rfloor$, and the scheme is perfectly correct.

The threshold decryption includes the following and is depicted in Figure 3:

1. TDec produces a proof $\pi_{\mathsf{ds},i}$ to show that partial decryption d_i is computed correctly (see Section 6). ds_i now also includes $\pi_{\mathsf{ds},i}$.
2. During Comb upon receiving ds_j, it first verifies each proof $\pi_{\mathsf{ds},j}$ and aborts with output j if any of them fails. Otherwise, it outputs ptx as above.

3.5 Proof of Security

We now prove that the protocol is secure against an active adversary $\mathcal{A}$ corrupting at most $t-1$ parties. We remark that we do not detail the running time of the adversary here and the rest of the paper, but it is straightforward to see that is tight with respect to the underlying assumptions.

$\underline{\mathsf{TDec}(\mathsf{ctx}^* = (u, v), \mathsf{sk}_i = (\mathsf{sk}_i, \rho_i), \mathcal{U})}$

if $\mathsf{ctx}^* = \perp$: **return** $\perp$

$m_i := \lambda_i \mathsf{sk}_i u, \quad E_i \leftarrow D_{\mathsf{TDec}}, \quad d_i := m_i + pE_i$

Compute NIZK $\pi_{\mathsf{ds},i}$ of boundedness and linear relations

return $\mathsf{ds}_i := (d_i, \pi_{\mathsf{ds},i})$

$\underline{\mathsf{Comb}(\mathsf{ctx}^*, \{\mathsf{ds}_j = (d_j, \pi_{\mathsf{ds},j})\}_{j \in \mathcal{U}})}$

if any $\pi_{\mathsf{ds},j}$ **is invalid**: **abort** (j)

$\mathsf{ptx} := v - \sum_{j \in \mathcal{U}} d_j \mod p$

return ptx

Fig. 3. Threshold decryption and share combination algorithms.

Theorem 1 (Indistinguishability). *Let the* $\mathsf{R\text{-}LWE}_{N,q,B_{\mathsf{tern}}}$ *and* $\mathsf{R\text{-}LWE}_{N,q,B_{\mathsf{TDec}}}$ *be* $\epsilon_{\mathsf{R\text{-}LWE}}$*- and* $\epsilon'_{\mathsf{R\text{-}LWE}}$*-hard. We model the hash function* H *as a random oracle and let* Q_H *be the number of queries made to* H*. Let the proof system* Π_x *for the relation* $\mathcal{R}_x$ *be* $(\epsilon_{\mathsf{ZK},x}, \epsilon_{\mathsf{extract},x})$*-secure and* Com *be* $\epsilon_{\mathsf{hiding}}$ *secure. Let* Q_{TDec} *be number of threshold decryption queries made. Finally, let the BGV scheme be perfectly correct and* ϵ_{BGV}*-IND-CPA secure with respect to* $(N, q, B_{\mathsf{KGen}}, B_{\mathsf{Dec}})$*. Then, for any adversary* $\mathcal{A}$ *corrupting up to* $t-1$ *parties:*

$$\epsilon_{\text{IND-CPA}} := \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{E}}(\mathcal{A}) \leq |\mathcal{H}| \cdot (\epsilon_{\mathsf{ZK,sk}} + |\mathcal{H}| \cdot \epsilon_{\mathsf{hiding}}) + |\mathcal{C}| \cdot \epsilon_{\mathsf{extract}} + \epsilon_{\mathsf{R\text{-}LWE}} + \frac{Q_H}{2^\ell} + Q_{\mathsf{TDec}} \cdot (|\mathcal{H}_{\mathcal{U}}| \cdot (\epsilon'_{\mathsf{R\text{-}LWE}} + \epsilon_{\mathsf{ZK,ds}})) + \epsilon_{\mathsf{BGV}}.$$

Proof. We prove security through a series of experiments:

Experiment G_0**.** The first experiment corresponds to the threshold ciphertext indistinguishability experiment shown in Figure 1.

Experiment G_1**.** We change how proofs during partial decryption are computed for honest parties $j \in \mathcal{H}_{\mathcal{U}}$. The proofs $\pi_{\mathsf{ds},j}$ are now computed by the simulator $\mathsf{Sim}_{\mathsf{ds}}$ of Π_{ds}. The rest of the experiment remains the same. G_1 is indistinguishable from G_0 by the zero-knowledge property of Π_{ds} and the distinguishing advantage of $\mathcal{A}$ per partial decryption query is $|\mathcal{H}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{ZK,ds}} \leq t \cdot \epsilon_{\mathsf{ZK,ds}}$ by a hybrid argument.

Experiment G_2**.** During DKGen, we use the knowledge extractor to obtain $\{s_{k,j}\}_{j \in [n]}$ for each corrupted party $\mathcal{P}_k$. (We abort the experiment if such extraction fails.) Then derive sk_k for $k \in \mathcal{C}$. The rest of the experiment remains the same. G_2 is then indistinguishable from G_1 by the knowledge extraction property of Π_{sk}. The distinguishing advantage of $\mathcal{A}$ is the cumulative bound on independent extraction failure properties, which is $|\mathcal{C}| \cdot \epsilon_{\mathsf{extract,sk}}$.

Experiment G_3**.** We replace how partial decryptions are computed for one of the honest parties. Let $m = \mathsf{ptx}$ be the message in a decryption query, and let (u, v) be the (honestly generated) encryption of m. Fix an index $i \in \mathcal{H}_{\mathcal{U}}$ and compute d_j for $j \neq i \in \mathcal{H}_{\mathcal{U}}$ as in the previous experiment. Then set the ith partial decryption share equal to $d'_i := v - m - \sum_{j \in \mathcal{U}, j \neq i} \lambda_j u\mathsf{sk}_j + pE'_i \mod q$. The rest of the experiment is the same.

In G_2 the partial decryption share d_i is computed as $d_i := \lambda_i u\mathsf{sk}_i + pE_i$, whereas here

$$\begin{aligned} d'_i &= v - m - \sum_{j \in \mathcal{U}, j \neq i} \lambda_j u\mathsf{sk}_j + pE'_i = v - m - (u\mathsf{sk} - \lambda_i u\mathsf{sk}_i) + pE'_i \\ &= v - u\mathsf{sk} - m + \lambda_i u\mathsf{sk}_i + pE'_i. \end{aligned}$$

The statistical distance between d_i and d'_i is thus the distance between pE_i and $v - u\mathsf{sk} - m + pE'_i$. First note that, E_i and E'_i are bounded uniforms from the same distribution, and $\|v - u\mathsf{sk}\|_\infty \leq B_{\mathsf{Dec}}$ and $\|m\|_\infty \leq p$. We also can rewrite $v - u\mathsf{sk} - m + pE'_i$ as $p((v - u\mathsf{sk} - m)/p + E'_i)$. The distinguishing probability between two distributions using a statistical argument is then how well the distribution of E_i and E'_i "smudges" the difference term $(v - u\mathsf{sk} - m)/p$:

$$\frac{\|(v - u\mathsf{sk} - m)/p\|_\infty}{\|E'_i\|_\infty} \leq \frac{(B_{\mathsf{Dec}} + p)/p}{(2^{\mathsf{sec}} B_{\mathsf{Dec}}/tp)} \approx t \cdot 2^{-\mathsf{sec}}.$$

G_2 and G_3 are thus statistically indistinguishable.

Experiment G_4**.** We replace the rest of the honest shares, i.e., all but the special party i. Instead of honestly computing d_j for $j \in \mathcal{H}_{\mathcal{U}}, j \neq i$, the simulator samples a uniform $d_j \in R_q$. In addition, d'_i is now replaced with as $d''_i := v - m - \sum_{j \in \mathcal{C}_{\mathcal{U}}} \lambda_j u\mathsf{sk}_j - \sum_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i} d_j \mod p + pE'_i \mod q$. The rest is the same as G_3. We need to show both d''_i and $\{d_j\}_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i}$ are indistinguishable from their counterparts in G_3. We first show the former assuming the latter. In G_3, d'_i is computed as $v - m - \sum_{j \in \mathcal{U}, j \neq i} \lambda_j u\mathsf{sk}_j + pE'_i$ whereas $d''_i = v - m - \sum_{j \in \mathcal{C}_{\mathcal{U}}} \lambda_j u\mathsf{sk}_j - \sum_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i} d_j \mod p + pE'_i \mod q$. If $\{d_j\}_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i}$ sampled uniformly is indistinguishable from $\{d_j\}_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i}$ computed as $d_j = \lambda_j u\mathsf{sk}_j + pE_j$, then the distributions of $\sum_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i} d_j \mod p$ are indistinguishable from $\sum_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i} \lambda_j u\mathsf{sk}_j \mod p$ since $d_j \mod p = \lambda_j u\mathsf{sk}_j + pE_j \mod p = \lambda_j u\mathsf{sk}_j \mod p$. Then if $\sum_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i} d_j \mod p$ and $\sum_{j \in \mathcal{H}_{\mathcal{U}}, j \neq i} u\mathsf{sk}_j$ are indistinguishable, so are d'_i and d''_i.

All it remains to show that $\lambda_j u\mathsf{sk}_j + pE_j$ is indistinguishable from uniform. Since λ_j are invertible in $\mathbb{Z}_q$, if we can argue $\lambda as + pe \approx as + \lambda^{-1}pe = as + p'e \approx as + e$ for uniform a and s, invertible p', and relatively short e then we can argue $\lambda as + pe$ is indistinguishable from uniform.

We first argue that $\mathcal{A}$ distinguishing $u\mathsf{sk}_j + \lambda_j^{-1} pE_j$ from uniform d_j can be used to break R-LWE: The R-LWE distinguisher interacts with $\mathcal{A}$ and uses its replies in G_4 to answer R-LWE challenges. The distinguisher obtains an R-LWE challenge (a_i, u_i), for each partial decryption query and sets partial ciphertext $u = a_i$ and partial decryption share $d_j = u_i$. During each query, if $\mathcal{A}$ behaves noticeably different than G_3, the R-LWE distinguisher replies that u_i is uniform.

Since λ_j is invertible and p is a prime, λ_j^{-1} exists and $\lambda^{-1}p$ is relatively prime with q. Hence, if u is an R-LWE instance, then there is no reason for the adversary to behave different than G_3 since $a_i\mathsf{sk}_j + e_i$ is indistinguishable from $a_i\mathsf{sk}_j + \lambda_j^{-1}pe_i$ for $\lambda_j^{-1}p$ relatively prime with q [BGV12]. If u_i is uniform, any significant advantage $\mathcal{A}$ has is R-LWE distinguisher advantage. Hence we conclude $u\mathsf{sk}_j + \lambda_j^{-1}pE_j$ is indistinguishable from uniform d_j.

If $u\mathsf{sk}_j+\lambda_j^{-1}pE_j$ is indistinguishable from uniform d_j then $\lambda_j(u\mathsf{sk}_j+\lambda_j^{-1}pE_j) = \lambda_j u\mathsf{sk}_j + pE_j$ is indistinguishable from $\lambda_j d_j$. Since d_j is uniform in R_q, so is $\lambda_j d_j$. Thus we conclude that $\lambda_j u\mathsf{sk}_j + pE_j$ is indistinguishable from uniform d_j by $\mathsf{R\text{-}LWE}_{N,q,B_{\mathsf{TDec}}}$ assumption.

Experiment G_5**.** Now that partial decryption does not rely on $\mathsf{sk}_{\mathcal{E}}$, we start to modify DKGen. Here we modify G_4 by simulating π_i sent during key generation for all honest parties. It follows immediately from the zero-knowledge property of the proof system that G_4 and G_5 are indistinguishable.

Experiment G_6**.** For all honest parties, we now replace the unopened commitments to $\{s_{i,j}\}$. After deriving $\boldsymbol{b}_{\mathcal{E}}$, each honest party commits to a random ring element instead. Since the commitments for the shares between honest parties are not opened, G_6 is indistinguishable from G_5 as long as the hiding property of the underlying commitment holds.

Experiment G_7**.** For i we now replace the public key share b_i. Sample a random b_i and t-out-of-n secret share it into $\{b_{i,j}\}_{j\in[n]}$. Then derive $s_{i,j}$,$e_{i.j}$ from $b_{i,j}$. The rest of the experiment remains the same.

We again show that $\mathcal{A}$ distinguishing between G_6 and G_7 can be used to break R-LWE assumption. We initiate the challenger for $\mathsf{R\text{-}LWE}_{N,q,B_{\mathsf{tern}}}$ with $a_{\mathcal{E}}$ being the uniform element. We then forward the answer of the challenger as b_i. If the $\mathcal{A}$ behaves significantly differently than G_5 we reply the challenger as the sample being uniform.

We have that $b_i = a_{\mathcal{E}}s_i + pe_i$ is an R-LWE sample, hence if the challenger returns an R-LWE sample, $\mathcal{A}$ has no reason to behave differently. If the challenger returns a uniform sample, however, any non-negligible advantage $\mathcal{A}$ has is non-negligible advantage in answering challenger's queries since any behavioral difference is forwarded directly. Since b_i is t-out-of-n secret shared, each $b_{i,j}$ is uniform in R_q. This is statistically indistinguishable from the real execution where s_i and e_i are t-out-of-n shared therefore $s_{i,j}$, $e_{i,j}$ and consequently $b_i = a_{\mathcal{E}}s_i + pe_i$ is uniform in R_q. G_6 is then indistinguishable from G_5 by R-LWE assumption and $\mathcal{A}$'s distinguishing advantage can be bounded by $\epsilon_{\mathsf{R\text{-}LWE}}$.

Experiment G_8**.** We now derive b_i a posteriori. Before step 1 of DKGen, $\mathcal{B}$ receives a BGV key pair $(a_{\mathcal{E}}, b_{\mathcal{E}})$. In step 1 samples a random $h_i \leftarrow \{0,1\}^{\ell}$. After receiving h_k for $k \in \mathcal{C}$ from $\mathcal{A}$ finds b_k such that $h_k = H(k, b_k)$. Then derive $b_i := b_{\mathcal{E}} - \sum_{j\neq i\in[n]} b_j$ and program H such that $H(i, b_i) = h_{b_i}$. If programming fails, abort. The rest of the experiment is the same as before.

The distribution of h_i is statistically indistinguishable from the output of H by the random oracle model. By the R-LWE assumption, $b_{\mathcal{E}}$ is indistinguishable

from uniform. Then, G_6 and G_7 are indistinguishable as long as the programming does not fail. The advantage of $\mathcal{A}$ is the probability of programming failing, which by a standard argument can be bounded by $\frac{Q_{\mathsf{DKGen}}}{2^\ell}$.

We now show that if $\mathcal{A}$ wins G_8, then $\mathcal{A}$ can be used to break the IND-CPA security of BGV scheme. $\mathcal{B}$ interacts with the challenger to IND-CPA security of BGV and obtains $a_{\mathcal{E}}, b_{\mathcal{E}}$ which then are used during DKGen. Whenever $\mathcal{A}$ sends the challenge plaintexts ptx_0 and ptx_1 $\mathcal{B}$ forwards them to the BGV challenger and obtains the ciphertext ctx^*. $\mathcal{B}$ sends ctx^* along with its simulated proof to $\mathcal{A}$. When $\mathcal{A}$ replies with 0 or 1, $\mathcal{B}$ forwards the response to BGV challenger.

The public key $\mathcal{A}$ generates as part of DKGen contains the same $(a_{\mathcal{E}}, b_{\mathcal{E}})$ as $\mathcal{B}$ received from the BGV oracle as a challenge. Hence ctx^* received by $\mathcal{B}$ and consequently $\mathcal{A}$ is a BGV ciphertext encrypted under $a_{\mathcal{E}}, b_{\mathcal{E}}$ which indistinguishable from ctx derived as part of the protocol execution. Then, a correct answer given by $\mathcal{A}$ is also a correct answer for $\mathcal{B}$ for the encryption oracle.

This concludes the proof. □

4 Passively Secure t-out-of-n Threshold Signatures

We give a passively secure version of our t-out-of-n threshold signature protocol $\mathcal{TS}$. For brevity, we omit the exact bounds and parameters as this section serves mainly as a warm-up. Let $\mathcal{E} = (\mathsf{DKGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{TDec}, \mathsf{Comb})$ be a threshold (linearly) homomorphic encryption scheme.

4.1 Dilithium without Aborts

We use a ring version of Dilithium without rejection sampling as the underlying signature scheme, extending [ASY22]. The underlying signature scheme is also similar to Raccoon [dPEK+23] (which was developed in concurrent work). We define the challenge set $\mathcal{C}_\nu = \{c \in R_q : \|c\|_\infty = 1, \|c\|_1 = \nu\} \subset R_q$ to be the set of polynomials with coefficients in $\{-1, 0, 1\}$ and exactly ν non-zero coefficients. We also let $\bar{\mathcal{C}}_\nu = \{c - c' : c, c' \in \mathcal{C}_\nu, c \neq c'\}$.

KGen_L Samples a uniform $a \in R_q$, then set $\boldsymbol{a} := \begin{bmatrix} a & 1 \end{bmatrix}$. Samples bounded uniform short secret key s_1, s_2 with $\|s_1\|_\infty = \|s_2\|_\infty \leq \eta$ then set $\mathbf{s} := \begin{bmatrix} s_1 & s_2 \end{bmatrix}$. Finally, computes $y := \langle \boldsymbol{a}, \mathbf{s} \rangle$ and outputs $\mathsf{sk} = \mathbf{s}$ and $\mathsf{pk} = (\boldsymbol{a}, y)$.

Sign_L Takes as input $(\mathsf{sk}, \mathsf{pk}, \mu)$, samples $r_1, r_2 \leftarrow D_\sigma$ from a Gaussian distribution with standard deviation σ, and set $\boldsymbol{r} := \begin{bmatrix} r_1 & r_2 \end{bmatrix}$. Computes $w := \langle \boldsymbol{a}, \boldsymbol{r} \rangle$, derive challenge $c := H(w, \mathsf{pk}, \mu) \in \mathcal{C}_\nu$, and outputs signature $(c, \boldsymbol{z} := c\mathbf{s} + \boldsymbol{r})$.

Vrfy_L Takes as input $(\mathsf{pk}, (\boldsymbol{z}, c), \mu)$ and checks that $\|\boldsymbol{z}\|_2 \leq B$ and $c = H_0(\langle \boldsymbol{a}, \boldsymbol{z} \rangle - cy, \mathsf{pk}, \mu)$ and then outputs 1, otherwise outputs 0.

Agrawal et al. [ASY22, Section 4] prove the correctness and security of this scheme:

Lemma 2. *Let $\gamma = \lambda + (N \log_2 q)/(\log_2(2\eta + 1))$, and furthermore set $\beta = 2B + 2 \cdot \eta \cdot \nu \cdot \sqrt{\gamma}$ and $\sigma \geq \nu \cdot \eta \cdot \sqrt{\gamma \cdot Q}$, where Q is the maximum number*

$\underline{\mathsf{KGen}_{\mathcal{TS}}(\mathsf{crs} = (\boldsymbol{a}_{\mathcal{TS}}, a_{\mathcal{E}}), t, n)}$

$\underline{\mathsf{DKGen}(\mathsf{crs} = a_{\mathcal{E}}, t, n)}$

$\longrightarrow$

$\longleftarrow$

return $\mathsf{pk}_{\mathcal{E}}, \mathsf{sk}_i$

$s_{i,1}, s_{i,2} \leftarrow D, \quad \mathbf{s}_i := \begin{bmatrix} s_{i,1} & s_{i,2} \end{bmatrix}$

$y_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}_i \rangle, \quad \mathsf{ctx}_{\mathbf{s},i} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i)$

$\xrightarrow{y_i, \mathsf{ctx}_{\mathbf{s},i}}$

$\xleftarrow{\{(y_j, \mathsf{ctx}_{\mathbf{s},j})\}_{j \neq i}}$

$y := \sum_{j \in [n]} y_j, \quad \mathsf{ctx}_{\mathbf{s}} := \sum_{j \in [n]} \mathsf{ctx}_{\mathbf{s},j}$

return $\mathsf{pk}_{\mathcal{TS}} := (\boldsymbol{a}_{\mathcal{TS}}, y), \quad \mathsf{sk}_i, \quad \mathsf{aux}_{\mathcal{TS}} := (\mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}})$

Fig. 4. Passively secure key-generation protocol for signer $\mathcal{S}_i$.

of signing queries an adversary can make, and let H be modeled as a random oracle. If the $\mathsf{R\text{-}SIS}_{N,q,\beta}$ *problem is hard, then the signature scheme above is* $\mathsf{uf\text{-}cma}$*-secure.*

4.2 Threshold Key Generation and Signing Protocols

We describe the underlying protocols of $\mathcal{TS}$ from the viewpoint of a single signer $\mathcal{S}_i$ with $i \in [n]$ (for DKGen) and $i \in \mathcal{U} \subset [n]$, $|\mathcal{U}| = t$ (for Sign). We once again assume access to a crs consisting of a uniform ring element $a_{\mathcal{TS}} \in R_q$. The key generation protocol is depicted in Figure 4.

The key generation $\mathsf{KGen}_{\mathcal{TS}}$ goes as follows:

1. The parties begin by invoking the passively secure distributed key generation protocol DKGen of the underlying threshold homomorphic encryption scheme with inputs t,n for all circuits consisting of one multiplication with an element from $\mathcal{C}_\nu$ and t additions and all circuits consisting of n additions. $\mathcal{S}_i$ learns the public encryption key $\mathsf{pk}_{\mathcal{E}}$ and its decryption key share sk_i.
2. The parties are given $\boldsymbol{a}_{\mathcal{TS}} = \begin{bmatrix} a_{\mathcal{TS}} & 1 \end{bmatrix}$ as the common reference string. Then $\mathcal{S}_i$ samples short $s_{i,1}, s_{i,2}$ and sets $\mathbf{s}_i = \begin{bmatrix} s_{i,1} & s_{i,2} \end{bmatrix}$ and $y_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}_i \rangle$. It computes the ciphertext $\mathsf{ctx}_{\mathbf{s},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i)$ and broadcasts it with y_i.
3. The parties compute $\mathsf{ctx} := \sum \mathsf{ctx}_{\mathbf{s},j} = \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s})$ and $y := \sum y_j = \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s} \rangle$, where $\mathbf{s} = \sum \mathbf{s}_j$, and define the public verification key to be $\mathsf{pk}_{\mathcal{TS}} := (\boldsymbol{a}_{\mathcal{TS}}, y)$. The secret key of $\mathcal{S}_i$ consists of its decryption key share sk_i, and the auxiliary information $\mathsf{aux} := (\mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}})$. The signing share $\mathbf{s}_i$ is deleted.

The signing protocol depicted in Figure 5 $\mathsf{Sign}_{\mathcal{TS}}$ goes as follows:

$\underline{\mathsf{Sign}_{\mathcal{TS}}(\mathsf{sk}_i, \mathsf{aux}_{\mathcal{TS}}, \mathcal{U}, \mu)}$

$r_{i,1}\ r_{i,2} \leftarrow D_r, \quad \boldsymbol{r}_i := \left[r_{i,1}\ r_{i,2}\right]$

$w_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{r}_i \rangle, \quad \mathsf{ctx}_{\boldsymbol{r},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_i)$

$\xrightarrow{\quad w_i, \mathsf{ctx}_{\boldsymbol{r},i} \quad}$

$\xleftarrow{\quad \{(w_j, \mathsf{ctx}_{\boldsymbol{r},j})\}_{j \in \mathcal{U} \setminus \{i\}} \quad}$

$w := \sum_{j \in \mathcal{U}} w_j, \quad c := H(w, \mathsf{pk}_{\mathcal{TS}}, \mu)$

$\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j \in \mathcal{U}} \mathsf{ctx}_{\boldsymbol{r},j}$

$\mathsf{ds}_i := \mathsf{TDec}(\mathsf{ctx}_{\boldsymbol{z}}, \mathsf{sk}_i, \mathcal{U})$

$\xrightarrow{\quad \mathsf{ds}_i \quad}$

$\xleftarrow{\quad \{\mathsf{ds}_j\}_{j \in \mathcal{U} \setminus \{i\}} \quad}$

$\boldsymbol{z} := \mathsf{Comb}(\mathsf{ctx}_{\boldsymbol{z}}, \{\mathsf{ds}_j\}_{j \in \mathcal{U}})$

return $\sigma := (c, \boldsymbol{z})$

Fig. 5. Passively secure t-out-of-n threshold signing protocol for signer $\mathcal{S}_i$.

1. To sign a message μ, party $\mathcal{S}_i$ first samples short ring elements $r_{i,1}$,$r_{i,2}$ and defines vector $\boldsymbol{r}_i := \left[r_{i,1}\ r_{i,2}\right]$. Signer $\mathcal{S}_i$ then computes $w_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{r}_i \rangle$ and generates the ciphertext $\mathsf{ctx}_{\boldsymbol{r},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_i)$ and broadcasts that ciphertext and w_i to the other signers in $\mathcal{U}$.
2. The signers compute $w := \sum_{j \in \mathcal{U}} w_j$ and $c := H(w, \mathsf{pk}_{\mathcal{TS}}, \mu) \in \mathcal{C}_\nu$, followed by the ciphertext $\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j \in \mathcal{U}} \mathsf{ctx}_{\boldsymbol{r},j}$. Party $\mathcal{S}_i$ then computes a decryption share $\mathsf{ds}_i := \mathsf{TDec}(\mathsf{sk}_i, \mathsf{ctx}_{\boldsymbol{z}}, \mathcal{U})$ and sends it to the other signers. The signers decrypt $\mathsf{ctx}_{\boldsymbol{z}}$ to obtain $\boldsymbol{z}$, and output the signature $(c, \boldsymbol{z})$.

$\mathsf{Vrfy}_{\mathcal{TS}}$: A signature $(c, \boldsymbol{z})$ on a message μ is valid with respect to the public key $\mathsf{pk}_{\mathcal{TS}} = (\boldsymbol{a}_{\mathcal{TS}}, y)$ if (1) $\boldsymbol{z}$ is short and (2) $H(\langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z} \rangle - cy, \mathsf{pk}_{\mathcal{TS}}, \mu) = c$.

For a signature $(c, \boldsymbol{z})$ output by the signing protocol on a message μ and a set of users $|\mathcal{U}| \geq t$, we have $\boldsymbol{z} = c \cdot \mathbf{s} + \sum_{j \in \mathcal{U}} \boldsymbol{r}_j$ by the linearity of the encryption scheme (assuming parameters are set so that decryption errors never occurs). Since $c \in \mathcal{C}_\nu$ and $\mathbf{s}, \{\boldsymbol{r}_j\}$ are short, then $\boldsymbol{z}$ is short as well. Moreover, we have

$$\langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z} \rangle - cy = c\langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s} \rangle + \langle \boldsymbol{a}_{\mathcal{TS}}, \sum_{j \in \mathcal{U}} \boldsymbol{r}_j \rangle - c\langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s} \rangle \ = \ w;$$

thus, $H(\langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z} \rangle - cy, \mathsf{pk}_{\mathcal{TS}}, \mu) = H(w, \mathsf{pk}_{\mathcal{TS}}, \mu) = c$ and verification succeeds.

4.3 Proof of Security

Theorem 2 (Informal). *The threshold signature scheme $\mathcal{TS}$ is threshold existentially unforgeable under chosen message attacks (*$\mathsf{ts\text{-}uf\text{-}cma}$*) in the random oracle model (ROM) if the underlying signature scheme is existentially unforgeable under chosen message attacks in the ROM, the threshold homomorphic encryption scheme $\mathcal{E}$ is indistinguishable, and the LWE assumption is secure.*

We prove the security of the scheme via a sequence of hybrid experiments. Starting from the threshold unforgeability experiment, we gradually define a simulator $\mathcal{B}$ interacting with the passive adversary $\mathcal{A}$. $\mathcal{B}$ has access to a challenger $\mathcal{D}$ to the unforgeability of Dilithium scheme without aborts as described in Section 4.1 and can query the challenger for a signature $\sigma = (c, \boldsymbol{z})$ on input of a message μ or the public key $\mathsf{pk}_{\mathcal{TS}} = (\boldsymbol{a}_{\mathcal{TS}}, y)$. We show that if $\mathcal{A}$ can forge a signature, it can be used by $\mathcal{B}$ as an answer to $\mathcal{D}$.

$\boldsymbol{G}_0$. The first experiment corresponds to a passive adversary $\mathcal{A}$ corrupting parties in $\mathcal{C} \subset [n]$ such that $|\mathcal{C}| < t$ and attacking the threshold signature scheme $\mathcal{TS}$ in the real world. Consider the view of $\mathcal{A}$ during this experiment. During key generation, $\mathcal{A}$'s view is generated as follows:

1. DKGen is run with t, n, d, and $\mathcal{A}$ is given the collective view $\mathsf{view}_{\mathcal{C}}$ of the parties in $\mathcal{C}$ that, in particular, includes a public key $\mathsf{pk}_{\mathcal{E}}$ and key shares $\{\mathsf{sk}_j\}_{j\in\mathcal{C}}$. This process defines secret key shares $\{\mathsf{sk}_j\}_{j\in\mathcal{H}}$ (not given to $\mathcal{A}$).
2. For $j \in [n]$, sample $s_{j,1}, s_{j,2} \leftarrow D$ and compute $\mathbf{s}_j := [s_{j,1}\ s_{j_2}]$, $y_j := \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}_j \rangle$, and $\mathsf{ctx}_{\mathbf{s},j} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_j)$; give $\{\mathbf{s}_j\}_{j\in\mathcal{C}}$, $\{(y_j, \mathsf{ctx}_{\mathbf{s},j})\}_{j\in\mathcal{H}}$ to $\mathcal{A}$. Set $y := \sum_{j\in[n]} y_i$ and $\mathsf{ctx}_{\mathbf{s}} := \sum_{j\in[n]} \mathsf{ctx}_{\mathbf{s},j}$.

$\mathcal{A}$ repeatedly invokes the signing protocol (possibly concurrently) by specifying a message μ and a set of parties $\mathcal{U}$. Whenever $\mathcal{A}$ queries the random oracle on $(w, \mathsf{pk}_{\mathcal{TS}}, \mu)$ then $\mathcal{B}$ forwards the query to $\mathcal{D}$, records the response in a table $\mathcal{HT}$, and forwards the response to $\mathcal{A}$. Letting $\mathcal{H}_{\mathcal{U}} = \mathcal{U} \cap \mathcal{H}$ and $\mathcal{C}_{\mathcal{U}} = \mathcal{U} \cap \mathcal{C}$, the view of $\mathcal{A}$ is generated as follows:

1. For $j \in \mathcal{U}$, sample $r_{j,1}, r_{j_2} \leftarrow D_r$ and compute $\boldsymbol{r}_j := [r_{j,1}\ r_{j,2}]$, $w_j := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{r}_j \rangle$ and $\mathsf{ctx}_{\boldsymbol{r},j} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_j)$. $\mathcal{A}$ gets $\{\boldsymbol{r}_j\}_{j\in\mathcal{C}_{\mathcal{U}}}$ and $\{(w_j, \mathsf{ctx}_{\boldsymbol{r},j})\}_{j\in\mathcal{U}}$. Set $w := \sum_{j\in\mathcal{U}} w_j$, $c := H(w, \mathsf{pk}_{\mathcal{TS}}, \mu)$, and compute $\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j\in\mathcal{U}} \mathsf{ctx}_{\boldsymbol{r},j}$.
2. For $j \in \mathcal{U}$, set $\mathsf{ds}_j := \mathsf{TDec}(\mathsf{sk}_j, \mathsf{ctx}_{\boldsymbol{z}}, \mathcal{U})$ and give $\{\mathsf{ds}_j\}_{j\in\mathcal{U}}$ to $\mathcal{A}$.

At the end of the experiment, $\mathcal{A}$ outputs a message/signature pair $(\mu^*, \sigma^* := (c^*, \boldsymbol{z}^*))$. If μ^* was never previously queried and the signature is valid with respect to y, then $\mathcal{A}$ succeeds. We have that

$$\Pr[\mathbf{G}_0] = \mathsf{Adv}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{TS}}(\mathcal{A}).$$

$\boldsymbol{G}_1$*:* This experiment is identical to $\mathbf{G}_1$ except how the ciphertexts are computed. Note that the decryption procedure is independent of the encrypted randomness $\boldsymbol{r}_i$. During key generation, $\mathcal{B}$ computes $\mathsf{ctx}_{\mathbf{s},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, 0)$ for $i \in \mathcal{H}$, and in step 1 of $\mathsf{Sign}_{\mathcal{TS}}$ $\mathcal{B}$ computes $\mathsf{ctx}_{\boldsymbol{r},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, 0)$ for $i \in \mathcal{H}_{\mathcal{U}}$. The rest of the execution is the same as $\mathbf{G}_1$.

If $\mathcal{A}$ can distinguish between $\mathbf{G}_0$ and $\mathbf{G}_1$, then it is possible to use $\mathcal{A}$ to break indistinguishability of $\mathcal{E}$. When interacting with the challenger to IND-CPA of $\mathcal{E}$, $\mathcal{B}$ submits $(0, \mathbf{s}, i)$ for $i \in \mathcal{H}$ during key generation and $(0, \boldsymbol{r}_i)$ for $i \in \mathcal{H}_{\mathcal{U}}$ during signing to the challenger to obtain $\mathsf{ctx}_{\mathbf{s},i}$ and $\mathsf{ctx}_{\boldsymbol{r},i}$ respectively. If $\mathcal{A}$ behaves noticeably different in any of these stages, $\mathcal{B}$ forwards 0 to the challenger, indicating the plaintext was 0.

If the IND-CPA challenger has originally given ciphertexts corresponding to $\mathbf{s}_i$ and $\boldsymbol{r}_i$ to $\mathcal{B}$, then $\mathbf{G}_1$ is exactly the same as $\mathbf{G}_0$, therefore there is no reason for $\mathcal{A}$ to have noticeable behavior difference. If the challenger has encrypted 0 at one stage however, $\mathcal{B}$ can use $\mathcal{A}$ to have noticeable advantage in IND-CPA game of $\mathcal{E}$. Note that this is possible since the views given to $\mathcal{A}$ are generated by $\mathcal{B}$ therefore $\mathcal{B}$ knows plaintext corresponding to each partial decryption query. Thus we conclude that $\mathbf{G}_0$ and $\mathbf{G}_1$ are indistinguishable as long as IND-CPA security of $\mathcal{E}$ holds:

$$|\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_0]| \leq \epsilon_{\text{IND-CPA}}.$$

$\boldsymbol{G}_2$*:* In this experiment, $\mathcal{B}$ removes the dependence on the signing key $\mathbf{s}$.

$\mathcal{B}$ first changes the key generation step for one honest party. At the start of key generation, $\mathcal{B}$ receives the public key from $\mathcal{D}$ where $\mathcal{D}$ is initialized with the parameters for the combined signature (we detail these in the active version of the proof). Let i be some index in $\mathcal{H}$ and $\mathcal{H}_{\mathcal{U}}$. During step 2, everything is computed in the same was as in $\mathbf{G}_2$ except for y_i where $s_{i,1}$ and $s_{i,2}$ are never sampled and $y_i := y - \sum_{j \in [n], j \neq i} y_j$.

Whenever μ is to be signed, $\mathcal{B}$ queries $\mathcal{D}$ to obtain a signature $(c, \boldsymbol{z})$. During the first step of signing for $j \in \mathcal{C}_{\mathcal{U}}$, sample $r_{j,1}, r_{j,2} \leftarrow D_r$ and set $\boldsymbol{r}_j := [r_{j,1}\ r_{j,2}]$ and $w_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{r}_j \rangle$ as before. Letting i denote some index in $\mathcal{H}_{\mathcal{U}}$, sample $w_j \leftarrow R_q$ uniformly for $j \in \mathcal{H}_{\mathcal{U}} \setminus \{i\}$, and set $w_i := w - \sum_{j \in \mathcal{U} \setminus \{i\}} w_j$, where w is computed from $(c, \boldsymbol{z})$ as $w := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z} \rangle - cy$. $\mathcal{B}$ encrypts the received $\boldsymbol{z}$ to obtain $\mathsf{ctx}_{\boldsymbol{z}} = \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{z})$. The rest of the experiment is as before.

$\mathcal{D}$ is initialized with the parameters for the combined signature. Hence, $y = \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}' \rangle$ for an unknown $\mathbf{s}'$ with $\|\mathbf{s}'\| \leq tB$. Consequently $y_i := y - \sum_{j \in [n] \setminus \{i\}} y_j = \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}'_i \rangle$ by the linearity of the operations for an unknown $\mathbf{s}'_i$ with $\|\mathbf{s}'_i\| \leq B$, which is the same distribution for the honest $\mathbf{s}_i$.

w is obtained from $\mathcal{D}$, which is $ar'_1 + r'_2$ for unknown r'_1, r'_2 and hence an R-LWE sample from combined parameters. Since $\{w_j\}_{j \in \mathcal{C}_{\mathcal{U}}}$ are computed according to the protocol, then $\{w_j\}_{j \in \mathcal{H}_{\mathcal{U}}}$ is computationally indistinguishable from uniform by the same R-LWE assumption. If $\mathcal{A}$ can distinguish between $\mathbf{G}_2$ and $\mathbf{G}_3$ then since $a_{\mathcal{TS}}, y, y_i$ are distributed in the same way as in $\mathbf{G}_2$, $\mathcal{A}$ can also distinguish $\{w_i\}_{i \in \mathcal{H}_{\mathcal{U}}}$ in games $\mathbf{G}_1$ and $\mathbf{G}_2$.

If $\mathcal{A}$ can distinguish w_i then it can be used to solve R-LWE. After $\mathcal{B}$ initializes the challenger for R-LWE for parameters of combined $\boldsymbol{r}$, $\mathcal{B}$ computes w_j for $j \neq i \in \mathcal{U}$ according to the protocol and derives $w_i := w - \sum_{j \in \mathcal{U} \setminus \{i\}}$ and sends $\{w_j\}_{j \in \mathcal{U}}$ to $\mathcal{A}$. If $\mathcal{A}$ acts with noticeable difference, $\mathcal{B}$ answer the challenger that $\boldsymbol{u}$ was uniform.

If the challenger returned an R-LWE sample for u, the derived w_i will have the same distribution as an honestly computed w_i in $\mathbf{G}_2$ therefore there is no reason for $\mathcal{A}$ to have a noticeable behavior difference. If the challenger sent a uniform u however, w_i should be computationally indistinguishable from uniform by R-LWE assumption hence if $\mathcal{A}$ can act with noticeable difference then R-LWE assumption should not hold and $\mathcal{B}$ can answer challenger's query with noticeable advantage using $\mathcal{A}$. Thus we conclude that $\mathbf{G}_1$ and $\mathbf{G}_2$ are indistinguishable as long as R-LWE holds:

$$|\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1]| \leq \epsilon_{\mathsf{R\text{-}LWE}} + \frac{Q_H}{2^\ell}.$$

We remark that the combined signature in $\mathbf{G}_2$ now has the same distribution for $(\boldsymbol{a}_{\mathcal{TS}}, y, c, \boldsymbol{z})$ as the underlying signature for combined parameters, since $(\boldsymbol{a}_{\mathcal{TS}}, y, c, \boldsymbol{z})$ are received from $\mathcal{D}$. If $\mathcal{A}$ is able to produce a forgery (μ^*, σ^*) now, this forgery is also against the underlying signature scheme, and hence, the adversary can be used to break the uf-cma security. Whenever $\mathcal{A}$ submits a forgery (μ^*, σ^*), $\mathcal{B}$ can submit the said forgery to $\mathcal{D}$, which would have noticeable probability of winning as long as $\mathcal{A}$ has noticeable advantage. Thus we have:

$$\Pr[\mathbf{G}_2] = \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{A}).$$

This concludes the proof. □

Since we remove the reliance on aborts, each signature may leak information about the secret key as discussed by Lyubashevsky [Lyu12]. Exact parameters rely on either limiting the number of signatures issued or on noise drowning. Parameters for active security are analyzed in detail in Section 7.

The actively secure version of our protocol is more involved since we assume a rushing and active adversary, and hence, we need parties to commit to specific values, provide zero-knowledge proofs, and conduct consistency checks to ensure the privacy and correctness of the protocol.

5 Actively Secure t-out-of-n Threshold Signatures

We now describe our main contribution: an actively secure threshold signature scheme, constructed by bootstrapping the passively secure protocol described in Section 4. The key generation and signing protocols are depicted in Figure 6 and Figure 7, respectively. We extend the previous section by giving concrete bounds and dimensions for the protocol, discussing the communication efficiency in each round, and giving a detailed security proof.

We start by modifying the underlying signature protocol. Instead of using w directly as part of the oracle input for challenge derivation, we use a commitment

com to w instead. This is the same approach taken by Damgård et al. [DOTT21] on Dilithium-G [DKL+18] and does not have any important security implications on the signatures as long as the underlying commitment scheme is secure.

5.1 Key Generation and Signing Protocols

We retain our notation and viewpoint from the passive protocol and introduce homomorphic commitments and non-interactive zero-knowledge proofs. Note that we change the signatures so that the challenge is computed as the hash of the sum of commitments to the values w_j instead of the values themselves, and openings are published afterward as a part of the signature.

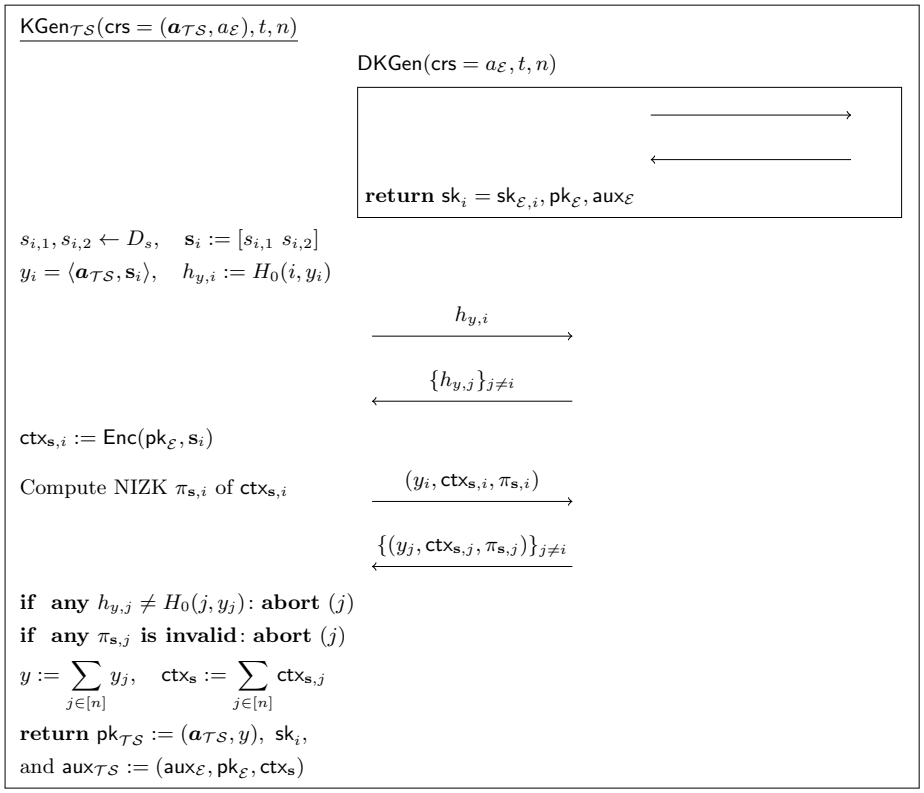

Fig. 6. Actively secure key generation protocol for signer $\mathcal{S}_i$.

$\mathsf{KGen}_{\mathcal{TS}}$ works as follows:

1. $\mathcal{S}_i$ starts by invoking the distributed key generation DKGen of the underlying encryption scheme $\mathcal{E}$ with inputs t and n as in the passive case and obtains the public encryption key $\mathsf{pk}_{\mathcal{E}}$, its threshold decryption key share sk_i and any auxiliary information $\mathsf{aux}_{\mathcal{E}}$ associated with $\mathcal{E}$.

2. $\mathcal{S}_i$ then samples short signing key $s_{i,1}, s_{i,2}$ sets $\mathbf{s}_i := [s_{i,1}\ s_{i,2}]$, $y_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}_i \rangle$, hash $h_{y,i} := H_0(i, y_i)$ as a commitment to y_i, and broadcasts $h_{y,i}$. Upon receiving $h_{y,j}$ for all $j \neq i$, $\mathcal{S}_i$ encrypts $\mathbf{s}_i$ as $\mathsf{ctx}_{\mathbf{s},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i)$ and computes a NIZK proof $\pi_{\mathbf{s},i}$ to prove that the correct $\mathbf{s}_i$ was used for y_i and $\mathsf{ctx}_{\mathbf{s},i}$ (see Section 6), then broadcasts $(y_i, \mathsf{ctx}_{\mathbf{s},i}, \pi_{\mathbf{s},i})$.
3. Finally, upon receiving $\mathsf{ctx}_{\mathbf{s},j}$, $\pi_{\mathbf{s},j}$ and y_j from each $j \neq i$, $\mathcal{S}_i$ verifies that $h_{y,j} = H_0(j, y_j)$ and that $\pi_{\mathbf{s},j}$ is valid with respect to $\mathsf{ctx}_{\mathbf{s},j}$ and y_j, and aborts with output j if any of them fails. If all checks succeed, it defines the public key $\mathsf{pk}_{\mathcal{TS}} = (\boldsymbol{a}_{\mathcal{TS}}, y)$, secret key sk_i, and auxiliary information $\mathsf{aux}_{\mathcal{TS}} = (\mathsf{aux}_{\mathcal{E}}, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s}})$ where $y := \sum y_j = \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s} \rangle$, $\mathsf{ctx}_{\mathbf{s}} := \sum \mathsf{ctx}_{\mathbf{s},j}$.

$\mathsf{Sign}_{\mathcal{TS}}$ works as follows:

1. Let $\mathcal{S}_i$ be one out of t signers in the set $\mathcal{U}$. Upon receiving the message μ to be signed, $\mathcal{S}_i$ samples per signatures randomness $r_{i,1}, r_{i,2} \leftarrow D_\sigma$ and commitment randomness $\rho_i \leftarrow \chi$, derives per message commitment key $\mathsf{ck} = H_1(\mathsf{pk}_{\mathcal{TS}}, \mu)$, set $\boldsymbol{r}_i := [r_{i,1}\ r_{i,2}]$, and computes $w_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{r}_i \rangle$ and commitment $\mathsf{com}_i := \mathsf{Com}_{\mathsf{ck}}(w_i, \rho_i)$. $\mathcal{S}_i$ then encrypts the randomness $\boldsymbol{r}_i$ with the encryption key $\mathsf{pk}_{\mathcal{E}}$ as $\mathsf{ctx}_{\boldsymbol{r},i} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_i)$, and computes a NIZK proof $\pi_{\boldsymbol{r},i}$ to prove that $\boldsymbol{r}_i$ was used for both com_i and $\mathsf{ctx}_{\boldsymbol{r},i}$ (see Section 6). $\mathcal{S}_i$ then sends $\mathsf{ctx}_{\boldsymbol{r},i}$, com_i and $\pi_{\boldsymbol{r},i}$ to all $j \in \mathcal{U} \setminus \{i\}$.
2. Upon receiving $\mathsf{ctx}_{\boldsymbol{r},j}$, com_j and $\pi_{\boldsymbol{r},j}$ for each $j \in \mathcal{U} \setminus \{i\}$, $\mathcal{S}_i$ aborts with output j if $\pi_{\boldsymbol{r},j}$ does not verify with respect to $\mathsf{ctx}_{\boldsymbol{r},j}$ and com_j. Otherwise it computes $\mathsf{com} := \sum \mathsf{com}_j$ for all $j \in \mathcal{U}$ and derives the challenge $c := H_2(\mathsf{com}, \mathsf{pk}_{\mathcal{TS}}, \mu)$. It then computes the encryption of the signature as $\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j \in \mathcal{U}} \mathsf{ctx}_{\boldsymbol{r},j}$ (that is, computing Eval on the ciphertexts where F is the function taking an element from $\mathcal{C}_\nu$, multiplying it with $\mathsf{ctx}_{\mathbf{s}}$, and adding t ciphertexts $\mathsf{ctx}_{\boldsymbol{r},j}$ to the result) and decrypts its share as $\mathsf{ds}_i := \mathsf{TDec}(\mathsf{ctx}_{\boldsymbol{z}}, \mathsf{sk}_i, \mathcal{U})$ and sends the partial decryption ds_i along with opening w_i and commitment randomness ρ_i to the signers in $\mathcal{U}$.
3. Upon receiving ds_j, w_j, and ρ_j for all $j \in \mathcal{U} \setminus \{i\}$, $\mathcal{S}_i$ aborts with output j if $\mathsf{Open}(\mathsf{com}_j, w_j, \rho_j) = 0$ for any j. Then tries to combine the decryptions as $\boldsymbol{z} := \mathsf{Comb}(\mathsf{ctx}_{\boldsymbol{z}}, \{\mathsf{ds}_j\}_{j \in \mathcal{U}})$ and aborts with output j if $\boldsymbol{z} = \perp$ and Comb aborts with output j. $\mathcal{S}_i$ finally outputs the signature $\sigma := (c, \boldsymbol{z}, \rho)$ where $\rho := \sum \rho_j$ for all $j \in \mathcal{U}$.

$\mathsf{Vrfy}_{\mathcal{TS}}$: Upon receiving $\sigma := (c, \boldsymbol{z}, \rho)$ and μ, checks that $\|\boldsymbol{z}\| \leq B_z$ and $\|\rho\| \leq B_\rho$, computes $w^* := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z} \rangle - cy$, derives $c^* := H_2(\mathsf{Com}(w^*; \rho), \mathsf{pk}_{\mathcal{TS}}, \mu)$, then finally outputs 1 if and only if checks hold and $c = c^*$, and 0 otherwise.

5.2 Correctness, Bounds, and Sizes

We proved the correctness of the passively secure signature scheme in Section 4.2, and as the commitment scheme and the zero-knowledge schemes are complete, then it follows that the actively secure signature scheme is correct. Furthermore, the bounds in the protocol depend on the distributions we sample from. If we sum t samples from a uniform distribution over the values $[-B, B]$, the sum will be

$\underline{\mathsf{Sign}_{\mathcal{TS}}(\mathsf{pk}_{\mathcal{TS}}, \mathsf{sk}_i, \mathsf{aux}_{\mathcal{TS}}, \mathcal{U}, \mu)}$

$\mathsf{ck} := H_1(\mathsf{pk}_{\mathcal{TS}}, \mu), \quad r_{i,1}, r_{i,2} \leftarrow D_\sigma$

$\boldsymbol{r}_i := [r_{i,1}\ r_{i,2}], \quad \rho_i \leftarrow \chi, w_i := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{r}_i \rangle$

$\mathsf{com}_i := \mathsf{Com}_{\mathsf{ck}}(w_i, \rho_i), \mathsf{ctx}_{\boldsymbol{r},i} \leftarrow \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_i)$

Compute NIZK proof $\pi_{\boldsymbol{r},i}$ of correct encryption $\xrightarrow{(\mathsf{ctx}_{\boldsymbol{r},i}, \mathsf{com}_i, \pi_{\boldsymbol{r},i})}$

$\xleftarrow{\{(\mathsf{ctx}_{\boldsymbol{r},j}, \mathsf{com}_j, \pi_{\boldsymbol{r},j})\}_{j \in \mathcal{U} \setminus \{i\}}}$

if any $\pi_{\boldsymbol{r},j}$ **is invalid**: **abort** (j)

$\mathsf{com} := \sum_{j \in \mathcal{U}} \mathsf{com}_j, \quad c := H_2(\mathsf{com}, \mathsf{pk}_{\mathcal{TS}}, \mu)$

$\mathsf{ctx}_{\boldsymbol{z}} := c \cdot \mathsf{ctx}_{\mathbf{s}} + \sum_{j \in \mathcal{U}} \mathsf{ctx}_{\boldsymbol{r},j}$

$\mathsf{ds}_i := \mathsf{TDec}(\mathsf{ctx}_{\boldsymbol{z}}, \mathsf{sk}_i, \mathcal{U})$

$\xrightarrow{(w_i, \rho_i, \mathsf{ds}_i)}$

$\xleftarrow{\{(w_j, \rho_j, \mathsf{ds}_j)\}_{j \in \mathcal{U} \setminus \{i\}}}$

if any $\mathsf{Open}(\mathsf{com}_j, w_j, \rho_j) = 0$: **abort** (j)

$\boldsymbol{z} := \mathsf{Comb}(\mathsf{ctx}_{\boldsymbol{z}}, \{\mathsf{ds}_j\}_{j \in \mathcal{U}}), \ \rho := \sum_{j \in \mathcal{U}} \rho_j$

if $\boldsymbol{z} = \perp$ with **abort** (j): **abort** (j)

return $\sigma := (c, \boldsymbol{z}, \rho)$

Fig. 7. Actively secure 2-round t-out-of-n threshold signature protocol for signer $\mathcal{S}_i$.

in the interval $[-tB, tB]$. However, suppose we sample from a discrete Gaussian distribution of standard deviation σ. In that case, each sample is with a high probability of 2-norm less than $2\sigma\sqrt{2N}$ for an integer vector of length $2N$, and the sum is bounded by $2\sigma\sqrt{2tN}$. Hence, the bounds $B_{\boldsymbol{z}}$ and B_ρ must be decided based on the distribution of choice, and the concrete choice of parameters and distribution impacts the security and efficiency overall.

When rejection sampling is removed, the signatures might leak information about the secret key, but this can be prevented by increasing the per-signature randomness $\boldsymbol{r}$ or limiting the number of signatures performed by the same key. We get optimal parameters if the key is used only once, as the key has high entropy and only leaks a few bits of information per signature. A recent analysis by Agrawal et al. [ASY22] using Rényi divergence shows that leakage scales with $\sqrt{Q}$ where Q is the number of signatures, and hence, we can keep the bounds on $\boldsymbol{r}$ small when limiting the number of signatures.

Looking at the key generation, each signer first executes the interactive key generation for the underlying encryption scheme, which has communication size $|\mathsf{DKGen}|$. Each signer sends a ring element of size $N \log_2 q$ bits. Each partial signing public key is of size $N \log_2 q$ bits. It also sends the ciphertext and a zero-knowledge proof, which we denote the sizes by $|\mathsf{ctx}|$ and $|\pi_{\mathbf{s}}|$ bits, respectively.

In the signature protocol, each party sends a ciphertext and a commitment of size $|\mathsf{ctx}|$ and $|\mathsf{com}|$, respectively, in addition to a zero-knowledge proof of size $|\pi_{\boldsymbol{r}}|$. They furthermore send values w_i of size $N \log_2 q$, opening randomness ρ_i of size $|\rho|$ and partial decryptions ds_i of size $N \log_2 q$.

5.3 Proof of Security

We prove security of our protocol by constructing an algorithm $\mathcal{B}$ interacting with an active adversary $\mathcal{A}$. Note we assume a rushing adversary, so honest users always send their messages first.

Theorem 3. *Let* $\mathsf{R\text{-}LWE}_{N,q,B_{\boldsymbol{r}}}$ *be* $\epsilon_{\mathsf{R\text{-}LWE}}$*-hard. Let* Q_S*,* Q_H *denote the number of signing queries and the total queries made to* H_0*,* H_1*, and* H_2 *respectively. Let* ℓ *be the bit-length of the output of* H_0*. Let* $\mathcal{E}$ *be* $\epsilon_{\text{IND-CPA}}$ *secure. Finally, let proof systems* $\Pi_{\boldsymbol{x}}$ *for the relations* $\mathcal{R}_{\boldsymbol{x}}$ *be* $\epsilon_{\mathsf{ZK},\boldsymbol{x}}$*, and* $\epsilon_{\mathsf{extract},\boldsymbol{x}}$*-secure. Then, the actively secure* t*-out-of-*n *threshold signature scheme* $\mathcal{TS}$*, described in Section 5 and depicted in Figure 7, is* $\mathsf{ts\text{-}uf\text{-}cma}$ *secure when* H_0 *is modeled as a programmable random oracle and* H_1 *and* H_2 *are modeled as random oracles. The advantage of adversary* $\mathcal{A}$ *is:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{TS}}(\mathcal{A}) \leq\ & e(Q_H + Q_S + 1)\big(|\mathcal{H}| \cdot \epsilon_{\mathsf{ZK},\mathbf{s}} + |\mathcal{H}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{ZK},\boldsymbol{r}} + |\mathcal{C}| \cdot \epsilon_{\mathsf{extract},\mathbf{s}} \\ & + |\mathcal{C}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{ZK},\boldsymbol{r}} + \frac{|\mathcal{C}|(Q_H + Q_S)}{2^\ell} + (Q_H + Q_S) \cdot \epsilon_{\mathsf{td}} \\ & + 2 \cdot \epsilon_{\text{IND-CPA}} + \epsilon_{\mathsf{R\text{-}LWE}} + \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{A})\big).\end{aligned}$$

Proof. We prove this through a series of hybrids written out in full detail. The simulator $\mathcal{B}$ is communicating with a challenger $\mathcal{D}$, and we show that a forgery by $\mathcal{A}$ can be used by $\mathcal{B}$ to answer $\mathcal{D}$. This time, however, $\mathcal{D}$ targets a variant of the signature scheme described in Section 4.1 where c is derived using the commitment to $\boldsymbol{w}$. Hence when queried for a message, $\mathcal{D}$ outputs $\sigma = (c, \boldsymbol{z}, \rho)$.

$\boldsymbol{G}_0$*:* The first game corresponds to the real world. Specifically, $\mathcal{B}$ follows the protocol according to the description, and $\mathcal{A}$ interacts with $\mathcal{B}$ arbitrarily. The oracles H_0,H_1, and H_2 are simulated using tables $\mathcal{HT}_0$, $\mathcal{HT}_1$, and $\mathcal{HT}_2$. Since we assume a rushing adversary, $\mathcal{B}$ sends its messages first.

When $\mathcal{A}$ outputs a forgery $(\sigma^* = (c^*, \boldsymbol{z}^*, \rho^*), \mu^*)$, $\mathcal{B}$ aborts if $\mu^* \in \mathcal{M}$. If not, $\mathcal{B}$ derives the commitment key $\mathsf{ck}^* := H_1(\mathsf{pk}_{\mathcal{TS}}, \mu^*)$, computes $w^* := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z}^* \rangle - c^* y$ and $\mathsf{com}^* := \mathsf{com}_{\mathsf{ck}^*}(w^*; \rho^*)$. If the challenge $c^* \neq H_2(\mathsf{com}^*, \mathsf{pk}_{\mathcal{TS}}, \mu^*)$, $\mathcal{B}$ aborts, otherwise halts with output (σ^*, μ^*) and $\mathcal{A}$ succeeds.

If the real world is indistinguishable from the programmable random oracle model, then the random oracle simulation is perfect, $\mathcal{B}$'s behavior is exactly the

same as in the unforgeability experiment, and we get that

$$\Pr[\mathbf{G}_0] = \mathsf{Adv}^{\mathsf{ts\text{-}uf\text{-}cma}}_{\mathcal{TS}}(\mathcal{A}).$$

G_1: In this game $\mathcal{B}$ changes how non-interactive zero-knowledge proofs are answered for honest parties. $\mathcal{B}$ executes the protocol the same as $\mathbf{G}_0$, but instead of honestly generating $\pi_{\mathbf{s},i}$ for $i \in \mathcal{H}$, $\pi_{\boldsymbol{r},i}$ for $i \in \mathcal{H}_{\mathcal{U}}$, $\mathcal{B}$ uses the corresponding honest-verifier zero-knowledge simulators, $\mathsf{Sim}_{\mathbf{s}}$ and $\mathsf{Sim}_{\boldsymbol{r}}$ for relations $\mathcal{R}_{\mathbf{s}}$ and $\mathcal{R}_{\boldsymbol{r}}$ respectively where $\pi_{\mathbf{s},i} := \mathsf{Sim}(\boldsymbol{a}_{\mathcal{TS}}, y_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s},i}, B_{\mathbf{s}})$ and $\pi_{\boldsymbol{r},i} := \mathsf{Sim}(\boldsymbol{a}_{\mathcal{TS}}, \mathsf{com}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\boldsymbol{r},i}, B_{\boldsymbol{r}})$. $\mathcal{B}$ follows the remaining parts of the protocol honestly. $\mathbf{G}_0$ and $\mathbf{G}_1$ is indistinguishable by HVZK of the NIZKs:

$$\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_0] \leq |\mathcal{H}| \cdot \epsilon_{\mathsf{ZK},\mathbf{s}} + |\mathcal{H}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{ZK},\boldsymbol{r}}.$$

G_2: $\mathbf{G}_2$ is the same as $\mathbf{G}_1$ except $\mathcal{B}$ uses the extractability properties of the NIZKs to learn the signing key shares and the signature randomness encrypted by parties in $\mathcal{C}$. During key generation, after receiving $\pi_{\mathbf{s},j}$ for $j \in \mathcal{C}$, $\mathcal{B}$ calls the extractor $\mathsf{s}_j := \mathsf{Extract}_{\mathbf{s}}(\pi_{\mathbf{s},j}; \boldsymbol{a}, y_j, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s},j}, B_{\mathbf{s}})$. Similarly, during signing after receiving $\pi_{\boldsymbol{r},i}$, $\mathcal{B}$ computes $\boldsymbol{r}_i := \mathsf{Extract}_{\boldsymbol{r}}(\pi_{\boldsymbol{r},i}; \boldsymbol{a}_{\mathcal{TS}}, \mathsf{com}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\boldsymbol{r},i}, B_{\boldsymbol{r}})$. If any of the extractions fails, $\mathcal{B}$ aborts.

By assumption, the extractor $\mathsf{Extract}_{\boldsymbol{x}}$ for $\mathcal{R}_{\boldsymbol{x}}$ is efficiently computable. If $\mathcal{B}$ does not abort due to a failed extraction, $\mathbf{G}_2$ and $\mathbf{G}_3$ is indistinguishable for $\mathcal{A}$. $\mathcal{A}$'s advantage in $\mathbf{G}_3$ is bounded by the sum of independent extraction failures:

$$\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1] \leq |\mathcal{C}| \cdot \epsilon_{\mathsf{extract},\mathbf{s}} + |\mathcal{C}_{\mathcal{U}}| \cdot \epsilon_{\mathsf{extract},\boldsymbol{r}}.$$

G_3: $\mathbf{G}_3$ is exactly the same as $\mathbf{G}_2$ except $\mathcal{B}$ embeds a trapdoor to the commitment key ck with high probability. Then the forgery by $\mathcal{A}$ must be for an honestly generated commitment key ck.

$\mathcal{B}$ initially generates only a single commitment key $\mathsf{ck} \leftarrow S_{\mathsf{ck}}$. $\mathcal{B}$ keeps a trapdoor table $\mathcal{TDT}$ similarly to other random oracles throughout the protocol. When there is a query for H_1 for a new message-public key pair, with probability ϕ, $\mathcal{B}$ samples a trapdoor $(\mathsf{ck}, \mathsf{td})$ by invoking TCGen, updates the corresponding entry in $\mathcal{TDT}$ to td and updates the corresponding entry in $\mathcal{HT}_2$ to ck. Otherwise $\mathcal{B}$ uses a freshly sampled $\mathsf{ck} \leftarrow S_{\mathsf{ck}}$.

When $\mathcal{A}$ queries H_1, $\mathcal{B}$ sets the flag *bad* and aborts if $\mathcal{TDT}[\mathsf{pk}_{\mathcal{TS}}, \mu] = \perp$. Otherwise, $\mathcal{B}$ obtains the trapdoor td. Then, for $j \in \mathcal{H}_{\mathcal{U}}$, $\mathcal{B}$ samples $r_{j,1}, r_{j,2}$ and computes $\boldsymbol{r}_j$,w_j according to $\mathsf{Sign}_{\mathcal{TS}}$. Furthermore, $\mathcal{B}$ samples commitments com_j by invoking TCom on input td. The rest of the first round continues as in $\mathbf{G}_2$. When $\mathcal{B}$ has received messages from all users, it checks if the proofs for $\boldsymbol{r}_j$ verify and computes the challenge and derives randomness $\rho_j \leftarrow \mathsf{Eqv}_{\mathsf{ck}}(\mathsf{td}, \mathsf{com}_j, w_j)$ for each $j \in \mathcal{H}_{\mathcal{U}}$. It then continues with the rest as before. When $\mathcal{A}$ sends a forgery $(\sigma^* := (c^*, \boldsymbol{z}^*, \rho^*), \mu^*)$, then $\mathcal{B}$ repeats the steps of $\mathbf{G}_2$. At the final step, if $\mathcal{TDT}[\mathsf{pk}_{\mathcal{TS}}, \mu^*] \neq \perp$, $\mathcal{B}$ aborts.

If $\mathcal{B}$ does not abort, then $\mathsf{ck}^* = H_1(\mathsf{pk}_{\mathcal{TS}}, \mu^*)$ and the simulated $\pi_{\boldsymbol{r},j}$ must verify for each honest part $j \in \mathcal{H}_{\mathcal{U}}$. This follows from the fact that the simulation

is only successful if the oracle uses the trapdoor commitment for all but one query to H_1 and uses the predefined ck for the one associated with forgery, i.e., *bad* is not set. The trapdoor commitment scheme ensures that:

$$\Pr[\mathbf{G}_3] \geq \phi^{Q_H+Q_S} \cdot (1-\phi) \cdot \Pr[\mathbf{G}_2] - (Q_H+Q_S) \cdot \epsilon_{\mathsf{td}}.$$

By setting $\phi = (Q_H+Q_S)/(Q_H+Q_S+1)$ we get

$$\Pr[\mathbf{G}_3] \geq \frac{\Pr[\mathbf{G}_2]}{e(Q_H+Q_S+1)} - (Q_H+Q_S) \cdot \epsilon_{\mathsf{td}},$$

where $(1/(1+1/(Q_H+Q_S)))^{Q_H+Q_S} \geq 1/e$ when $Q_H+Q_S \geq 0$.

$\boldsymbol{G}_4$*:* In this hybrid $\mathcal{B}$ changes how ciphertexts are computed similar to $\mathbf{G}_1$ in the passive case. $\mathcal{B}$ fixes some index $i' \in \mathcal{H}$ which is also in $\mathcal{H}_{\mathcal{U}}$. $\mathcal{B}$ computes $\mathsf{ctx}_{\mathbf{s},i'} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, 0)$ during key generation and $\mathsf{ctx}_{\boldsymbol{r},i'} := \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, 0)$ during signing. The rest of the game is as it is in $\mathbf{G}_4$. Since $\boldsymbol{r}_j$ and $\mathbf{s}_j$ for $j \in \mathcal{C}_{\mathcal{U}}$ are extracted in $\mathbf{G}_2$, $\mathcal{B}$ can compute $\boldsymbol{z} = c\sum_{i\in\mathcal{U}} \mathbf{s}_i + \sum_{i\in\mathcal{U}} \boldsymbol{r}_i$, which is the combined signature hence the threshold decryption can be simulated. Similar to the $\mathbf{G}_1$ in the passive case, we can now reduce to the indistinguishability of $\mathcal{E}$:

$$|\Pr[\mathbf{G}_4] - \Pr[\mathbf{G}_3]| \leq 2 \cdot \epsilon_{\text{IND-CPA}}.$$

$\boldsymbol{G}_5$*:* In this hybrid $\mathcal{B}$ changes how $h_{y,i'}$ is computed. $\mathcal{B}$ sends a random $h_{y,i'} \in \{0,1\}^{\ell_1}$ as the commitment $y_{i'}$ then programs H_0 such that $H_0(i', y_{i'}) = h_{y,i'}$. If programming fails, $\mathcal{B}$ aborts. Then the only difference in $\mathcal{A}'s$ view is how $h_{y,i'}$ is computed. By the oracle assumption, the distributions are indistinguishable between these two games hence from the view of the adversary $\mathcal{A}$, $\mathbf{G}_4$ and $\mathbf{G}_5$ are indistinguishable as long as $\mathcal{B}$ does not abort during programming. Thus we can bound the advantage of $\mathcal{A}$ in distinguishing $\mathbf{G}_4$ and $\mathbf{G}_5$ by the failure probability of programming of H_0:

$$|\Pr[\mathbf{G}_5] - \Pr[\mathbf{G}_4]| \leq \frac{|\mathcal{C}|(Q_H+Q_S)}{2^{\ell_1}}.$$

$\boldsymbol{G}_6$*:* Now, $\mathcal{B}$ removes its reliance on the secret key $\mathbf{s}$ for signing similar to $\mathbf{G}_3$ in the passive case. During key generation, $\mathcal{B}$ initializes the signing challenger $\mathcal{D}$ with the parameters of the combined signature and queries the oracle to obtain public keys, then sets $\mathsf{crs} = \boldsymbol{a}_{\mathcal{TS}}$. Instead of honestly computing $y_{i'}$, $\mathcal{B}$ derives $y_{i'} = y - \sum_{j\in[n]\setminus\{i'\}} y_j$ where y is received from $\mathcal{D}$ and continues the rest of key generation as in $\mathbf{G}_5$.

For $i \in \mathcal{H}_{\mathcal{U}} \setminus \{i'\}$, signing starts as $\mathbf{G}_5$. When a signing query for μ comes, $\mathcal{B}$ commits to a random $w_{i'}$ as part of the first message then forwards μ to $\mathcal{D}$ to obtain $(c, \boldsymbol{z}, \rho)$. $\mathcal{B}$ then computes $w := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z}\rangle - cy$ and uses $\boldsymbol{z}$ for simulation to obtain ds_i. For $j \in \mathcal{U} \setminus \{i'\}$, w_j is computed the same. $\mathcal{B}$ then derives $w_{i'} = w - \sum_{j\in\mathcal{U}\setminus\{i'\}} w_j$ and equivocates the commitment to get the randomness $\rho_{i'} := \mathsf{Eqv}(\mathsf{td}, \mathsf{com}_{i'}, w_{i'})$ using trapdoor td. Otherwise, signing proceeds as $\mathbf{G}_5$.

Since $\boldsymbol{a}_{\mathcal{TS}}, y$ received from $\mathcal{D}$ is a valid R-LWE instance, the distribution of $a_{\mathcal{TS}}$ therefore $y_{i'}$ in $\mathbf{G}_6$ is same as $\mathbf{G}_5$ by linearity of operations the derived $y_{i'} = \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}'_{i'} \rangle$ for an unknown $\mathbf{s}'_{i'}$.

$\mathcal{B}$ now has to fix the signature and its shares in a way consistent with the first round of communication in signing. Since $\mathcal{D}$ uses the variant that derives c based on a commitment to w, it is possible to obtain consistent $(c, \boldsymbol{z})$ such that $w := \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{z} \rangle - cy$. The remaining difference in $\mathcal{A}$'s view is the $w_{i'}$, which is indistinguishable from uniform the way it is computed. Using the same argument in $\mathbf{G}_3$ of passive, an adversary distinguishing between $w_{i'}$ in $\mathbf{G}_5$ and $\mathbf{G}_6$ can be used as a distinguisher for R-LWE. We then have:

$$|\Pr[\mathbf{G}_6] - \Pr[\mathbf{G}_5]| \leq \epsilon_{\mathsf{R\text{-}LWE}}.$$

The signatures in $\mathbf{G}_7$ are independent of secret key material defined as part of the protocol and solely depend on the signature received from the challenger $\mathcal{D}$. A forgery against the $\mathcal{TS}$ scheme is then a forgery against the underlying signature scheme. If $\mathcal{A}$ outputs a forgery $(c^*, \boldsymbol{z}^*, \rho^*), \mu^*$, $\mathcal{B}$ can submit the same forgery to $\mathcal{D}$ as a valid forgery for the underlying signature. Hence if $\mathcal{A}$ can output a forgery at the end of $\mathbf{G}_7$, it can be used to break the uf-cma security of the underlying scheme, the advantage if $\mathcal{A}$ can then be bounded as:

$$|\Pr[\mathbf{G}_6]| \leq \mathsf{Adv}^{\mathsf{uf\text{-}cma}}(\mathcal{A}).$$

This concludes the proof. □

6 Instantiating the NIZKPoKs

There are four main relations of the signature protocol (including the homomorphic encryption subprotocols) to be proven in zero-knowledge. Here we define the exact relations that have to be proven and show how we can instantiate corresponding zero-knowledge proofs using proofs of shortness and linear relations.

- The relation $\mathcal{R}_{\mathsf{sk}}$ during $\mathsf{KGen}_{\mathcal{E}}$ defines the correctness of partial key shares as part of the key generation. For fixed public parameters B_{tern} and p, the common reference string $\mathsf{crs} = a_{\mathcal{E}}$, public b_i, $\{b_{i,j}\}_{i\in[n]}$, and public commitments to secret *short* s_i, e_i and secret $s_{i,j}$, $e_{i,j}$ the relation shows (1) $b_i = a_{\mathcal{E}} s_i + pe_i$; (2) $\mathsf{com}_i, \mathsf{com}'_i$ are commitments to s_i, e_i; and (3) $\{\mathsf{com}_{i,j}\}_{i\in[n]}$, $\{\mathsf{com}'_{i,j}\}_{i\in[n]}$ are commitments to $\{s_{i,j}\}_{i\in[n]}, \{e_{i,j}\}_{i\in[n]}$ such that $\{s_{i,j}\}_{i\in[n]}$, $\{e_{i,j}\}_{i\in[n]}$ are valid t-of-n secret sharings of s_i and e_i; (4) $\{s_{i,j}\}_{j\in[n]}$ are correct t-out-of-n shares of s_i; (5) $\{e_{i,j}\}_{j\in[n]}$ are correct t-out-of-n shares of e_i; (6) $\{b_{i,j}\}_{j\in[n]}$ is correctly computed from $s_{i.j}$ and $e_{i,j}$. More formally:

$$\mathcal{R}_{\mathsf{sk}} := \left\{ (x, w) \;\middle|\; \begin{array}{c} x := (b_i, \{b_{i,j}\}, \mathsf{com}_i, \mathsf{com}'_i, \{\mathsf{com}_{i,j}\}_{i\in[n]}, \{\mathsf{com}'_{i,j}\}_{i\in[n]}) \wedge \\ w := (s_i, \rho_i, e_i, \rho'_i, \{s_{i,j}\}_{j\in[n]}, \{\rho_{i,j}\}_{j\in[n]}, \{e_{i,j}\}_{j\in[n]}, \{\rho_{i,j}\}_{j\in[n]}): \\ \\ \|s_i\|, \|e_i\| \leq B_{\mathsf{tern}} \;\wedge\; b_i = a_{\mathcal{E}} s_i + pe_i \;\wedge\; \forall j \; b_{i,j} = a_{\mathcal{E}} s_{i,j} + pe_{i,j} \\ \wedge\; b_j = \mathsf{Rec}_{t,n}(\{b_{i,j}\}) \;\wedge\; s_i = \mathsf{Rec}_{t,n}(\{s_{i,j}\}) \;\wedge\; e_i = \mathsf{Rec}_{t,n}(\{e_{i,j}\}) \\ \wedge\; \mathsf{Open}(\mathsf{com}_i, s_i, \rho_i) \;\wedge\; \mathsf{Open}(\mathsf{com}'_i, e_i, \rho'_i) \;\wedge \\ \forall i \; \mathsf{Open}(\mathsf{com}_{i,j}, s_{i,j}, \rho_{i,j}) \;\wedge\; \forall j \; \mathsf{Open}(\mathsf{com}'_{i,j}, e_{i,j}, \rho'_{i,j}) \end{array} \right\}.$$

The knowledge of s_i and e_i, and consequently $s_{i,j}$ and $e_{i,j}$ can be proven based on b_i and $b_{i,j}$ using proofs of linearity and shortness once commitments to s_i, e_i, and $e_{i,j}$ are available. For which we modify the distributed key generation algorithm given in Figure 2 as follows: Party $\mathcal{P}_i$ as part of π_i commits to these secret values and broadcasts them as part of the proof. For partial shares that are sent through private channels, $\mathcal{P}_i$ also sends opening i.e. randomness $\rho'_{i,j}$ corresponding to the commitment to $e_{i,j}$ as well.

It is also possible to show s_i and e_i are correctly secret shared. Once knowledge of s_i, e_i, $s_{i,j}$ and $\boldsymbol{e}_{i,j}$ is proven, since b_i and $b_{i,j}$ is public, it is possible to check if every subset of size t of $b_{i,j}$ can be used to reconstruct b_i.

– The relation $\mathcal{R}_{\mathsf{ds}}$ during TDec defines the correctness of the partial decryptions where the correct secret key share sk_i was used to generate the public decryption share $\boldsymbol{d}_i$ for a given ciphertext $\mathsf{ctx} = (u, v)$. For fixed parameters $\lambda_i, p, B_{\mathsf{TDec}}$, common reference string $\mathsf{crs} = (a_{\mathcal{E}}, b_{\mathcal{E}})$ and public commitments to secret sk_i, E_i the relation shows: (1) The secret error E_i has norm smaller than B_{TDec}, (2) $\boldsymbol{d}_i$ is correctly computed with respect to $\lambda_i, \mathsf{sk}_i, u, p, E_i$:

$$\mathcal{R}_{\mathsf{ds}} := \left\{ (x, w) \middle| \begin{array}{c} x := (\boldsymbol{d}_i, \mathsf{ctx}, \mathsf{com}_{\mathsf{sk},i}, \mathsf{com}_{E,i}) \ \wedge \ w := (\mathsf{sk}_i, \rho_{\mathsf{sk},i}, E_i, \rho_{E,i}) : \\ \|E_i\|_\infty \leq B_{\mathsf{TDec}} \ \wedge \ \boldsymbol{d}_i = \lambda_i \mathsf{sk}_i u + pE_i \ \wedge \\ \mathsf{Open}(\mathsf{com}_{\mathsf{sk},i}, \mathsf{sk}_i, \rho_{\mathsf{sk},i}) \ \wedge \ \mathsf{Open}(\mathsf{com}_{E,i}, E_i, \rho_{E,i}) \end{array} \right\}.$$

Since the commitment to the secret key shares are also included as part of the distributed key generation, it is trivial to prove the above statements using the proof of linear relations as before, once the commitments to E_i are available. For which we instantiate the proof $\pi_{\mathsf{ds},i}$ as a proof of linear relation to show d is correctly computed alongside a commitment to E_i.

– The relation $\mathcal{R}_{\mathbf{s}}$ for $\mathsf{KGen}_{\mathcal{TS}}$ proves that the short $\mathbf{s}_i$ was both used to compute the public key share y_i and was encrypted in $\mathsf{ctx}_{\mathbf{s},i}$. For a publicly fixed parameter $B_{\mathbf{s}}$, the common reference string $\mathsf{crs} = (\boldsymbol{a}_{\mathcal{TS}}, \mathsf{pk}_{\mathcal{E}})$ and public y_i and the commitment to the secret $\mathbf{s}_i$, the relation $\mathcal{R}_{\mathbf{s}}$ shows:(1) secret $\mathbf{s}_i$ has norm smaller than $B_{\mathbf{s}}$ (2) $\mathbf{s}_i$ is the same $\mathbf{s}_i$ used for the calculation of y_i (3) $\mathsf{ctx}_{\mathbf{s},i}$ is the encryption of the $\mathbf{s}_i$ using $\mathsf{pk}_{\mathcal{E}}$.

$$\mathcal{R}_{\mathbf{s}} := \left\{ (x, w) \middle| \begin{array}{c} x := (\boldsymbol{a}_{\mathcal{TS}}, y_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\mathbf{s},i}, B_{\mathbf{s}}) \ \wedge \ w := \mathbf{s}_i \ \wedge \\ \|\mathbf{s}_i\| \leq B_{\mathbf{s}} \ \wedge \ y_i = \langle \boldsymbol{a}_{\mathcal{TS}}, \mathbf{s}_i \rangle \ \wedge \ \mathsf{ctx}_{\mathbf{s},i} = \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \mathbf{s}_i) \end{array} \right\}.$$

Since $\mathbf{s}_i$ is boundend, the relation above is once again a linear relation with respect to $\boldsymbol{a}$. y_i and $\mathbf{s}_i$. Only non-trivial part is to show the knowledge of $\mathbf{s}_i$ as plaintext, which we discuss at the end of this section. So $\pi_{\mathbf{s},i}$ includes commitment to $\mathbf{s}_i$ and proof linear relation and plaintext knowledge for $\mathbf{s}_i$.

– The relation $\mathcal{R}_{\boldsymbol{r}}$ for $\mathsf{Sign}_{\mathcal{TS}}$ proves that a bounded signature randomness $\boldsymbol{r}_i$ was both committed to in com_i and encrypted in $\mathsf{ctx}_{\boldsymbol{r},i}$. For a publicly fixed parameter $B_{\boldsymbol{r}}$, the common reference string $\mathsf{crs} = (\boldsymbol{a}_{\mathcal{TS}}, \mathsf{pk}_{\mathcal{E}})$, public com_i,$\mathsf{ctx}_{\boldsymbol{r},i}$ for secret w_i,$\boldsymbol{r}_i$, ρ_i the relation $\mathcal{R}_{\boldsymbol{r}}$ shows: (1) randomness $\boldsymbol{r}_i$ has a norm smaller than $B_{\boldsymbol{r}}$ (2) $\boldsymbol{r}_i$ is the same $\boldsymbol{r}_i$ used for w_i and therefore the

commitment com_i using randomness ρ_i (3) $\mathsf{ctx}_{\boldsymbol{r},i}$ is the encryption of the $\boldsymbol{r}_i$.

$$\mathcal{R}_{\boldsymbol{r}} := \left\{ (x,w) \middle| \begin{array}{c} x = (\boldsymbol{a}_{\mathcal{TS}}, \mathsf{com}_i, \mathsf{pk}_{\mathcal{E}}, \mathsf{ctx}_{\boldsymbol{r},i}, B_{\boldsymbol{r}}) \ \wedge \ w = (w_i, \boldsymbol{r}_i, \rho_i) \\ \wedge \ \|\boldsymbol{r}_i\| \leq B_{\boldsymbol{r}} \ \wedge \ \mathsf{ctx}_{\boldsymbol{r},i} = \mathsf{Enc}(\mathsf{pk}_{\mathcal{E}}, \boldsymbol{r}_i) \\ \wedge \ w_i = \langle \boldsymbol{a}_{\mathcal{TS}}, \boldsymbol{r}_i \rangle \ \wedge \ \mathsf{com}_i = \mathsf{Com}_{\mathsf{ck}}(w_i, \rho_i) \end{array} \right\}.$$

$\mathcal{R}_{\boldsymbol{r}}$ is almost exactly the same as $\mathcal{R}_{\mathbf{s}}$ with the exception of knowledge of openings to trapdoor commitments. For which, we do the same treatment to the protocol where $\pi_{\boldsymbol{r},i}$ includes a proof of linear relation with respect to $\boldsymbol{r}_i$ alongside a proof of plaintext knowledge which we will show next. Both $\mathcal{R}_{\mathbf{s}}$ and $\mathcal{R}_{\boldsymbol{r}}$ rely on the availability of a proof of plaintext knowledge where a given ciphertext corresponds to, for which, we introduce a final relation $\mathcal{R}_{\mathsf{Enc}}$ which also can be instantiated through proofs of linear relations since we use BGV as our underlying encryption scheme.

– The $\mathcal{R}_{\mathsf{Enc}}$ shows that, for fixed public parameters B_{Enc}, p, and a common reference string $\mathsf{crs} = (a_{\mathcal{E}}, b_{\mathcal{E}})$ and a statement consisting of ciphertext $\mathsf{ctx} = (u, v)$ and public commitments com_r, $\mathsf{com}_{e'}$, $\mathsf{com}_{e''}$, com_m to short secrets r, e', e'', m. Then the proof shows that (1) commitments open to suitably short values r, e', e'', m, respectively, such that (2) $u = a_{\mathcal{E}} r + pe'$ and (3) $v = b_{\mathcal{E}} r + pe'' + m$:

$$\mathcal{R}_{\mathsf{Enc}} := \left\{ (x,w) \middle| \begin{array}{c} x := (\mathsf{ctx}, \mathsf{com}_r, \mathsf{com}_{e'}, \mathsf{com}_{e''}, \mathsf{com}_m) \ \wedge \ w := (r, e', e'', m, \rho_r, \rho_{e'}, \rho_{e''}, \rho_m) : \\ \|r\|_\infty, \|e'\|_\infty, \|e''\|_\infty \leq B_{\mathsf{Enc}} \ \wedge u = a_{\mathcal{E}} r + pe' \ \wedge \ v = b_{\mathcal{E}} r + pe'' + m \ \wedge \\ \mathsf{Open}(\mathsf{com}_r, r, \rho_r) \ \wedge \ \mathsf{Open}(\mathsf{com}_{e'}, e', \rho_{e'}) \ \wedge \\ \mathsf{Open}(\mathsf{com}_{e''}, e'', \rho_{e''}) \ \wedge \ \mathsf{Open}(\mathsf{com}_m, m, \rho_m) \ \wedge \ \|m\| \leq p \end{array} \right\}.$$

The proof of plaintext knowledge proofs in both $\mathcal{R}_{\mathbf{s}}$ and $\mathcal{R}_{\boldsymbol{r}}$ then can be instantiated as a proof of linear relation where $\mathbf{s} = m$ and $\boldsymbol{r} = m$ with different B_{Enc} respectively. $\pi_{\mathbf{s}}$ already includes a commitment to $\mathbf{s}_i$, so we can add commitments to remaining encryption randomnesses as well as a linear relation proof for $\mathsf{ctx}_{\mathbf{s},i}$ to complete. $\pi_{\boldsymbol{r},i}$ however is a bit non trivial as commitments sent as part of the message are trapdoor commitments.

Trapdoor commitments and normal commitments can be instantiated using the commitment scheme by Damgård et al. [BDL+18, DOTT21], and the proofs of linearity and exact proof of shortness can be instantiated using the zero-knowledge protocols by Lyubashevsky et al. [BDL+18, BLNS21, LNP22] combined with the Katsumata-transform [Kat21].

7 Example Uses Cases and Performance

To estimate the practicality of our actively secure scheme we give example parameters for $(3, 5)$-threshold signatures in three different settings: (1) where a signature is only produced once (1-SIG), (2) where at most β signatures are produced (β-SIG), for some moderate β, and (3) where up to 2^{64} signatures is issued (∞-SIG), as per NIST recommendations. Schemes secure for issuing one signature may be of interest for cryptocurrency applications, and there has

been ongoing discussion in the context of the NIST post-quantum cryptography standardization effort[4], especially with regard to SPHINCS+ [Kö22], about applications where the assumption of a bounded number of signatures might be reasonable.

For simplicity, we let the distribution χ_{tern} be the uniform ternary distribution in each of the three cases. We use the threshold scheme described in Section 3 as the underlying additive homomorphic encryption scheme and the homomorphic trapdoor commitment scheme by Damgård et al. [DOTT21].

We emphasize that these are rough estimates, and a more careful analysis is needed before this scheme is ready for real-world use. The main point of this exercise is to showcase that we can achieve practical signatures and verification keys using our techniques. We acknowledge that the total amount of communication in the protocol is quite large, and we will provide more details on communication in the full version of the paper. Furthermore, this section assumes trusted setup and only focuses on the signing protocol, not distributed key generation. We summarize the results in Table 1.

Comm.	1-SIG	1-PK	β-SIG	β-PK	∞-SIG	∞-PK
Size	8.5 KB	2.6 KB	10.4 KB	3.1 KB	46.6 KB	13.6 KB

Table 1. Estimated sizes of $(3, 5)$-threshold signatures SIG and public keys PK for settings: 1) where the signature is only produced once, 2) where a signature is produced at most $\beta = 365$ times, and 3) where a signature can be produced $\infty = 2^{64}$ times. These parameters achieve 128 bits R-SIS / R-LWE security using the lattice estimator [APS15].

7.1 One-Time Signatures

The simplest case is when each key is only used to create a single signature before it is discarded and never used again. One such setting is Bitcoin transactions, where some funds are tied to a specific public key. When a new transaction is performed, the remaining funds are sent to a new address tied to a different public key owned by the same user(s). The setup can be done in advance, independent of the blockchain and future transactions, and one key is used for each transaction. This leads to smaller keys and signatures to minimize on-chain data.

With no rejection sampling, publishing a single signature leaks minimal information about the secret key. Agrawal et al. [ASY22] show that the leakage in each signature grows linearly in the square root of the number of signatures produced, but since we only output a single signature, we can keep the parameters identical to when rejection sampling is performed.

All signing key shares are ternary, which means that even secrets of absolute norm 1 should ensure that the R-SIS and R-LWE problems are hard. The R-SIS

[4] See https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/LUczQNCw7HA.

problem is hard when the logarithm of the ℓ_2 norm of the secret is less than $2\sqrt{N \log_2 q \log_2 \delta}$, see [MR09], and we get more roughly 128 bits of security when $\delta \approx 1.005$, and better when it is smaller. The ring dimension N must be a power of two, so we set $N = 1024$. Then we let elements of $\boldsymbol{r}$ be sampled from a Gaussian distribution D_σ with standard deviation $\sigma = \nu \cdot \sqrt{\gamma}$, where $\gamma = 128 + (N \log_2 q)/(\log_2(2\eta + 1))$.

Each signer must prove in zero-knowledge that these bounds are satisfied to ensure protocol correctness. The most efficient exact zero-knowledge proofs used today are the proof systems by Lyubashevsky et al. [BLNS21, LNP22], allowing us to prove the exact maximum norm of the secret values $\boldsymbol{r}$. The latter proof system is improved by Aranha et al. [ABGS23] and extended by Hough et al. [HSS23] to large values using bit decomposition.

We let the ℓ_2 norm of $\boldsymbol{z}$ be $B = 2 \cdot \sigma \cdot \sqrt{2 \cdot t \cdot N}$ in our signature scheme. We finally set $\nu = 16$ (so that $|\bar{\mathcal{C}}| > 2^{128}$) to get $\sigma \approx 2^{10.9}$, $B_z \approx 2^{18.1}$ and $q \approx 2^{20}$. This leads to more than 128 bits of R-SIS security when inserting the parameters into the equation above and more than 128 bits of R-LWE security according to the LWE-estimator [APS15] when revealing only one signature per key.

The absolute norm of each coefficient in $\boldsymbol{z}$ is with high probability bounded by $4 \cdot \sqrt{t} \cdot \sigma$, and this leads to $\boldsymbol{z}$ being approximately $2N \log_2(4 \cdot \sqrt{t} \cdot \sigma)$ bits. Following [ENS+23], the standard deviation to sample ρ should be roughly $1.17 \cdot \sqrt{q}$ when using NTRU trapdoors, which means that ρ is of size $3 \cdot N \log_2(4.68 \cdot \sqrt{t \cdot q})$ bits. This leads to a signature 1-SIG of size 8.5 KB with the given parameters. The verification key 1-PK is of size $N \log_2 q$ bits ($\boldsymbol{a}_{\mathcal{TS}}$ can be generated by a random oracle as in Dilithium [DKL+18]), resulting in a verification key of 2.6 KB.

The communication consists of commitments, ciphertexts, and NIZKs being sent in the signing protocol. Similar to Damgård et al. [DOTT21] we use the commitment scheme by Baum et al. [BDL+18] but with NTRU trapdoors [ENS+23], and instantiate this with module dimension one to match our security assumptions. We then need to increase the dimension and the moduli of the encryption scheme to be able to encrypt partial signatures, and use noise drowning to achieve secure threshold decryption. This leads to communication of ≈ 750 KB per party.

7.2 Bounded Number of Signatures

Another interesting setting is where a service is used at most once a day each year, and a signature is required to use the service. An example is FIDO authentication[5] where the signing key is secret shared over several devices to ensure both that the user can log in despite lost devices and at the same time that no one can impersonate the user even two devices are stolen. Hence, we must make sure that the signing key does not leak when up to 365 signatures are produced.

Following a similar analysis as above, where we extend the standard deviation to $\sigma = \nu \cdot \sqrt{\gamma \cdot 365}$ (see [ASY22, Theorem 4.1]) to ensure that the signature does not leak too much information when producing at most 365 signatures. We keep

[5] See https://fidoalliance.org for more details.

$N = 1024$, and get $\sigma \approx 2^{15.2}$, $B \approx 2^{22.5}$ and $q \approx 2^{24}$ to ensure at least 128 bits of security with respect to the hardness of R-SIS and R-LWE. The signatures are of size 10.4 KB and the verification key is of size 3.1 KB.

Since q is similar in this setting as in the previous, the size of intermediate communication within the signing protocol is essentially the same as above.

7.3 Unbounded Number of Signatures

In general, it is undesirable to upper-bound the number of signatures that can be produced with a signing key before it is not secure to use it anymore. One reason for this is that it is hard to keep a state over a longer time, and if the signing is running in a virtual environment it might be re-booted from a backup with an older state and a fresh counter, and hence, end up producing more signatures than initially recommended. In practice, we often upper limit the number of signatures by 2^{64}, or some other number that is close to the capacity of what modern computers can compute when choosing concrete parameters for certain security levels. Hence, we expand the number of signatures and use the square-root bound as above to compute the parameters for general-use signatures.

This leads to $\sigma = \nu \cdot \sqrt{\gamma \cdot 2^{64}} \approx 2^{44}$ for $\nu = 14$ when $N = 2048$ and get $B \approx 2^{51.9}$. Hence, we need to set $q \approx 2^{53}$ to get correctness and 128 bits of security. We get signatures of size 46.7 KB, and verification keys of 13.6 KB.

Since q is approximately twice the number of bits, the dimension of the lattice needs to be doubled, and we estimate the intermediate communication to be approximately four times larger compared to the previous settings, leading to communication of ≈ 3 MB per party.

8 Extensions

There are several possible considerations for increased efficiency and security:

Compression. The most efficient lattice-based schemes in the literature use compression techniques to reduce the size of communication. The compression rate is chosen based on the hardness of the underlying assumptions so that one gives an approximate relation instead with a fine-tuned reduction to a problem of the appropriate hardness level, see for example Kyber [SAB+20] or Dilithium [LDK+20] for details. These techniques can potentially reduce the size of public keys and signatures in our case as well, in addition to reducing the communication on our protocol where the security is much higher to ensure the correctness of the distributed decryption protocol.

Reducing communication via an optimistic approach. Assuming non-malicious behavior we can omit sending w_j and $\pi_{\mathsf{ds},j}$ for $j \in \mathcal{U}$, and send only ds_j and ρ_j for the second round of signing. If signature verification fails then each signer sends w_j and $\pi_{\mathsf{ds},j}$ in a third round as proof of correct computation. In an honest execution, this saves $t \cdot (|w_j| + |\pi_{\mathsf{ds},j}|)$ bits per party, significantly

reducing the overall communication. For one-time signatures, roughly 400 KB of 750 KB per party communication is due to π_{ds}, which then can be removed.

Removing trapdoor commitments for pre-processing. The pre-processing in Boschini et al. [BTT22] to remove the trapdoor commitments is an immediate extension to our protocol, as the committed values in both protocols are similar to the ones in [DOTT21]. However, the application is a bit trickier and results in an increase in communication. Unlike their work, the commitments in our protocol are to the encryptions of per-signature randomness. This would require each commitment to have an associated NIZK of correct encryption, increasing the communication size for a set of commitments. This also raises the non-trivial question of computing NIZKs for random linear combinations of bounded values, which gives an extensive overhead to the protocol.

Acknowledgments

The work of Kamil Doruk Gur and Jonathan Katz was supported in part by NSF award CNS-2154705. We thank Shuichi Katsumata, Mary Maller, and Thomas Prest for pointing out a mistake in our parameter estimates in an earlier version of this work and for helping us correct them.

References

ABGS23. Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In *ACM CCS 2023*, page 1467–1481. ACM, 2023.

APS15. Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

ASY22. Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-Optimal Lattice-Based Threshold Signatures, Revisited. In Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

BD10. Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 201–218, Zurich, Switzerland, 2010. Springer, Heidelberg, Germany.

BDL+18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 368–385, Amalfi, Italy, 2018. Springer.

BGG+18. Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596, Santa Barbara, CA, USA, 2018. Springer, Heidelberg, Germany.

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.

BKP13. Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 218–236, Banff, AB, Canada, 2013. Springer.

BLNS21. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. More efficient amortization of exact zero-knowledge proofs for LWE. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021: 26th European Symposium on Research in Computer Security, Part II*, volume 12973 of *Lecture Notes in Computer Science*, pages 608–627, Darmstadt, Germany, October 4–8, 2021. Springer, Heidelberg, Germany.

BP23. Luís T. A. N. Brandão and René Peralta. NIST first call for multi-party threshold schemes, January 2023.

BS23. Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part I*, volume 14438 of *Lecture Notes in Computer Science*, pages 371–404, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany.

BTT22. Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 276–305, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

CATZ24. Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Partially non-interactive two-round lattice-based threshold signatures, 2024. Available at https://eprint.iacr.org/2024/467.

CCL+20. Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 266–296, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

CGG+20. Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Gio-

vanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1769–1787, Virtual Event, USA, November 9–13, 2020. ACM Press.

CGRS23. Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. Practical Schnorr threshold signatures without the algebraic group model. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 743–773, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.

Che23. Yanbo Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 716–747, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany.

CS19. Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *Lecture Notes in Computer Science*, pages 128–153, Oxford, UK, December 16–18, 2019. Springer, Heidelberg, Germany.

CSS+22. Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Efficient threshold FHE with application to real-time systems, 2022. Available at https://eprint.iacr.org/2022/1625.

DDEK+23. Morten Dahl, Daniel Demmler, Sarah El Kazdadi, Arthur Meyre, Jean-Baptiste Orfila, Dragos Rotaru, Nigel P. Smart, Samuel Tap, and Michael Walter. Noah's ark: Efficient threshold-fhe using noise flooding. In *Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC '23, page 35–46, New York, NY, USA, 2023. Association for Computing Machinery.

DJN+20. Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bæksvang Østergaard. Fast threshold ECDSA with honest majority. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 382–400, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany.

DKL+18. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, 2018. https://tches.iacr.org/index.php/TCHES/article/view/839.

DKLs19. Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy*, pages 1051–1066, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press.

DLN+21. Julien Devevey, Benoît Libert, Khoa Nguyen, Thomas Peters, and Moti Yung. Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Prac-*

tice of Public Key Cryptography, Part I, volume 12710 of *Lecture Notes in Computer Science*, pages 659–690, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany.

DOTT21. Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 99–130, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany.

dPEK+23. Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Melissa Rossi, and Markku-Juhani Saarinen. Raccoon: A side-channel secure signature scheme, 2023.

dPKM+24. Rafael del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions, 2024. Eurocrypt 2024, to appear. Available at https://eprint.iacr.org/2024/184.

EKT24. Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. Two-round threshold signature from algebraic one-more learning with errors, 2024. Available at https://eprint.iacr.org/2024/496.

ENS+23. Thomas Espitau, Thi Thu Quyen Nguyen, Chao Sun, Mehdi Tibouchi, and Alexandre Wallet. Antrag: Annular NTRU trapdoor generation - making mitaka as secure as falcon. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 3–36, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany.

FH20. Masayuki Fukumitsu and Shingo Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model. In Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang, editors, *ProvSec 2020: 14th International Conference on Provable Security*, volume 12505 of *Lecture Notes in Computer Science*, pages 45–64, Singapore, November 29 – December 1, 2020. Springer, Heidelberg, Germany.

FSZ22. Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 1109–1123, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

GG18. Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1179–1194, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

GKPV10. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 230–240, Tsinghua University, Beijing, China, January 5–7, 2010. Tsinghua University Press.

GKS24. Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. PQCrypto, 2024.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

HSS23. Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. Concrete NTRU security and advances in practical lattice-based electronic voting, 2023. Available at https://eprint.iacr.org/2023/933.

Kat21. Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure NIZKs. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 580–610, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

KG20. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Heidelberg, Germany.

Kö22. Stefan Kölbl. A note on SPHINCS$^+$ parameter sets, 2022. Available at https://eprint.iacr.org/2022/1725.

LDK+20. Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

Lin17. Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 613–644, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

Lin24. Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. *IACR Communications in Cryptology*, 1(1), 2024.

LN18. Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1837–1854, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plan**c**con. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 71–101, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in*

Computer Science, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

Mic02. Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *43rd Annual Symposium on Foundations of Computer Science*, pages 356–365, Vancouver, BC, Canada, November 16–19, 2002. IEEE Computer Society Press.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

MR09. Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer, 2009.

Pei10. Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

RST+22. Dragos Rotaru, Nigel P. Smart, Titouan Tanguy, Frederik Vercauteren, and Tim Wood. Actively secure setup for SPDZ. *Journal of Cryptology*, 35(1):5, January 2022.

SAB+20. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

TPCZ23. Guofeng Tang, Bo Pang, Long Chen, and Zhenfeng Zhang. Efficient lattice-based threshold signatures with functional interchangeability. *IEEE Transactions on Information Forensics and Security*, 18:4173–4187, 2023.